RSL15 Hardware Reference

M-20872-006
September 2023

© SCILLC, 2023
Previous Edition © 2022

Table of Contents

Page

RSL15 Hardware Reference 1

1. Introduction 39

1.1 Purpose 39

1.2 Intended Audience 39

1.3 Document Conventions 39

1.4 Further Reading 41

2. System Overview 42

2.1 System Architecture 42

2.2 Security System Architecture 44

2.2.1 Integrated IoT Cybersecurity Platform 44

2.2.2 Secure Boot with Root of Trust 44

2.2.3 Arm Lifecycle State Management 44

2.2.4 Application and Data Cryptographic Services 44

2.2.5 TrustZone Arm Cortex-M33 Peripherals 44

2.3 Radio System Architecture 45

2.3.1 RF Subsystem 45

2.3.2 Bluetooth Low Energy 46

2.4 Power 46

2.4.1 Power Management Unit (PMU) Battery Monitoring and Resets 47

2.4.2 Power Modes Overview 47

2.5 Clocking 48

2.5.1 RTC 48

2.6 Memory 48

2.7 Interfaces 49

2.7.1 General Purpose Input/Output (GPIO) 49

2.7.2 Communications Interfaces ... 49

2.7.2.1 I2C ... 49

2.7.2.2 LIN ... 50

2.7.2.3 PCM ... 50

2.7.2.4 PWM ... 50

2.7.2.5 SPI ... 50

2.7.2.6 UART ... 50

2.7.3 Sensor Support Interfaces ... 50

2.7.3.1 Analog Comparator ... 50

2.7.3.2 SAR ADC ... 50

2.7.3.3 Pulse Counter ... 50

2.7.3.4 Low Speed ADC ... 51

2.7.3.5 Simple DAC ... 51

2.7.3.6 Current Source ... 51

2.7.3.7 ULP Data Acquisition Subsystem ... 51

2.8 Peripherals ... 51

2.8.1 Activity Counter ... 51

2.8.2 Asynchronous Clock Counter ... 52

2.8.3 CRC ... 52

2.8.4 DMA Controller ... 52

2.8.5 Flash Copier ... 52

2.8.6 Time of Flight ... 52

2.8.7 Timers ... 53

2.8.8 Watchdog ... 53

2.9 Firmware Reference Libraries ... 53

2.10 SoC Identification ... 53

2.11 SoC Identification Registers ... 54

2.11.0.1 AHBREGS_CHIP_ID_NUM ... 54

3. Arm Cortex-M33 Processor ... 55

3.1 Introduction ... 55

3.2 Debug Port ... 55

3.3 Private Peripherals ... 57

3.3.1 SysTick ... 57

3.3.2 Nested Vector Interrupt Controller (NVIC) ... 58

3.3.2.1 Interrupt Priority Registers ... 61

3.3.3 System Control ... 62

3.3.4 Memory Protection Unit ... 62

3.3.5 Security Attribution Unit ... 63

3.3.6 Debug Controller ... 63

3.3.6.1 Halting Debug Configuration and Status ... 63

3.3.6.2 Debug Monitor Configuration ... 64

3.3.6.3 Arm Cortex-M33 Processor Core Register Access ... 65

3.3.6.4 Debug Fault Status Register ... 65

3.3.7 Software Interrupts ... 66

3.3.8 Floating-Point Extension ... 66

3.4 Private Peripherals Registers ... 66

3.4.1 SysTick Registers ... 66

3.4.1.1 SysTick_CTRL ... 67

3.4.1.2 SysTick_LOAD ... 67

3.4.1.3 SysTick_VAL ... 67

3.4.1.4 SysTick_CALIB ... 68

3.4.2 NVIC Registers ... 68

3.4.2.1 NVIC_ISER0 70

3.4.2.2 NVIC_ISER1 72

3.4.2.3 NVIC_ICER0 74

3.4.2.4 NVIC_ICER1 77

3.4.2.5 NVIC_ISPR0 79

3.4.2.6 NVIC_ISPR1 81

3.4.2.7 NVIC_ICPR0 83

3.4.2.8 NVIC_ICPR1 86

3.4.2.9 NVIC_IABR0 88

3.4.2.10 NVIC_IABR1 91

3.4.2.11 NVIC_ITNS0 94

3.4.2.12 NVIC_ITNS1 97

3.4.2.13 NVIC_IPR0 99

3.4.2.14 NVIC_IPR1 99

3.4.2.15 NVIC_IPR2 100

3.4.2.16 NVIC_IPR3 100

3.4.2.17 NVIC_IPR4 100

3.4.2.18 NVIC_IPR5 100

3.4.2.19 NVIC_IPR6 100

3.4.2.20 NVIC_IPR7 101

3.4.2.21 NVIC_IPR8 101

3.4.2.22 NVIC_IPR9 101

3.4.2.23 NVIC_IPR10 101

3.4.2.24 NVIC_IPR11 101

3.4.2.25 NVIC_IPR12 102

3.4.2.26 NVIC_IPR13 102

3.4.2.27 NVIC_IPR14 102

3.4.2.28 NVIC_ISER0_NS 102

3.4.2.29 NVIC_ISER1_NS 102

3.4.2.30 NVIC_ICER0_NS 102

3.4.2.31 NVIC_ICER1_NS 102

3.4.2.32 NVIC_ISPR0_NS 102

3.4.2.33 NVIC_ISPR1_NS 102

3.4.2.34 NVIC_ICPR0_NS 103

3.4.2.35 NVIC_ICPR1_NS 103

3.4.2.36 NVIC_IABR0_NS 103

3.4.2.37 NVIC_IABR1_NS 103

3.4.2.38 NVIC_IPR0_NS 103

3.4.2.39 NVIC_ICPR1_NS 103

3.4.2.40 NVIC_IPR2_NS 103

3.4.2.41 NVIC_IPR3_NS 103

3.4.2.42 NVIC_IPR4_NS 103

3.4.2.43 NVIC_IPR5_NS 103

3.4.2.44 NVIC_IPR6_NS 103

3.4.2.45 NVIC_IPR7_NS 103

3.4.2.46 NVIC_IPR8_NS 104

3.4.2.47 NVIC_IPR9_NS 104

3.4.2.48 NVIC_IPR10_NS 104

3.4.2.49 NVIC_IPR11_NS 104

3.4.2.50 NVIC_IPR12_NS 104

3.4.2.51 NVIC_IPR13_NS 104

3.4.2.52 NVIC_IPR14_NS 104

3.4.3 Memory Protection Unit Registers ... 104
3.4.3.1 MPU_TYPE ... 105
3.4.3.2 MPU_CTRL ... 105
3.4.3.3 MPU_RNR ... 105
3.4.3.4 MPU_RBAR ... 106
3.4.3.5 MPU_RLAR ... 106
3.4.3.6 MPU_RBAR1 ... 107
3.4.3.7 MPU_RLAR1 ... 107
3.4.3.8 MPU_RBAR2 ... 107
3.4.3.9 MPU_RLAR2 ... 107
3.4.3.10 MPU_RBAR3 ... 107
3.4.3.11 MPU_RLAR3 ... 108
3.4.3.12 MPU_MAIR0 ... 108
3.4.4 Security Attribution Unit Registers ... 108
3.4.4.1 SAU_CTRL ... 108
3.4.4.2 SAU_TYPE ... 109
3.4.4.3 SAU_RNR ... 109
3.4.4.4 SAU_RBAR ... 109
3.4.4.5 SAU_RLAR ... 109
3.4.4.6 SAU_SFSR ... 109
3.4.4.7 SAU_SFAR ... 109
3.4.5 Debug Control Block Registers ... 110
3.4.5.1 DEBUG_DHCSR ... 110
3.4.5.2 DEBUG_DCRSR ... 111
3.4.5.3 DEBUG_DCRDR ... 113
3.4.5.4 DEBUG_DEMCR ... 113

3.4.5.5 DEBUG_DAUTHCTRL ... 113

3.4.5.6 DEBUG_DSCSR ... 113

3.4.6 Software Interrupts Registers ... 114

3.4.6.1 SIG_STIR ... 114

3.4.7 Floating Point Extension Registers ... 114

3.4.7.1 FPE_FPCCR ... 114

3.4.7.2 FPE_FPCAR ... 115

3.4.7.3 FPE_FPDSCR ... 116

3.4.7.4 FPE_MVFR0 ... 116

3.4.7.5 FPE_MVFR1 ... 116

3.4.7.6 FPE_MVFR2 ... 116

4. Arm TrustZone CryptoCell-312 Security IP ... 117

4.1 Cryptographic Features ... 117

4.2 True Random Number Generator (TRNG) ... 117

4.3 Power Configuration ... 118

4.4 TrustZone ... 119

4.4.1 Secure Execution ... 119

4.4.2 Non-Secure Execution ... 120

4.4.3 Non-Secure Use of Secured Features ... 120

4.5 Arm TrustZone CryptoCell-312 Registers ... 120

4.5.0.1 SYSCTRL_CRYPTOCELL_PWR_CFG ... 121

4.5.0.2 SYSCTRL_NS_ACCESS_PERIPH_CFG0 ... 122

4.5.0.3 SYSCTRL_NS_ACCESS_PERIPH_CFG1 ... 125

4.5.0.4 SYSCTRL_NS_ACCESS_RAM_CFG0 ... 126

4.5.0.5 SYSCTRL_NS_ACCESS_RAM_CFG1 ... 127

4.5.0.6 SYSCTRL_DEU_STATUS ... 128

4.5.0.7 SYSCTRL_DEU_DATA 128

4.5.0.8 SYSCTRL_PROD_STATUS 128

4.5.0.9 SYSCTRL_CC_DCU_EN0 129

4.5.0.10 SYSCTRL_CC_DCU_EN1 130

4.5.0.11 SYSCTRL_CC_DCU_EN2 130

4.5.0.12 SYSCTRL_CC_DCU_EN3 130

4.5.0.13 SYSCTRL_CC_DCU_LOCK0 130

4.5.0.14 SYSCTRL_CC_DCU_LOCK1 131

4.5.0.15 SYSCTRL_CC_DCU_LOCK2 132

4.5.0.16 SYSCTRL_CC_DCU_LOCK3 132

4.5.0.17 SYSCTRL_CC_STATUS 132

4.5.0.18 SYSCTRL_CC_FEATURES_CTRL 134

5. RF Front-End 140

5.1 Overview 140

5.2 RFFE Interface 142

5.2.1 RFFE PowerUp 142

5.2.2 VDDPA Dynamic Control 143

5.2.3 RFFE Data Steaming 145

5.2.3.1 7.2.6.3.1 Phase ADC and RSSI data streaming 145

5.2.3.2 I/Q data streaming 147

5.3 RFFE System Resources 147

5.3.1 48 MHz Crystal Oscillator 149

5.3.2 Radio Clocks 150

5.3.3 Registers 150

5.3.4 FSM and Global Configuration 151

5.3.4.1 TX and RX Start Sequence Example 152

5.3.4.2 RX Timeout 153

5.3.4.3 MAC Timers 153

5.3.4.4 Protocol Timers 153

5.3.4.5 Time Stamps 153

5.3.4.6 Delta Time 154

5.3.4.7 Commands 154

5.3.4.8 Protocol Timer, Delta Timer, and Command Example 155

5.3.5 RFFE GPIO 156

5.3.6 Interrupts 158

5.3.7 FIFOs 159

5.3.7.1 Accessing the FIFOs and FIFO Status 159

5.4 Packet handling 161

5.4.1 Packet Format 161

5.4.1.1 Preamble 162

5.4.1.2 Pattern 162

5.4.1.3 Packet Length 162

5.4.1.4 Address 163

5.4.1.5 Multi-Frame 163

5.4.1.6 CRC 163

5.4.2 Bluetooth Low Energy Direct Test Mode 164

5.4.3 Bluetooth Low-Energy Direction Finding 164

5.4.3.1 TX Mode 164

5.4.3.2 RX Mode 165

5.4.3.3 CTE Sampling 165

5.4.3.4 Antenna Switching Pattern 165

5.4.4 Coding 166

5.4.4.1 Manchester Coding ... 166

5.4.4.2 Data-Whitening ... 167

5.4.4.3 IEEE 802.15.4 Bit to Chip Conversion ... 167

5.4.4.4 New IEEE 802.15.4 Chip to Bit Decoder ... 167

5.4.4.5 Linear to Frequency ... 167

5.4.4.6 Convolutional Codes ... 168

5.4.4.7 Puncturing Codes ... 168

5.4.4.8 4FSK coding ... 168

5.4.4.9 Bluetooth Low Energy Long Range ... 169

5.4.4.10 Miscellaneous and Encoding Order ... 170

5.5 Radio Configuration ... 171

5.5.1 Data Rate ... 171

5.5.1.1 Fractional Data Rate ... 171

5.5.2 Central Frequency ... 171

5.5.3 Channels ... 172

5.5.4 Pulse Shape ... 172

5.5.5 Modulation Index ... 173

5.5.6 Interpolator ... 174

5.5.7 Channel Filter Configuration ... 174

5.5.8 Phase and RSSI Fractional Decimation ... 175

5.5.9 Carrier Recovery ... 176

5.5.9.1 Rough Carrier Recovery ... 176

5.5.9.2 Fine Carrier Recovery ... 176

5.5.9.3 Carrier Recovery Boundaries ... 177

5.5.9.4 RSSI Detection ... 178

5.5.9.5 Delay Line Synchronization ... 180

5.5.10 Matched Filtering ... 181

5.5.11 Clock and Data-Rate Recovery ... 181

5.5.12 Decision ... 183

5.5.12.1 Viterbi Algorithm ... 183

5.5.13 RSSI Filtering and AGC ... 183

5.5.13.1 Peak Detector ... 185

5.5.13.2 RSSI and Peak-Detector Combined AGC Strategy ... 185

5.6 Continuous Wave (CW) Configuration ... 186

5.7 Direct Test Mode (DTM) ... 186

5.8 Power Amplifier and Ramp Up/Down Sequence ... 187

5.9 RF Front-End Registers ... 188

5.9.0.1 RF_REG00 ... 197

5.9.0.2 RF_REG01 ... 199

5.9.0.3 RF_REG02 ... 201

5.9.0.4 RF_REG03 ... 202

5.9.0.5 RF_PADS_03 ... 206

5.9.0.6 RF_PADS_47 ... 207

5.9.0.7 RF_CENTER_FREQ ... 208

5.9.0.8 RF_PADS_89 ... 208

5.9.0.9 RF_REG08 ... 209

5.9.0.10 RF_CODING ... 210

5.9.0.11 RF_PACKET_HANDLING ... 212

5.9.0.12 RF_SYNC_PATTERN ... 214

5.9.0.13 RF_REG0C ... 214

5.9.0.14 RF_PACKET_EXTRA ... 215

5.9.0.15 RF_CRC_POLYNOMIAL ... 217

5.9.0.16 RF_CRC_RST 217

5.9.0.17 RF_REG10 218

5.9.0.18 RF_REG11 219

5.9.0.19 RF_TX_PULSE_SHAPE_1 220

5.9.0.20 RF_TX_PULSE_SHAPE_2 221

5.9.0.21 RF_TX_PULSE_SHAPE_3 221

5.9.0.22 RF_TX_PULSE_SHAPE_4 222

5.9.0.23 RF_FRONTEND 222

5.9.0.24 RF_RX_PULSE_SHAPE 223

5.9.0.25 RF_REG18 224

5.9.0.26 RF_REG19 226

5.9.0.27 RF_REG1A 228

5.9.0.28 RF_REG1B 229

5.9.0.29 RF_RSSI_CTRL 231

5.9.0.30 RF_REG1D 233

5.9.0.31 RF_AGC_LUT1 234

5.9.0.32 RF_AGC_LUT2 234

5.9.0.33 RF_AGC_LUT3 235

5.9.0.34 RF_AGC_LUT4 235

5.9.0.35 RF_AGC_LUT5 236

5.9.0.36 RF_AGC_ATT1 237

5.9.0.37 RF_AGC_ATT2 238

5.9.0.38 RF_REG25 239

5.9.0.39 RF_BIAS_0_2 240

5.9.0.40 RF_BIAS_3_6 241

5.9.0.41 RF_BIAS_7_9 242

5.9.0.42 RF_BIAS_10_12 243

5.9.0.43 RF_REG2A 244

5.9.0.44 RF_PLL_CTRL 245

5.9.0.45 RF_DLL_CTRL 248

5.9.0.46 RF_REG2D 251

5.9.0.47 RF_REG2E 253

5.9.0.48 RF_XTAL_CTRL 254

5.9.0.49 RF_SUBBAND 258

5.9.0.50 RF_REG31 259

5.9.0.51 RF_DEMOD_CTRL 262

5.9.0.52 RF_REG33 264

5.9.0.53 RF_REG34 266

5.9.0.54 RF_BLE_LR 267

5.9.0.55 RF_REG36 269

5.9.0.56 RF_PROT_TIMER 271

5.9.0.57 RF_CTE_OPTS 273

5.9.0.58 RF_PT_DELTA_0 274

5.9.0.59 RF_PT_DELTA_1 275

5.9.0.60 RF_CTE_IF 275

5.9.0.61 RF_CTE_CTRL 277

5.9.0.62 RF_AGC_ADVANCED 277

5.9.0.63 RF_DATA_STREAMING 278

5.9.0.64 RF_REVISION 279

5.9.0.65 RF_FSM_CTRL 280

5.9.0.66 RF_IQFIFO_STATUS 283

5.9.0.67 RF_TXFIFO 285

5.9.0.68 RF_RXFIFO 285

5.9.0.69 RF_IQFIFO 285

5.9.0.70 RF_REG45 286

5.9.0.71 RF_DESER_STATUS 286

5.9.0.72 RF_BLE_AEC_CCM 287

5.9.0.73 RF_IRQ_STATUS 288

5.9.0.74 RF_RSSI_MIN_MAX 289

5.9.0.75 RF_REG4A 289

5.9.0.76 RF_FEI 290

5.9.0.77 RF_REG4C 290

5.9.0.78 RF_ANALOG_INFO 291

5.9.0.79 RF_SAMPLE_RSSI 292

5.9.0.80 RF_RSSI_THERM 292

5.9.0.81 RF_LUT_ANTENNA_ARRAY_1 293

5.9.0.82 RF_LUT_ANTENNA_ARRAY_2 294

5.9.0.83 RF_LUT_ANTENNA_ARRAY_3 295

5.9.0.84 RF_LUT_ANTENNA_ARRAY_4 296

5.9.0.85 RF_REG50 297

5.9.0.86 RF_REG51 298

5.9.0.87 RF_REG52 298

5.9.0.88 RF_REG53 299

5.9.0.89 RF_REG54 299

5.9.0.90 RF_REG55 299

5.9.0.91 RF_REG56 299

6. Bluetooth Low Energy Baseband Controller 301

6.1 Overview 301

6.2 System Resources 304

6.2.1 Exchange Memory 304

6.2.2 Filtering Lists and Device Filtering 306

6.2.3 AES-128 On Software Request 307

6.2.4 RF Test Modes 308

6.2.5 Angle of Arrival (AoA) and Angle of Departure (AoD) Support 309

6.3 Baseband Controller Interface 311

6.3.1 Baseband Controller Clock 311

6.3.2 WLAN Coexistence 312

6.3.3 BB Controller Interrupts 314

6.3.4 Baseband Counters and Timers 315

6.3.4.1 Active Mode 315

6.3.4.2 Low Power Mode 317

6.3.5 Bluetooth Low Energy Stack Deep Sleep Mode 322

6.3.5.1 Bluetooth Low Energy Stack Deep Sleep Mode Registers 325

6.3.6 Baseband Controller Registers and Non-CTE Antenna Definition 325

6.4 Baseband Controller Interface Registers 325

6.4.0.1 BBIF_CTRL 326

6.4.0.2 BBIF_STATUS 326

6.4.0.3 BBIF_COEX_CTRL 327

6.4.0.4 BBIF_COEX_STATUS 327

6.4.0.5 BBIF_COEX_INT_CFG 328

6.4.0.6 BBIF_COEX_INT_STATUS 329

6.4.0.7 BBIF_ID_NUM 330

6.5 Baseband Timer Registers 330

6.5.0.1 ACS_BB_TIMER_CTRL 330

6.6 Baseband Controller Registers 331

6.6.0.1 BB_RWBBCNTL 334

6.6.0.2 BB_VERSION 337

6.6.0.3 BB_RWBLEBCONF 337

6.6.0.4 BB_INTCNTL0 339

6.6.0.5 BB_INTSTAT0 340

6.6.0.6 BB_INTACK0 340

6.6.0.7 BB_INTCNTL1 340

6.6.0.8 BB_INTSTAT1 341

6.6.0.9 BB_INTACK1 342

6.6.0.10 BB_ACTFIFOSTAT 343

6.6.0.11 BB_CURRENTRXDESCPTR 344

6.6.0.12 BB_ETPR 345

6.6.0.13 BB_DEEPSLCNTL 345

6.6.0.14 BB_DEEPSLWKUP 346

6.6.0.15 BB_DEEPSLSTAT 346

6.6.0.16 BB_ENBPRESET 346

6.6.0.17 BB_FINECNTCORR 347

6.6.0.18 BB_CLKNCNTCORR 347

6.6.0.19 BB_DIAGCNTL 347

6.6.0.20 BB_DIAGSTAT 348

6.6.0.21 BB_DEBUGADDMAX 349

6.6.0.22 BB_DEBUGADDMIN 349

6.6.0.23 BB_ERRORTYPESTAT 349

6.6.0.24 BB_SWPROFILING 352

6.6.0.25 BB_RADIOCNTL0 353

6.6.0.26 BB_RADIOCNTL1 353

6.6.0.27 BB_RADIOCNTL2 355

6.6.0.28 BB_RADIOCNTL3 356

6.6.0.29 BB_RADIOPWRUPDN0 357

6.6.0.30 BB_RADIOPWRUPDN1 358

6.6.0.31 BB_RADIOPWRUPDN2 358

6.6.0.32 BB_RADIOPWRUPDN3 358

6.6.0.33 BB_RADIOTXRXTIM0 359

6.6.0.34 BB_RADIOTXRXTIM1 359

6.6.0.35 BB_RADIOTXRXTIM2 359

6.6.0.36 BB_RADIOTXRXTIM3 360

6.6.0.37 BB_SPIPTRCNTL0 360

6.6.0.38 BB_SPIPTRCNTL1 360

6.6.0.39 BB_SPIPTRCNTL2 361

6.6.0.40 BB_SPIPTRCNTL3 361

6.6.0.41 BB_AESCNTL 361

6.6.0.42 BB_AESKEY31_0 362

6.6.0.43 BB_AESKEY63_32 362

6.6.0.44 BB_AESKEY95_64 362

6.6.0.45 BB_AESKEY127_96 362

6.6.0.46 BB_AESPTR 362

6.6.0.47 BB_TXMICVAL 363

6.6.0.48 BB_RXMICVAL 363

6.6.0.49 BB_RFTESTCNTL 363

6.6.0.50 BB_RFTESTTXSTAT 364

6.6.0.51 BB_RFTESTRXSTAT 365

6.6.0.52 BB_TIMGENCNTL 365

6.6.0.53 BB_FINETIMTGT 365

6.6.0.54 BB_CLKNTGT1 365

6.6.0.55 BB_HMICROSECTGT1 366

6.6.0.56 BB_CLKNTGT2 366

6.6.0.57 BB_HMICROSECTGT2 366

6.6.0.58 BB_SLOTCLK 366

6.6.0.59 BB_FINETIMECNT 367

6.6.0.60 BB_ACTSCHCNTL 367

6.6.0.61 BB_STARTEVTCLKNTS 367

6.6.0.62 BB_STARTEVTFINECNTTS 368

6.6.0.63 BB_ENDEVTCLKNTS 368

6.6.0.64 BB_ENDEVTFINECNTTS 368

6.6.0.65 BB_SKIPEVTCLKNTS 368

6.6.0.66 BB_SKIPEVTFINECNTTS 368

6.6.0.67 BB_ADVTIM 369

6.6.0.68 BB_ACTSCANCNTL 369

6.6.0.69 BB_WPALCNTL 369

6.6.0.70 BB_WPALCURRENPTR 369

6.6.0.71 BB_SEARCH_TIMEOUT 370

6.6.0.72 BB_COEXIFCNTL0 370

6.6.0.73 BB_COEXIFCNTL1 373

6.6.0.74 BB_COEXIFCNTL2 373

6.6.0.75 BB_BLEMPRIO0 374

6.6.0.76 BB_BLEMPRIO1 374

6.6.0.77 BB_BLEMPRIO2 375

6.6.0.78 BB_RALCNTL 375

6.6.0.79 BB_RALCURRENTPTR 376

6.6.0.80 BB_RAL_LOCAL_RND 376

6.6.0.81 BB_RAL_PEER_RND 376

6.6.0.82 BB_DFCNTL0_1US 377

6.6.0.83 BB_DFCNTL0_2US 377

6.6.0.84 BB_DFCNTL1_1US 377

6.6.0.85 BB_DFCNTL1_2US 378

6.6.0.86 BB_DFCURRENTPTR 378

6.6.0.87 BB_DFANTCNTL 378

6.6.0.88 BB_DFIFCNTL 379

7. Clock Components 381

7.1 Overview 381

7.2 Clock Generation 383

7.2.1 RC Oscillator 383

7.2.2 48 MHz Crystal Oscillator 383

7.2.3 Standby RC Oscillator 384

7.2.4 32 kHz Crystal Oscillator 384

7.2.5 Debug Port Clock 386

7.3 Clock Generation Registers 387

7.3.0.1 ACS_RTC_CFG 387

7.3.0.2 ACS_RTC_CTRL 388

7.3.0.3 ACS_RTC_COUNT_THRES 389

7.3.0.4 ACS_RTC_COUNT 389

7.3.0.5 ACS_RTC_SECONDS 389

7.3.0.6 ACS_RTC_COUNT_LOAD 389

7.3.0.7 ACS_BB_TIMER_CTRL 389

7.4 Clock Distribution 390

7.4.1 System Clock (SYSCLK) 390

7.4.2 Standby Clock (STANDBYCLK) 390

7.4.3 Slow Clock (SLOWCLK) 391

7.4.4 Baseband Clock (BBCLK) 392

7.4.5 Real Time Clock (RTC) 392

7.4.5.1 RTC Output Control 394

7.4.6 User Clock (USRCLK) 394

7.4.7 Power Supply Clocks 395

7.4.8 Interface Clocks 395

7.5 Clock Distribution Registers 396

7.5.0.1 CLK_SYS_CFG 397

7.5.0.2 CLK_DIV_CFG0 398

7.5.0.3 CLK_DIV_CFG1 399

7.5.0.4 CLK_DIV_CFG2 400

7.5.0.5 ACS_RCOSC_CTRL 401

7.5.0.6 ACS_XTAL32K_CTRL 402

7.6 Clock Detector and System Monitor 404

7.7 Clock Detector Registers 404

7.7.0.1 ACS_CLK_DET_CTRL 405

8. Power Components 406

8.1 Power Management Unit 406

8.2 Power Supply Overview 406

8.2.1 Power Supply Inputs 407

8.2.1.1 Battery Supply Voltage (VBAT) 407

8.2.1.2 Digital Output Supply Voltage (VDDO) 408

8.2.2 Internal Power Supply Voltages 408

8.2.2.1 DC-DC Converter (VCC) 408

8.2.2.2 Internal Bandgap Reference Voltages 410

8.2.2.3 RF Supply Voltage 411

8.2.2.4 VDDPA 411

8.2.2.4.1 Connecting and Using VDDPA 412

8.2.2.4.2 Dynamic VDDPA Control 412

8.2.2.5 Digital Supply Voltages 414

8.2.2.6 Analog Supply Voltage (VDDA) 415

8.2.2.7 Charge Pump Supply Voltage (VDDCP) 416

8.2.2.8 Flash Memory Supply Voltage 416

8.3 Power Supply Registers 416

8.3.0.1 ACS_VCC_CTRL 417

8.3.0.2 ACS_BG_CTRL 419

8.3.0.3 ACS_VDDRF_CTRL 420

8.3.0.4 ACS_VDDPA_CTRL 421

8.3.0.5 ACS_VDDC_CTRL 422

8.3.0.6 ACS_VDDM_CTRL 423

8.3.0.7 ACS_VDDRET_CTRL 424

8.3.0.8 ACS_VDDCP_CTRL 424

8.3.0.9 ACS_VDDFLASH_CTRL 425

8.3.0.10 ACS_WEDAC_CTRL 426

8.4 Resets 427

8.5 Resets Registers 428

8.5.0.1 ACS_RESET_STATUS 429

8.6 Power Modes ... 431

8.6.1 Keeping the Debugger Connected in Low Power Modes ... 435

8.6.2 Run Mode ... 435

8.6.2.1 Idle Mode ... 435

8.6.3 Sleep Mode ... 435

8.6.3.1 Smart Sense Mode ... 437

8.6.4 Standby Mode ... 437

8.6.5 Entering a Low Power Mode ... 438

8.6.6 Waking Up from a Low Power Mode ... 439

8.6.7 Wakeup Sources ... 440

8.6.8 Adding a Low-Power Mode to an Existing Application ... 440

8.6.8.1 Setting Up the Application ... 440

8.6.8.2 Updating the Application's Control Flow ... 441

8.6.8.3 Configuring the Low-Power Module ... 443

8.6.8.4 Attempting to Enter a Low-Power State ... 446

8.6.8.5 Handling Wakeup Sources ... 449

8.6.8.6 Saving and Restoring RF Register States ... 450

8.6.8.7 Saving and Restoring Peripheral States ... 451

8.7 Power Modes Registers ... 452

8.7.0.1 SYSCTRL_WAKEUP_ADDR ... 453

8.7.0.2 ACS_PWR_MODES_CTRL ... 453

8.7.0.3 ACS_SLEEP_MODE_CFG ... 453

8.7.0.4 ACS_WAKEUP_CTRL ... 454

8.7.0.5 ACS_WAKEUP_CFG ... 457

8.7.0.6 ACS_WAKEUP_STATE ... 458

8.7.0.7 ACS_BOOT_CFG ... 459

8.7.0.8 ACS_BOOT_GP_DATA 460

8.7.0.9 ACS_PWR_CTRL 461

8.8 Timing 461

8.8.1 Startup Timing 461

8.8.2 Sleep Power Mode Related Timing 463

8.8.3 Standby Power Mode Related Timing 464

9. Memory 466

9.1 Architecture 466

9.1.1 Memory Instances 466

9.1.2 Memory Buses 469

9.1.3 Memory Arbitration 469

9.2 Memory Map and Usage 469

9.2.1 Flash Memory 471

9.2.1.1 Manually Configuring Memory Allocation 471

9.2.2 Data Memory 473

9.2.3 Other Memory Mapped Areas 474

9.2.3.1 Peripherals and Interfaces 474

9.2.3.2 Private Peripherals 474

9.3 Flash 474

9.3.1 Flash Characteristics 474

9.3.2 Flash Delays 475

9.3.3 Flash Memory Operations 475

9.3.4 Redundancy Sectors 480

9.3.5 Error-Correction Coding 481

9.4 Flash Memory Registers 483

9.4.0.1 FLASH_IF_CTRL 484

9.4.0.2 FLASH_MAIN_WRITE_UNLOCK ... 485

9.4.0.3 FLASH_MAIN_CTRL ... 485

9.4.0.4 FLASH_DELAY_CTRL ... 487

9.4.0.5 FLASH_CMD_CTRL ... 488

9.4.0.6 FLASH_IF_STATUS ... 489

9.4.0.7 FLASH_ADDR ... 495

9.4.0.8 FLASH_DATA ... 495

9.4.0.9 FLASH_NVR_WRITE_UNLOCK ... 495

9.4.0.10 FLASH_NVR_CTRL ... 495

9.4.0.11 FLASH_PATCH_ADDR ... 497

9.4.0.12 FLASH_COPY_CFG ... 497

9.4.0.13 FLASH_COPY_CTRL ... 498

9.4.0.14 FLASH_COPY_SRC_ADDR_PTR ... 498

9.4.0.15 FLASH_COPY_DST_ADDR_PTR ... 498

9.4.0.16 FLASH_COPY_WORD_CNT ... 499

9.4.0.17 FLASH_ECC_CTRL ... 499

9.4.0.18 FLASH_ECC_STATUS ... 500

9.4.0.19 FLASH_ECC_ERROR_ADDR ... 500

9.4.0.20 FLASH_ECC_UNCOR_ERROR_CNT ... 500

9.4.0.21 FLASH_ECC_COR_ERROR_CNT ... 501

9.4.0.22 FLASH_NVM_STATUS ... 501

9.4.0.23 FLASH_MAIN_MASK ... 501

9.4.0.24 FLASH_IF_ID_NUM ... 503

9.5 Memory Registers ... 504

9.5.0.1 SYSCTRL_CM33_LOOP_CACHE_CFG ... 504

9.5.0.2 SYSCTRL_ACCESS_ERROR ... 505

9.5.0.3 SYSCTRL_MEM_POWER_STARTUP 505

9.5.0.4 SYSCTRL_MEM_POWER_ENABLE 506

9.5.0.5 SYSCTRL_MEM_ACCESS_CFG 507

9.5.0.6 SYSCTRL_WAKEUP_ADDR 508

9.5.0.7 SYSCTRL_MEM_RETENTION_CFG 509

9.5.0.8 SYSCTRL_MEM_TIMING_CFG 509

10. General Purpose Input/Output 511

10.1 Overview 511

10.2 Functional Configuration 512

10.2.1 JTAG I/O Functional Configuration 516

10.2.2 Analog Test Output 516

10.3 Physical Configuration 516

10.3.1 nRESET Pull Resistor Configuration 517

10.3.2 JTAG I/O Pad Configuration 517

10.4 Direct Control 518

10.4.1 GPIO Interrupts 518

10.5 GPIO Registers 519

10.5.0.1 GPIO_CFG 521

10.5.0.2 GPIO_INPUT_DATA 526

10.5.0.3 GPIO_OUTPUT_DATA 527

10.5.0.4 GPIO_OUTPUT_DATA_SET 529

10.5.0.5 GPIO_OUTPUT_DATA_CLR 530

10.5.0.6 GPIO_DIR 531

10.5.0.7 GPIO_MODE 533

10.5.0.8 GPIO_INT_CFG 535

10.5.0.9 GPIO_INT_STATUS_S 537

10.5.0.10 GPIO_INT_STATUS 537

10.5.0.11 GPIO_INT_DEBOUNCE 538

10.5.0.12 GPIO_JTAG_SW_PAD_CFG 538

10.5.0.13 GPIO_SRC_SPI 539

10.5.0.14 GPIO_SRC_SPI_IO 540

10.5.0.15 GPIO_SRC_UART 543

10.5.0.16 GPIO_SRC_I2C 544

10.5.0.17 GPIO_SRC_PCM 545

10.5.0.18 GPIO_SRC_LIN 547

10.5.0.19 GPIO_SRC_NMI 548

10.5.0.20 GPIO_SRC_BB_RX 549

10.5.0.21 GPIO_SRC_BB_SPI 551

10.5.0.22 GPIO_SRC_BB_COEX 552

10.5.0.23 GPIO_SRC_BB_IQ_DATA 553

10.5.0.24 GPIO_SRC_BB_IQ_DATA_P 556

10.5.0.25 GPIO_SRC_RF_SPI 557

10.5.0.26 GPIO_SRC_RF_GPIO03 559

10.5.0.27 GPIO_SRC_RF_GPIO47 562

10.5.0.28 GPIO_SRC_RF_GPIO89 565

10.5.0.29 GPIO_SRC_RF_CTE 566

10.5.0.30 GPIO_SRC_ASCC 568

10.5.0.31 GPIO_SRC_EXTCLK 570

10.5.0.32 ACS_AOUT_CTRL 571

11. Communication Interfaces 575

11.1 I2C Interfaces 575

11.1.1 Slave Mode Specific Configuration 579

11.1.2 Master Mode Specific Configuration 580

11.1.3 Setting the System Clock 581

11.1.4 I2C Interrupts 581

11.1.4.1 Operation Using Manual Acknowledgment 581

11.1.4.2 Operation Using Automatic Acknowledgment 582

11.1.4.3 I2C Main Modes of Operation 582

11.2 LIN 588

11.2.1 LIN Responder Mode 589

11.2.2 Lin Commander Mode 589

11.3 Pulse Code Modulation (PCM) Interface 590

11.3.1 PCM Signal Configuration 592

11.3.1.1 Frame Signal Configuration and Timing 592

11.3.1.2 Data Serial Input and Output Configuration 595

11.3.2 PCM/I2S Data Interface 595

11.3.3 PCM Interrupt Configuration 596

11.3.4 I2S Configuration and Usage 597

11.4 PWM 598

11.4.1 ACS_PWM 599

11.5 Serial Peripheral Interfaces (SPI) 600

11.5.1 SPI Data Transfers 600

11.5.2 SPI Interrupts 605

11.5.3 SPI DMA Control 606

11.6 Universal Asynchronous Receiver-Transmitter (UART) Interfaces 607

11.6.1 UART Interrupts 608

11.7 Communication Interfaces Registers 609

11.7.1 I2C Registers 609

11.7.1.1 I2C_CFG 609

11.7.1.2 I2C_CTRL 613

11.7.1.3 I2C_ADDR_START 614

11.7.1.4 I2C_STATUS 615

11.7.1.5 I2C_TX_DATA 617

11.7.1.6 I2C_RX_DATA 617

11.7.1.7 I2C_RX_DATA_MIRROR 617

11.7.1.8 I2C_ID_NUM 618

11.7.2 LIN Registers 618

11.7.2.1 LIN_CFG 619

11.7.2.2 LIN_CTRL 619

11.7.2.3 LIN_ERROR 620

11.7.2.4 LIN_PID 621

11.7.2.5 LIN_DLB 621

11.7.2.6 LIN_DLBR 622

11.7.2.7 LIN_DATA 622

11.7.2.8 LIN_DATA_WORD0 623

11.7.2.9 LIN_DATA_WORD1 623

11.7.2.10 LIN_CHECKSUM 623

11.7.2.11 LIN_SYNCH 623

11.7.2.12 LIN_ID_NUM 623

11.7.3 PCM Registers 624

11.7.3.1 PCM_CTRL 624

11.7.3.2 PCM_CFG 624

11.7.3.3 PCM_STATUS 628

11.7.3.4 PCM_TX_DATA0 629

11.7.3.5 PCM_TX_DATA1 629

11.7.3.6 PCM_RX_DATA0 629

11.7.3.7 PCM_RX_DATA1 629

11.7.3.8 PCM_ID_NUM 629

11.7.4 PWM Registers 630

11.7.4.1 PWM_PERIOD 630

11.7.4.2 PWM_CTRL 630

11.7.4.3 PWM_OFFSET 631

11.7.4.4 PWM_HIGH 632

11.7.4.5 PWM_ID_NUM 632

11.7.4.6 ACS_PWM_AO_CFG 632

11.7.4.7 ACS_PWM_AO_CTRL 632

11.7.4.8 ACS_PWM_AO_COUNT 633

11.7.5 SPI Registers 633

11.7.5.1 SPI_CFG 633

11.7.5.2 SPI_CTRL 636

11.7.5.3 SPI_STATUS 637

11.7.5.4 SPI_TX_DATA 638

11.7.5.5 SPI_RX_DATA 638

11.7.5.6 SPI_RX_DATA_NO_START 638

11.7.5.7 SPI_RX_DATA_MIRROR 639

11.7.5.8 SPI_ID_NUM 639

11.7.6 UART Registers 639

11.7.6.1 UART_CFG 639

11.7.6.2 UART_CTRL 641

11.7.6.3 UART_STATUS 641

11.7.6.4 UART_TX_DATA ... 642
11.7.6.5 UART_RX_DATA ... 642
11.7.6.6 UART_ID_NUM ... 642
12. Sensor Interfaces ... 643
12.1 Analog Comparator ... 645
12.1.1 Functional Description ... 645
12.1.2 Block Diagram ... 645
12.2 SAR-ADC ... 646
12.2.1 Power Source ... 646
12.2.2 Functional Description ... 646
12.2.3 Block Diagram ... 646
12.2.4 ADC Calibration ... 648
12.2.5 ADC Outputs ... 648
12.2.6 Timing Diagrams ... 649
12.3 Pulse Counter ... 650
12.4 LSAD ... 651
12.4.1 LSAD Input Configuration ... 651
12.4.2 LSAD Sampling Configuration ... 652
12.4.3 LSAD Output Data ... 654
12.4.4 Voltage Monitoring ... 654
12.4.5 LSAD and Voltage Monitoring Interrupt ... 655
12.4.6 Using LSAD calibration data ... 655
12.5 Simple Low-Power DAC ... 655
12.5.1 Functional Description ... 655
12.6 Current Source ... 656
12.6.1 Description ... 656

12.6.2 Current Source Configuration ... 657

12.6.3 Temperature Calculation ... 658

12.7 Internal Temperature Sensor ... 658

12.7.1 Enabling the Temperature Sensor ... 659

12.7.2 Calibration Procedures ... 659

12.7.2.1 One-Point Calibration ... 659

12.7.2.2 Two-Point Calibration ... 659

12.8 Ultra-Low Power Data Acquisition Subsystem ... 660

12.8.1 Clock Source Configuration ... 661

12.8.2 Input Configuration ... 661

12.8.3 Sample Accumulation ... 661

12.8.4 Sample Storage ... 662

12.8.5 Signal Threshold Detection ... 663

12.9 Sensor Interfaces Registers ... 663

12.9.1 Analog Comparator Registers ... 664

12.9.1.1 ACS_ACOMP_CFG ... 664

12.9.1.2 ACS_ACOMP_OUT ... 666

12.9.2 SAR-ADC Registers ... 666

12.9.2.1 SENSOR_SAR_CFG ... 666

12.9.2.2 SENSOR_SAR_CTRL ... 668

12.9.2.3 SENSOR_SAR_DATA ... 669

12.9.3 Pulse Counter Registers ... 669

12.9.3.1 SENSOR_PC_CFG ... 669

12.9.3.2 SENSOR_PC_COUNT ... 670

12.9.4 LSAD Registers ... 670

12.9.4.1 LSAD_DATA_TRIM_CH ... 671

12.9.4.2 LSAD_DATA_AUDIO_CH 671

12.9.4.3 LSAD_INPUT_SEL 671

12.9.4.4 LSAD_CFG 672

12.9.4.5 LSAD_OFFSET 673

12.9.4.6 LSAD_INT_ENABLE 673

12.9.4.7 LSAD_MONITOR_CFG 674

12.9.4.8 LSAD_MONITOR_COUNT_VAL 675

12.9.4.9 LSAD_MONITOR_STATUS 675

12.9.4.10 LSAD_PRE_SEL_INPUT 676

12.9.4.11 LSAD_DUTY 677

12.9.4.12 LSAD_ID_NUM 679

12.9.5 SDAC Registers 679

12.9.5.1 ACS_SDAC_CFG 680

12.9.6 Current Source Registers 680

12.9.6.1 ACS_TEMP_CURR_CFG 680

12.9.7 Internal Temperature Sensor Registers 681

12.9.7.1 ACS_TEMP_SENSOR_CFG 681

12.9.8 ULP Registers 682

12.9.8.1 SENSOR_CFG 682

12.9.8.2 SENSOR_CTRL 683

12.9.8.3 SENSOR_FIFO_CFG 684

12.9.8.4 SENSOR_PROCESSING 685

12.9.8.5 SENSOR_FIFO_DATA 685

13. Peripherals 686

13.1 Activity Counters 686

13.2 Asynchronous Clock Counter 686

13.3 Cyclic Redundancy Check (CRC) Generator 688

13.4 Direct Memory Access (DMA) Controller 690

13.4.1 Block Overview 690

13.4.2 Functional Description 691

13.4.2.1 DMA to Peripherals Interface 691

13.4.2.2 Configuration and Status 693

13.4.2.3 DMA Interrupt Configuration 697

13.4.2.4 Circular Mode 698

13.4.2.5 Word Size 698

13.4.2.6 Packing and Unpacking 699

13.4.2.7 DMA Operation 703

13.4.2.8 DMA Channel Arbiter 704

13.5 Flash Copier 705

13.6 Time of Flight 706

13.6.1 BLE Link Filtering 708

13.7 Timers 708

13.7.1 Starting or Stopping Timers 709

13.7.2 Mode Selection 709

13.7.3 GPIO Interrupt Capture Mode 709

13.8 Watchdog Timer 710

13.9 Peripherals Registers 711

13.9.1 Activity Counters Registers 711

13.9.1.1 SYSCTRL_CNT_CTRL 711

13.9.1.2 SYSCTRL_SYSCLK_CNT 712

13.9.1.3 SYSCTRL_CM33_CNT 712

13.9.1.4 SYSCTRL_CBUS_CNT 712

13.9.1.5 SYSCTRL_FLASH_READ_CNT 712

13.9.2 Asynchronous Clock Counter Registers 712

13.9.2.1 ASCC_CTRL 712

13.9.2.2 ASCC_CFG 714

13.9.2.3 ASCC_CNT 714

13.9.2.4 ASCC_PHASE_CNT 715

13.9.2.5 ASCC_PERIOD_CNT 715

13.9.2.6 ASCC_ID_NUM 715

13.9.3 CRC Registers 715

13.9.3.1 CRC_CFG 716

13.9.3.2 CRC_VALUE 716

13.9.3.3 CRC_ADD_1 717

13.9.3.4 CRC_ADD_8 717

13.9.3.5 CRC_ADD_16 717

13.9.3.6 CRC_ADD_24 717

13.9.3.7 CRC_ADD_32 717

13.9.3.8 CRC_FINAL 717

13.9.3.9 CRC_ID_NUM 717

13.9.4 DMA Registers 718

13.9.4.1 DMA_CFG0 719

13.9.4.2 DMA_CFG1 725

13.9.4.3 DMA_CTRL 725

13.9.4.4 DMA_STATUS 728

13.9.4.5 DMA_SRC_ADDR 729

13.9.4.6 DMA_DEST_ADDR 729

13.9.4.7 DMA_CNTS 729

13.9.4.8 DMA_SRC_BUFFER 729

13.9.4.9 DMA_ID_NUM 730

13.9.5 Flash Copier Registers 730

13.9.5.1 FLASH_COPY_CFG 730

13.9.5.2 FLASH_COPY_CTRL 731

13.9.5.3 FLASH_COPY_SRC_ADDR_PTR 732

13.9.5.4 FLASH_COPY_DST_ADDR_PTR 732

13.9.5.5 FLASH_COPY_WORD_CNT 732

13.9.6 Time of Flight Registers 732

13.9.6.1 TOF_CFG 732

13.9.6.2 TOF_CTRL 735

13.9.6.3 TOF_STATUS 735

13.9.6.4 TOF_LINK_CFG 736

13.9.6.5 TOF_DATA 737

13.9.6.6 TOF_MIN_DATA 737

13.9.6.7 TOF_MAX_DATA 737

13.9.6.8 TOF_AVG_DATA 737

13.9.6.9 TOF_ID_NUM 737

13.9.7 Timers Registers 738

13.9.7.1 TIMER_CFG0 738

13.9.7.2 TIMER_CFG1 739

13.9.7.3 TIMER_CTRL 740

13.9.7.4 TIMER_VAL 741

13.9.7.5 TIMER_VAL_CAPTURE 741

13.9.7.6 TIMER_ID_NUM 741

13.9.8 Watchdog Timer Registers 741

13.9.8.1 WATCHDOG_CFG 742

13.9.8.2 WATCHDOG_CTRL 742

13.9.8.3 WATCHDOG_ID_NUM 742

A. Registers 744

A.1 System Control 745

A.2 Clock Generation 763

A.3 Reset 765

A.4 Watchdog Timer 766

A.5 General-Purpose Timers 767

A.6 Flash Interface 772

A.7 GPIO Interface and Digital Pad control 781

A.8 DMA Controller 786

A.9 LIN Interface 797

A.10 LSAD 801

A.11 Sensor Interface 804

A.12 SPI Interface 807

A.13 PCM Interface 813

A.14 I2C Interface 815

A.15 TEST Control Interface 822

A.16 UART Interface 823

A.17 PWM Interface 825

A.18 CRC Generator Control 826

A.19 ASCC 827

A.20 ACS 829

A.21 AHB registers 842

A.22 Baseband Controller Interface 843

A.23 Baseband Controller 845

A.24 RF Front-End 2.4 GHz 872

A.25 Implementation Control Block 1029

A.26 SysTick Timer 1030

A.27 Time Of Flight Timer 1031

A.28 System Control and ID register not in the SCB 1034

A.29 Nested Vector Interrupt Controller 1035

A.30 System Control Block 1056

A.31 Memory Protection Unit 1062

A.32 Security Attribution Unit 1064

A.33 Debug Control Block 1065

A.34 Software Interrupt Generation 1066

A.35 Floating-Point Extension 1066

CHAPTER 1

Introduction

1.1 Purpose

IMPORTANT: onsemi plans to lead in replacing the terms “white list", "master" and "slave” as noted in this product release. We have a plan to work with other companies to identify an industry wide solution that can eradicate non-inclusive terminology but maintains the technical relationship of the original wording. Once new terminologies are agreed upon, we will update all documentation live on the website and in all future released documents.

This group of topics provides a reference to the system hardware for application developers working with RSL15. These topics describe all of the components belonging to RSL15, including:

- Bluetooth® and generic RF support
- Data processing and control components
- Security elements
- Clocking components and power supply
- Memory components
- Interfaces
- Peripherals

The data processing and control component information provided by this reference complements the Arm® Cortex®-M33 core description in *The Definitive Guide to the Arm Cortex-M23 and Cortex-M33 Processors* and other third-party documentation for the Arm Cortex-M33 processor.

The *RSL15 Hardware Reference* describes how each component of the RSL15 SoC can be used in the implementation of an application, and provides information about the configuration of the various system components. This group of topics is a part of the RSL15 Software Development Kit (RSL15 SDK).

1.2 Intended Audience

This reference 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 RSL15. People who are developing local interfaces between an external device and RSL15, as well as those interested in the details of the audio, power supply, and clocking components, will find this group of topics particularly helpful, as it focuses on the configuration and use of system components.

This reference assumes that readers are familiar with C-level programming, RF application concepts, and Bluetooth Low Energy technology.

1.3 Document Conventions

The following typographical conventions are used in this documentation:

`monospace font`
: Assembly code, macros, functions, registers, defines and addresses.

italics
: File and path names, or any portion of them.

<angle brackets and bold>

Optional parameters and placeholders for specific information. To use an optional parameter or replace a placeholder, specify the information within the brackets; do not include the brackets themselves.

Bold

GUI items (text that can be seen on a screen).

Note, Important, Caution, Warning

Information requiring special notice is presented in several attention-grabbing formats depending on the consequences of ignoring the information:

NOTE: Significant supplemental information, hints, or tips.

IMPORTANT: Information that is more significant than a Note; intended to help you avoid frustration.

CAUTION: Information that can prevent you from damaging equipment or software.

WARNING: Information that can prevent harm to humans.

Registers:

Registers are shown in `monospace font` using their full descriptors, depending on which core the register is accessing. The full description takes the form **`<PREFIX><GROUP>_<REGISTER>`**.

All registers are accessible from the Arm Cortex-M33 processor.

A register prefix of `D_` is used in the following circumstances:

- In cases where there are multiple instances of a block of registers, the summary of the registers at the beginning of the Register section have slightly different names from the detailed register sections below that table. For example, the `DMA*_CFG0` registers are referred to as `DMA_CFG0` when we are defining bit-fields and settings.

The firmware provides access to these registers in two ways:

- In the flat header files (e.g.: *sk5_hw_flat_cid*.h*), each register is individually accessible by directly using the naming provided in this manual. This is helpful for assembly and low-level C programming.
- In the normal header files (e.g.: *sk5_hw_cid*.h*), each register group forms a structure, with the registers being defined as members within that structure. The structures defined by these header files provide access to registers under the naming conventions `PREFIX_GROUP->REGISTER` (for the structure) and `GROUP->REGISTER` (for the register).
- For more information, see the Hardware Definitions chapter of the *RSL15 Firmware Reference*.

Default settings for registers and bit fields are marked with an asterisk (*).

Any undefined bits must be written to 0, if they are written at all.

Numbers

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.

Sample Rates

All sample rates specified are the final decimated sample rates, unless stated otherwise.

1.4 Further Reading

The following documents are installed with the RSL15 system, in the default location *C:/Users/*`<your_user_name>`*/AppData/Local/Arm/Packs/ONSemiconductor/RSL15/*`<version_number>`*/documentation*. These manuals are available only in PDF format:

- *Arm TrustZone CryptoCell-312 Software Developers Manual*
- multiple CEVA manuals in the */ceva* folder

For even more information, consult these publicly-available documents:

- *Armv8M Architecture Reference Manual* (PDF download available from https://developer.arm.com/documentation/ddi0553/latest).
- *Arm Cortex-M33 Processor Technical Reference Manual*, revision r1p0, from https://developer.arm.com/documentation/100230/0100
- *Bluetooth Core Specification version 5.2*, available from https://www.bluetooth.com/specifications/adopted-specifications
- TrustZone documentation available from the Arm website at https://developer.arm.com/ip-products/security-ip/trustzone/trustzone-for-cortex-m
- Other ArmCortex-M33 publications, available from the Arm website at https://developer.arm.com/ip-products/processors/cortex-m/cortex-m33

For information about the Evaluation and Development Board Manual and its schematics, go to the RSL15 web page and navigate to the EVB page.

CHAPTER 2

System Overview

RSL15 is a highly integrated secure Arm Cortex-M33-based Bluetooth Low Energy 5.2 wireless MCU system-on-a-chip, with flash and RAM, built-in power management, and an extensive set of peripherals. The wide supply voltage input, flexible I/O, and clocking scheme offer maximum design flexibility.

In RSL15, the lowest levels of common hardware usage are supported by a hardware abstraction layer (HAL). This HAL can be employed to simplify the use of the hardware, and to render more readable code that uses low-level hardware operations.

This system overview includes cross-references to relevant sections in the HAL topic of the *RSL15 Firmware Reference*. These links indicate where to look for low-level firmware that can be used to abstract unnecessary hardware details away from your code.

2.1 System Architecture

The "Detailed Block Diagram of the RSL15 System" figure (Figure 1) shows a detailed block diagram of the RSL15 system.

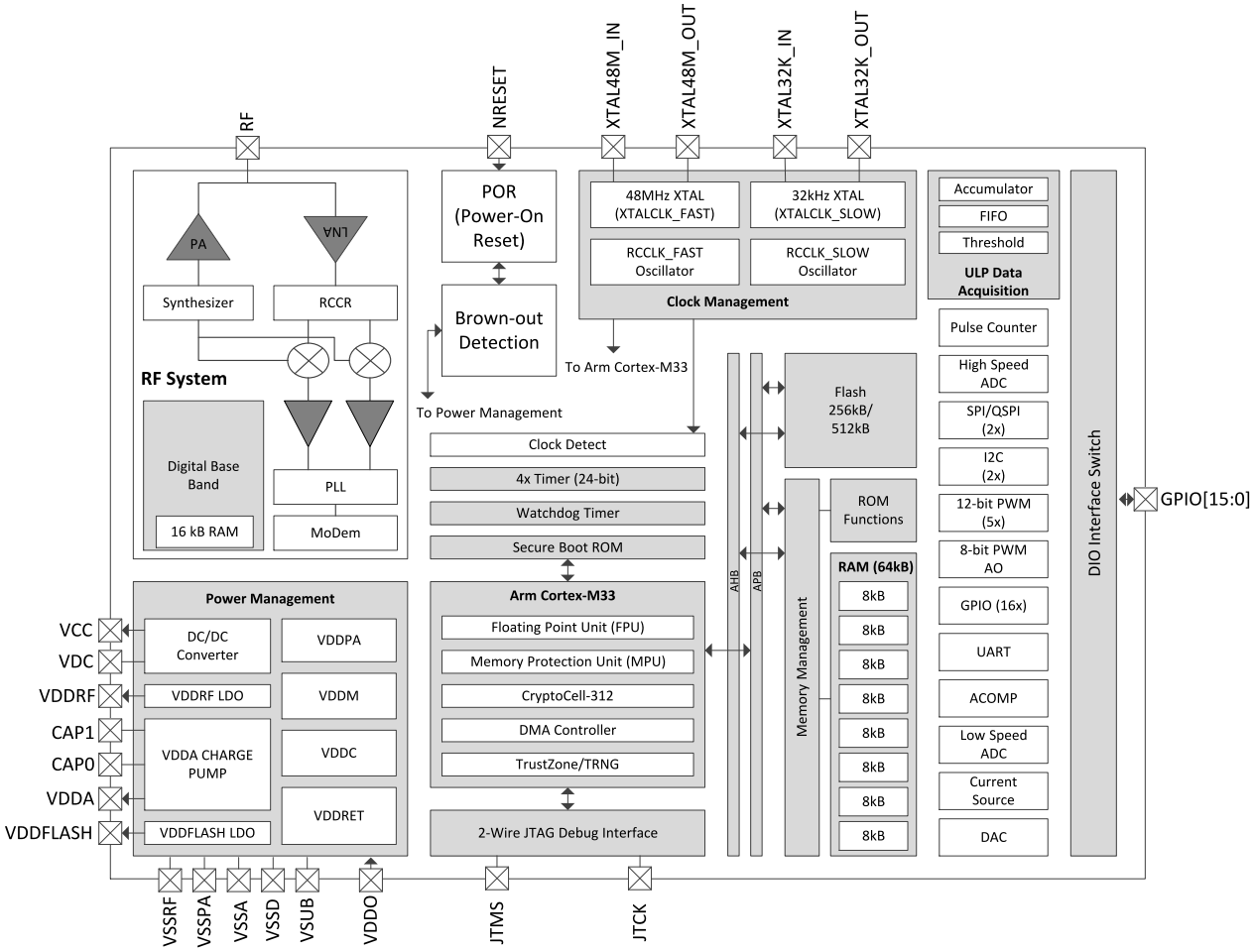


Figure 1. Detailed Block Diagram of the RSL15 System

RSL15 is based around four key components:

1. **The Arm Cortex-M33 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 is tightly integrated with the TrustZone Armv8-M security extensions, and is used to provide secure general-purpose processing and control of the RSL15 system's components, including the RF front-end and Bluetooth baseband.
2. **TrustZone® Arm CryptoCell™-312:** Provides a comprehensive security solution, with features protecting the authenticity, integrity and confidentiality of code and data, along with cryptographic hardware services, TRNG, and lifecycle state management.
3. **Radio Frequency Front-End:** Based on a 2.4 GHz RF transceiver, the radio frequency front-end (RFFE) implements the physical layer of the Bluetooth Low Energy technology standard, and can be used to implement the RF layer of other standard and custom protocols.
4. **Bluetooth Protocol Baseband Hardware:** Bluetooth low energy technology compliant, including support for all mandatory and optional features of the Bluetooth core 5.2 technology release, and compliance with the Blutetooth 5.2 core specification.

2.2 Security System Architecture

2.2.1 Integrated IoT Cybersecurity Platform

The Arm Cortex-M33 processor with TrustZone Armv8-M security extensions forms the basis of the security platform. The Arm CryptoCell-312 allows for end-to-end product security, with Secure Boot with Root of Trust, secure lifecycle management, secure key management, and application and data encryption using symmetric or asymmetric cryptography. Arm TrustZone enables secure software access control.

For more information about the Arm Cortex-M33 processor, see Chapter 3.1 "Introduction" on page 55. For more information about the TrustZone CryptoCell-312 security IP integrated into RSL15, see Chapter 4 "Arm TrustZone CryptoCell-312 Security IP" on page 117.

2.2.2 Secure Boot with Root of Trust

The secure boot ROM authenticates firmware in flash with a certificate-based mechanism using a private-public key scheme. This is the basis of the hardware Root of Trust. This same mechanism ensures continuity of the hardware Root of Trust after secure Firmware-Over-The-Air (FOTA) update.

2.2.3 Arm Lifecycle State Management

The term *lifecycle states* refers to the multiple states RSL15 goes through during its lifetime. The Arm lifecycle states are:

1. Chip Manufacture (CM) Lifecycle State
2. Device Manufacture (DM) Lifecycle State - used during device manufacturing
3. Secure (SE) Lifecycle State - used during field deployment
4. Return to Manufacturer (RMA) Lifecycle State - used during failure analysis of devices that are returned to the manufacturer.

Lifecycle state management ensures the authenticity, integrity and confidentiality of code and data belonging to different stakeholders at each lifecycle.

In addition to the Arm Lifecycle States, an Energy Harvesting (EH) Mode is available for applications that require fast cold startup (initial application of VBAT) — for example, when cold startup is applied on reset — but do not require Root of Trust. This mode is especially useful when RSL15 is used in energy harvesting systems.

2.2.4 Application and Data Cryptographic Services

User available cryptographic services allow for development of custom proprietary security solutions. Cryptographic services include:

- Encryption and decryption schemes
- Hash schemes
- Message Authentication Coding
- Key generation and exchange algorithms

These cryptographic services are supported by a True Random Number Generator (TRNG).

2.2.5 TrustZone Arm Cortex-M33 Peripherals

The secure attribution unit (SAU) and its associated secure faults are integrated with the Arm Cortex-M33 processor and its NVIC. These blocks enable secure software access control to protect critical software and hardware resources, and support secure execution of both secured and unsecured application elements in the same device. The

SAU is completed by the implementation defined attribute unit (IDAU), a hardware unit external to the processor.

2.3 Radio System Architecture

The RSL15 system is built around a radio system architecture that supports the implementation of Bluetooth and other proprietary RF protocol stacks.

The most common use case for RSL15 devices is in applications that use Bluetooth technology.

The 2.4 GHz radio front-end is based on a low-IF architecture, and is comprised of the following building blocks:

- High performance single-ended RF port that alleviates the need for an external balun
- On-chip matching network with 50 Q RF input
- Low power LNA (low noise amplifier) and mixer
- PA (Power Amplifier) with up to +6 dBm output power for Bluetooth
- RSSI (Received Signal Strength Indication) with 60 dB nominal range in 1 dB steps (not considering AGC)
- Fully integrated ultra-low power frequency synthesis with fast settling time, featuring direct digital modulation in transmission (pulse shape programmable)
- Configurable direct access to streamed radio data, including ADC and RSSI data, or I/Q samples for signal phase measurements
- 48 MHz XTAL oscillator
- Fully-integrated FSK-based modem with programmable pulse shape, data rate, and modulation index
- Digital baseband (DBB) with link layer functionalities, including automatic packet handling with preamble and sync, CRC, and separate RX and TX 128-byte FIFOs
- 2.4 GHz radio front-end, containing a full transceiver with the following features:
    - IEEE 802.15.4 chip encoding and decoding
    - Manchester encoding
    - Data whitening
- Highly-flexible digital baseband — in terms of modulation schemes, configurability and programmability — supporting Bluetooth Low Energy technology, 802.15.4 OQPSK and DSSS, and proprietary protocols. This baseband allows for programmable data rates from 62.5 kbps up to 2 Mbps, and FSK with programmable pulse shape and modulation index.
- A simple baseband for direct packet handling for the 2.4 GHz radio front-end, which includes:
    - Automatic preamble and sync word insertion
    - Automatic packet length handler
    - Basic address check
    - Automatic CRC calculation and verification with a programmable CRC polynomial
    - Multi-frame support
- Time-of-Flight (ToF) — a proprietary mechanism that counts the number of system clock cycles between two configurable start and stop triggers

2.3.1 RF Subsystem

The RSL15 2.4 GHz radio front-end implements the physical layer for the Bluetooth Low Energy standard, and for other standard, proprietary, and custom protocols. The modem is of the FSK type with a single-ended RF Port, which alleviates the need for an external balun.

The radio front-end operates in the worldwide deployable 2.4 GHz ISM band (2.4000 to 2.4835 GHz).

For more information about the RF front-end, see Chapter 5 "RF Front-End" on page 140.

For HAL firmware information related to the RF front-end, see the *RSL15 Firmware Reference*: "RFFE Radio Frequency Front End" on page 1.

2.3.2 Bluetooth Low Energy

RSL15 is Bluetooth 5.2 certified, with the following Bluetooth Low Energy features:

- Angle of Arrival (AoA) and Angle of Departure (AoD)
- Low Energy Long Range (LE Coded PHY)
- 2 Msym/s symbol rates (LE 2M PHY)
- Low Energy Advertising extensions
- High Duty Cycle Non-Connectable Advertising
- Low Energy Channel Selection Algorithm #2
- Advertising Channel Index
- GATT Caching
- HCI support for debug keys in Low Energy Secure Connections
- Sleep clock accuracy update mechanism
- ADI field in scan response data
- Host channel classification for secondary advertising
- Periodic Advertising Sync Transfer
- Backward compatibility and support for earlier Bluetooth Low Energy specifications including 5.1, 5.0, and 4.2

  NOTE: 4.0 and 4.1 have been deprecated by the Bluetooth SIG.

The hardware enables implementation of custom protocols.

For more information about the Bluetooth baseband hardware, see Chapter 6 "Bluetooth Low Energy Baseband Controller" on page 301. For more information about the Bluetooth stack, see the *RSL15 Firmware Reference*, and related 3rd party documentation provided from Riviera Waves for this product.

For HAL firmware information related to the Bluetooth baseband interface, see the *RSL15 Firmware Reference*: "Baseband Interface" on page 1.

2.4 POWER

The flexible power management of RSL15 allows for a wide range of battery voltages without the need for external power conversion. Two modes for the system supply are:

1. LDO Mode Operation (the default)
2. DC-DC Mode Operation

In both modes the battery supply voltage (VBAT) is regulated to a system supply (VCC) of 1.2 V. This supply is converted (using a charge pump) to an approximate 2.4 V analog supply voltage (VDDA), which is used to power the analog blocks (excluding the RF Blocks). These supplies are used as inputs to several additional supply voltages:

- VDDFLASH is a regulated voltage used to power the flash for erase and program operations.
- VDDRF is a regulated voltage used to supply the RF system.
- VDDC is a regulated voltage used for the internal digital blocks, excluding digital memories (ROM, RAM, flash) and GPIOs.
- VDDM is a regulated voltage used for the memory (ROM, RAM, flash) blocks.
- VDDPA is a regulated voltage used to supply the RF power amplifier (used in RF TX mode). The VDDPA level depends on the output power level selected.

All power supplies are trimmed by onsemi as part of the device manufacturing process.

VDDO is a power supply input to RSL15 and constitutes the logical high level for the digital I/Os, i.e., if VDDO is connected to VBAT, the GPIO signal swing is between GND and VBAT.

For more information about power supplies, see Chapter 8.2 "Power Supply Overview" on page 406.

For HAL firmware information related to power supply, see the *RSL15 Firmware Reference*: "POWER" on page 1.

2.4.1 Power Management Unit (PMU) Battery Monitoring and Resets

The RSL15 Power Management Unit (PMU) prevents system brown-outs in case the battery voltage dips below the specified minimum voltage required for reliable operation.

The PMU does this by:

1. Monitoring the power supply and safely shutting down the system if needed
2. Preventing possible damage to RSL15 when the battery is inserted or removed
3. Allowing operation across wide temperature and voltage ranges at low power consumption

The PMU automatically resets the internal systems during power supply disruptions, such as insufficient battery voltage or battery insertion/removal. Upon power supply rise (such as battery insertion), the system is held in Power-On-Reset until sufficient internal voltages are reached and stabilized. When POR is released, the boot ROM execution begins using the RC clock at 3 MHz.

A reset can also be issued by:

- Assertion of the nRESET pin
- Software triggers
- Watchdog timer expiration
- Invalid or missing clock detected by the clock detector

For more information about power management, see Section 8.1 “Power Management Unit” on page 406. For more information about resets, see Section 8.4 “Resets” on page 427.

2.4.2 Power Modes Overview

The power modes are available to reduce power consumption while maintaining system responsiveness. The low power modes are Sleep Mode and Standby Mode. Smart Sense is a low-power feature that works as a subset of other power modes, and Idle Mode is a subset of Standby Mode.

- Sleep Mode is the lowest power mode, but with the longest wakeup time.
  - Smart Sense Mode takes advantage of the low power capability of Sleep Mode, but also allows some digital and analog peripherals to remain active with minimal processor intervention. Smart Sense Mode allows RSL15 to not only remain responsive to external events, but also to monitor and acquire data from external sensors with very low system-level power consumption.
- Standby Mode is low power, but with a faster wakeup time than Sleep Mode.
  - Idle Mode allows for some power savings with the fastest wakeup time through disabling internal clocks.

Smart Sense, Sleep, and Standby Modes have the ability of RAM retention (a configurable amount of RAM to be retained), and allow for configurable wakeup sources. Wakeup sources include:

- GPIO transition (pin-based wakeup)
- RTC Timer

- Analog comparator
- Threshold trigger or sample FIFO full detections in the ultra-low power data acquisition block

For more information about power modes, see Section 8.6 “Power Modes” on page 431.

For HAL firmware information related to power modes, see the *RSL15 Firmware Reference*: "HAL Power Modes" on page 1.

2.5 CLOCKING

The following oscillators are available as sources for the system clock used for active operation of the system:

- 48 MHz crystal oscillator (`XTALCLK`), typically used in Run Mode when RF operation is required. Prescalers exist to provide divided clocks (including system clock) to other parts of the system.
- A fast RC oscillator (`RCCLK`) can provide an alternative to the 48 MHz crystal oscillator. However, RF operation is not possible using the fast RC Oscillator.

The following oscillators are available as sources for the standby clock used for system timing:

- A 32 kHz crystal oscillator (`XTAL32K`) typically used in Sleep and Standby Modes for precision timing and to maintain the real-time clock (RTC)
- A standby RC oscillator (`RC32`) that meets the Bluetooth Low Energy sleep clock accuracy requirements, which can be an alternative to the 32 kHz crystal oscillator; see "RC Oscillator" on page 383

A built−in clock detector ensures a proper system reset in case the system clock goes below 2 kHz.

Flexible clock management allows the different clock sources to be used in power−efficient ways, and minimizes the use of external components. Internal RC oscillators can be used for fast startup, and then easily be switched out for crystal oscillators when precision timing is required. The `RC32` 32 kHz RC oscillator can meet the Bluetooth Low Energy Sleep Clock accuracy requirements, which can alleviate the need for an external 32 kHz crystal.

The system clock can also be sourced from external sources.

For more information on clock generation and distribution in the RSL15 SoC, see Chapter 7.2 "Clock Generation" on page 383.

For HAL firmware information related to clocking, see the *RSL15 Firmware Reference*: Section 1.1 “Clock Configuration” on page 1.

2.5.1 RTC

The RTC timer consists of a 32-bit free-running up-counter, clocked by the standby clock.

For HAL firmware information related to the RTC, see the *RSL15 Firmware Reference*: "RTC Function Documentation" on page 1.

2.6 MEMORY

All aspects of the RSL15 system, including all memory instances, registers, and other components, are accessible from the Arm Cortex-M33 processor through one or more of the processor’s standard buses. This structure supports the control and configuration of all of the components of the system, and simplifies control of the RF front-end, Bluetooth protocol baseband hardware, and security elements.

The memory space of the RSL15 system is subdivided into four main segments:

1. The program memory used for storing and/or executing code on the Arm Cortex-M33 processor. This segment of the RSL15 memory map contains:
    - A 20-KB ROM instance
    - A 284-KB (RSL15-284) flash instance with 264 KB code and 20 KB data memory
    - A 512-KB (RSL15-512) flash instance with 352 KB code and 160 KB data memory
    - Four 256-byte non-volatile records with restricted use (four additional non-volatile records are reserved for use by the CryptocCell-312 only, and one is restricted for device manufacturing use only).
2. The data memory used for storing data and intermediate variables of the Arm Cortex-M33 processor, and/or the Bluetooth protocol baseband hardware. This segment of the RSL15 memory map contains:
    - Eight 8-KB data RAM instances only accessible by the Arm Cortex-M33 processor
    - Two 8-KB baseband data RAM instances acting as exchange memory between the Bluetooth protocol baseband hardware and the Arm Cortex-M33 processor
3. The peripheral bus accessible memory-mapped registers
4. The private peripheral bus accessible Arm Cortex-M33 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 (for more information on the DMA controller, see Section 13.4 “Direct Memory Access (DMA) Controller” on page 690)
- 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 RSL15-based system, see Chapter 9.1 "Architecture" on page 466.

2.7 INTERFACES

The RSL15 system has a number of interfaces to support communicating with external devices. These interfaces and pads are described briefly in the sections below, and in detail in Chapter 10.1 "Overview" on page 511, Chapter 11 "Communication Interfaces" on page 575, and Chapter 12 "Sensor Interfaces" on page 643.

2.7.1 General Purpose Input/Output (GPIO)

RSL15 contains up to 16 highly flexible general purpose input/output (GPIO) pins that can be configured as digital input or output, communication interfaces, clocks, wakeup sources, or analog functions. Communication interfaces can be routed to any GPIO. Other functions are available on select GPIOs (see section Pin Definition and Multiplexing). Each GPIO has a software configurable pull up/down resistor, LPF for I^2C, and four drive strengths options.

For HAL firmware information related to GPIO, see the *RSL15 Firmware Reference*: "General-Purpose I/O Interface" on page 1.

2.7.2 Communications Interfaces

2.7.2.1 I^2C

The I^2C controller consists of two independent channels of the two-wire interface, including a bidirectional clock line (SCL) and bidirectional data line (SDA). The I^2C interface supports both master and slave mode operation. Both 100 kHz, 400 kHz, and 1 MHz modes are supported.

For HAL firmware information related to the I^2C interface, see the *RSL15 Firmware Reference*: "I2C" on page 1.

2.7.2.2 LIN

The Local Interconnect Network (LIN) controller is designed to be paired with a LIN transceiver, such as the onsemi NCV7420, to provide a LIN connection that is compliant to specification revision 2.2 (backward compatible to version 1.3). The LIN controller supports one asynchronous 2-wire interface including both receive and transmit communications.

For HAL firmware information related to the LIN interface, see the *RSL15 Firmware Reference*: "LIN" on page 1.

2.7.2.3 PCM

The highly configurable PCM (Pulse Code Modulation) interface can be used to stream data into and out of RSL15.

2.7.2.4 PWM

The RSL15 SoC contains two pulse width modulation (PWM) interfaces:

- The primary PWM controller can output on five independent channels with configurable period, duty cycle, and offset. The PWM has 12-bit resolution with an optional 8-bit dithering per channel for lighting applications.
- The analog control system's PWM controller operates on one channel, has configurable period and duty cycle, and can be kept enabled in any power mode.

For HAL firmware information related to the PWM interface, see the *RSL15 Firmware Reference*: "Pulse-Width Modulation" on page 1.

2.7.2.5 SPI

The SPI controller consists of two independent channels, with the standard 4-wire interface of SCLK, MOSI, MISO and CS supporting master and slave mode. Each channel also supports dual (DSPI) and quad (QSPI) modes in half or full duplex mode.

For HAL firmware information related to the SPI interface, see the *RSL15 Firmware Reference*: "SPI" on page 1.

2.7.2.6 UART

The general-purpose universal asynchronous receiver-transmitter (UART) uses a standard data format with one start bit, eight data bits, and one stop bit.

For HAL firmware information related to the UART interface, see the *RSL15 Firmware Reference*: "UART" on page 1.

2.7.3 Sensor Support Interfaces

2.7.3.1 Analog Comparator

RSL15 contains a low-power comparator that can be active in Standby, Sleep, and Smart Sense Modes. It has three different settings to trade off response time with power consumption: Low Power, Normal, and High Speed. See Section 12.1 “Analog Comparator” on page 645.

2.7.3.2 SAR ADC

The successive approximation (SAR) ADC generates 12−bit samples up to 2 Msps sample frequency. This ADC is auto calibrated during operation for optimal INL/DNL performance.

2.7.3.3 Pulse Counter

A pulse counter can be driven by one of GPIO[3:0]. It counts pulses from the selected GPIO during a set interval.

2.7.3.4 Low Speed ADC

The low speed ADC (LSAD) is a combined integrating and algorithmic ADC that has a resolution varying from 8 to 14 bits depending on configuration. While converting, the input signal can be integrated across one or more clock cycles (depending on configuration). The low speed ADC sampling rate can be up to 50 ksps. In addition, the ADC can be configured to measure single ended or differential input voltages, and supports monitoring the VBAT input voltage.

2.7.3.5 Simple DAC

RSL15 contains a simple low-power DAC that can be used for sensor biasing purposes. To optimize power consumption, there is also a buffer that can be disabled if the load is high impedance.

2.7.3.6 Current Source

A built-in current source features adjustable output from 1 µA to 16 µA. The current source may be applied for temperature measurements using an external thermistor connected to a GPIO.

2.7.3.7 ULP Data Acquisition Subsystem

The ULP data acquisition subsystem comprises a small FIFO, accumulator, and threshold comparator that can be used in combination with the high-speed ADC, and a pulse counter to perform data acquisition and rudimentary data processing and decision making. This is available in all power modes, as shown in the "ULP Data Acquisition Subsystem" figure (Figure 2).

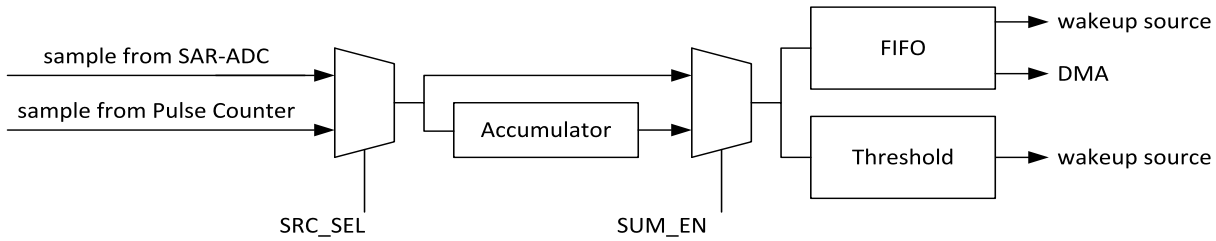


Figure 2. ULP Data Acquisition Subsystem

2.8 Peripherals

The RSL15 system has a number of peripherals to support auxiliary non-RF tasks. These peripherals are described briefly in the sections below, and in detail in Chapter 13.1 "Activity Counters" on page 686.

Additional peripherals that are only accessible to the Arm Cortex-M33 processor have been defined by Arm and integrated into the RSL15 system. These blocks are described in Section 3.3 “Private Peripherals” on page 57.

2.8.1 Activity Counter

The activity counters help to analyze how long the RSL15 system has been running, and how much the CPU and the flash have been used by the application in a period of time. This is useful information for estimating and optimizing the power consumption of the application.

For HAL firmware information related to the activity counters, see the *RSL15 Firmware Reference*: "Activity Counter" on page 1.

2.8.2 Asynchronous Clock Counter

The asynchronous clock counter (ASCC) measures the timing of a clock signal, such as STANDBYCLK or a clock provided on a GPIO input, relative to the system clock.

For HAL firmware information related to the ASCC, see the *RSL15 Firmware Reference*: "Asynchronous Clock Counter" on page 1.

2.8.3 CRC

This block provides an implementation of two standard cyclic redundancy code (CRC) algorithms (CRC-CCITT and CRC-32 - IEEE 802.3), which can be used to ensure the integrity of a user application's code and data.

For HAL firmware information related to the CRC, see the *RSL15 Firmware Reference*: "Cyclic Redundancy Check" on page 1.

2.8.4 DMA Controller

The Direct Memory Access (DMA) Controller allows background transfers between peripherals and memories without processor intervention. The system can be in the idle low power state, with the processor waiting for an interrupt or event, or in use for other computational tasks while the transfer occurs. The DMA is connected to the processor, peripherals, and RAM memories, and has four independent channels.

For HAL firmware information related to the DMA, see the *RSL15 Firmware Reference*: "Direct Memory Access" on page 1.

2.8.5 Flash Copier

The flash copier supports several operations on data that has been stored to flash memory:

- A background copy of data stored in the flash to another area in memory
- A background copy of flash data to the CRC block for data integrity validation
- A comparison operation over a block with a fixed value, used primarily to validate if data has been erased or has been cleared by being overwritten to zero

For HAL firmware information related to the flash copier, see the *RSL15 Firmware Reference*: "Flash Copier" on page 1

2.8.6 Time of Flight

RSL15 features a time of flight module, which enables measurement of how long certain operations take to complete, particularly RF operations.

The time of flight module can be stopped and started through software control, or via hardware control using RF front-end interrupts. Features of the module include averaging, maximum and minimum data point recording, sample counts, and automatic data transfer via DMA.

For more information about the time of flight module and its use, see "Time of Flight" on page 706.

For HAL firmware information related to time of flight, see the *RSL15 Firmware Reference*: "Time of Flight" on page 1.

2.8.7 Timers

There are four independent 24-bit timers that can operate as single-shot, multi-shot, or free-run. Interrupt or GPIO output can be configured with timer value capture on timer expiration.

For HAL firmware information related to timers, see the *RSL15 Firmware Reference*: "General-Purpose Timer" on page 1.

2.8.8 Watchdog

The watchdog timer must be reloaded at regular intervals. If it is not refreshed and the timer expires, a reset is issued to the system. The watchdog timer is disabled when the RSL15 system is connected to an external debugger, but otherwise cannot be disabled.

For HAL firmware information related to the watchdog timer, see the *RSL15 Firmware Reference*: "WATCHDOG" on page 1.

2.9 Firmware Reference Libraries

The RSL15 SoC hardware described in this manual is supported by a set of firmware header files, libraries, and other components that are described in the *RSL15 Firmware Reference*. In particular, a hardware abstraction layer (HAL) library has been provided, to assist with the following RSL15 development factors:

- Simplification of common interactions between the hardware and user-developed firmware
- Implementation details that require grouping or specific ordering of several operations
- Readability of user applications

2.10 SoC Identification

The RSL15 System on a Chip provides the `AHBREGS_CHIP_ID_NUM` register, which contains static values that identify the chip and provide version information about it. This register can be used to verify compatibility for both user applications and external applications that communicate through the debug port. A list of these the fields provided in this register and a brief description of each are found in the "Device Identification Registers" table (Table 1).

Table 1. Device Identification Registers

Register	Description
`CHIP_FAMILY`	The chip technology family to which the chip belongs. RSL15 belongs to the chip family 11 (0xB).
`CHIP_VERSION`	Version number for the chip. Used to indicate major updates which might not be backward compatible. RSL15 uses a chip version of 0x02.
`CHIP_MAJOR_REVISION`	Revision number for the chip. An update of the major revision number (bits [15:8]) 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.
`CHIP_MINOR_REVISION`	Revision number for the chip. An update of the minor revision number (bits [7:0]) indicates minor backward-compatible or non-functional updates to the system that require no update to source code or binary objects.

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. Chips are also identified using only the chip version number and major revision number as a chip identifier (CID) value. For a chip version of X.YY.ZZ, the CID would be XYY (for example, 102).

Field Name	Value Symbol	Value Description	Hex Value
CHIP_FAMILY	CHIP_FAMILY_NB	BLE family	0xB*
CHIP_VERSION	CHIP_VERSION_NB	Value for RSL15	0x2*
CHIP_MAJOR_REVISION	CHIP_MAJOR_REVISION_NB	Value for RSL15 silicon version 2.0	0x2*
CHIP_MINOR_REVISION	CHIP_MINOR_REVISION_NB	Value for RSL15 silicon version 2.0	0x0*

2.11 SoC Identification Registers

Register Name	Register Description	Address
AHBREGS_CHIP_ID_NUM	Chip ID number	0x1FFFFFFC

2.11.0.1 AHBREGS_CHIP_ID_NUM

Bit Field	Read/Write	Field Name	Description
31:24	R	CHIP_FAMILY	Chip Family number
23:16	R	CHIP_VERSION	Chip Version number
15:8	R	CHIP_MAJOR_REVISION	Chip Major Revision number
7:0	R	CHIP_MINOR_REVISION	Chip Minor Revision number

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:24	CHIP_FAMILY	CHIP_FAMILY_NB	BLE family	0xB*
23:16	CHIP_VERSION	CHIP_VERSION_NB	Value for Nechako	0x2*
15:8	CHIP_MAJOR_REVISION	CHIP_MAJOR_REVISION_NB	Value for Nechako SV2	0x2*
7:0	CHIP_MINOR_REVISION	CHIP_MINOR_REVISION_NB	Value for Nechako SV2	0x0*

CHAPTER 3

Arm Cortex-M33 Processor

The Arm Cortex-M33 processor plays a role as the central controller for the RSL15 microcontroller system. The Arm Cortex-M33 processor has access to all memories and memory-mapped components, and as a result provides users with an interface for configuring and controlling all of the other system components and handling all data transferred through the RSL15 system.

3.1 Introduction

The Arm Cortex-M33 processor implements the Armv8-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
- Includes floating point, DSP extensions
- Includes security extension, and secure protected memory regions (MPU) including the Secure Attribution Unit (SAU)
- 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

Following a power-on reset (POR), the system starts executing the Boot ROM on the Arm Cortex-M33 processor, and uses this ROM application to initialize the system, and validate a system application is both likely to be valid and is secured by the Root of Trust. For more information about the Boot ROM code, see the *RSL15 Firmware Reference*. For more information on the security system, see Chapter 4 "Arm TrustZone CryptoCell-312 Security IP" on page 117.

To reduce the current consumption from fetching instructions, the Arm Cortex-M33 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_CM33_LOOP_CACHE_CFG` register.

3.2 Debug Port

All applications executing on the Arm Cortex-M33 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-M33 processor registers, and to the memory available through the memory buses attached to the Arm Cortex-

M33 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.

IMPORTANT: When operating in energy harvesting mode, full access to the system is available through the debug port. In other life cycles states, access to the device is restricted and controlled by debug certificates. For more information, see Chapter 4 "Arm TrustZone CryptoCell-312 Security IP" on page 117.

IMPORTANT: When a debug port connection has been established, the debug port link remains active in low power modes if the Arm Cortex-M33-Debug power domain is not explicitly disabled. This supports debugging the functionality of an application that uses low power modes across a low power mode-wakeup cycle. For more information, see Section 8.6.1 “Keeping the Debugger Connected in Low Power Modes” on page 435.

The SWJ-DP is assigned to two dedicated pads (JTCK, JTMS), which can be used as the debug port in serial wire (SW) mode. These two pads are augmented for JTAG mode using GPIOs 2 and 3 when the interface is used as a 4-wire JTAG interface, and optionally by GPIO 4 if the interface is used as a 5-wire JTAG interface. By default, GPIOs 2 through 4 are assigned to the SWJ-DP and must be deselected if a user wishes to reassign the GPIOs to other functionality. For more information about the functional configuration of GPIOs for this mode, see Section 10.2 “Functional Configuration” on page 512.

When the SWJ-DP is used in serial wire mode, data is transferred using a single clock and single data signal. This interface mode is useful when the number of signals needed to form a debug port connection needs to be limited, but it is only half-duplex and has a relatively high overhead for transferring 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 signals used to form the debug port connection is not limited, and a higher throughput is required, as JTAG mode uses less data overhead than serial wire mode for full-duplex transfers.

To configure the GPIOs to support the SWJ-DP interface in JTAG mode, take the following steps before configuring the interface for JTAG mode:

- The JTAG test-reset signal (JNTRST) can be connected to GPIO 4 by setting the `GPIO_JTAG_SW_PAD_CFG_CM33_JTAG_TRST_EN` bit from the `GPIO_JTAG_SW_PAD_CFG` register; this is only required when using the SWJ-DP as a 5-pin JTAG interface.
- The JTAG data signals can be connected to GPIOs 2 (JTDI) and 3 (JTDO) by setting the `GPIO_JTAG_SW_PAD_CFG_CM33_JTAG_DATA_EN` bit from the `GPIO_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. 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 0xE73C (transmitted LSB first) using SWDIO. This initialization pattern switches the debug port to JTAG mode.
3. Send at least 5 clock cycles on SWCLK 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 clock cycles on SWCLK 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 degree of physical configuration. The output pads support a configurable drive strength, and the input pads support configurable pull-up resistances. Configuration of these interface pads uses the `GPIO_JTAG_SW_PAD_CFG` register. The physical configuration parameters are set as follows:

- JTMS (SWDIO) pad configuration:
  - A pull-up or pull-down resistor can be configured and applied to the pad using the `GPIO_JTAG_SW_PAD_CFG_JTMS_PULL` bit-field in the `GPIO_JTAG_SW_PAD_CFG` register.
  - The drive strength can be configured using the `GPIO_JTAG_SW_PAD_CFG_JTMS_DRIVE` bit-field in the same register.
  - A low-pass filter can be enabled or disabled using the `GPIO_JTAG_SW_PAD_CFG_JTMS_LPF` bit in the same register.
- JTCK (SWCLK) pad configuration:
  - A pull-up or pull-down resistor can be configured and applied to the pad using the `GPIO_JTAG_SW_PAD_CFG_JTCK_PULL` bit-field in the `GPIO_JTAG_SW_PAD_CFG` register.
  - A low-pass filter can be enabled or disabled using the `GPIO_JTAG_SW_PAD_CFG_JTCK_LPF` bit in the same register.
- Physical configurations of the GPIOs 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-M33 core debug controller that is on the private peripheral bus. For more information about the debug controller, see Section 3.3.6 “Debug Controller” on page 63.

3.3 Private Peripherals

The RSL15 system’s private peripherals are accessible only to the Arm Cortex-M33 processor. These peripherals include the SysTick timer, nested vector interrupt controller, system control block, memory protection unit, security attribution unit, debug controller, software interrupts, and floating-point extension unit.

3.3.1 SysTick

The Arm Cortex-M33 core peripherals include the system tick (SysTick) count-down timer from the Arm Cortex-M33 processor implementation. This block is a private peripheral, tightly tied to the Arm Cortex-M33 implementation. It is described in the *ARM Cortex-M33 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 7.4.1 “System Clock (SYSCLK)” on page 390. For more information about SLOWCLK, see Section 7.4.3 “Slow Clock (SLOWCLK)” on page 391.

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:

$$DELAY = (SysTick\_Load + 1)/fSYSCLK\ or\ STCLK$$

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 reading of this register; the bit 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-M33 processor is gated due to the use of a wait-for-interrupt (WFI) or wait-for-event (WFE) instruction, the SysTick timer is clocked at a much slower rate while waiting for the interrupt or event to occur. When 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 needs to be faster.

For this peripheral's registers, see Section 3.4.1 "SysTick Registers" on page 66.

3.3.2 Nested Vector Interrupt Controller (NVIC)

The Arm Cortex-M33 processor is closely tied to a nested vectored interrupt controller (NVIC), which is a private peripheral provided for the processor that supports interrupt and fault handling functionality. This block is implemented with the Arm Cortex-M33 processor and is described in the *ARM Cortex-M33 Technical Reference Manual*.

The Arm Cortex-M33 processor as implemented for RSL15 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), predefined interrupts, several faults, and a set of general-purpose interrupts that are external to the Arm Cortex-M33 processor and are linked to its interfaces and peripherals. A list of the interrupts supported by the NVIC for the Arm Cortex-M33 processor is provided in the "Interrupts in the Arm Cortex-M33 Processor" table (Table 2)

IMPORTANT: For best practices in error, fault, and watchdog interrupt handling see Chapter 1 "Diagnostic Strategies" on page 1 from the *RSL15 Developer's Guide*.

Table 2. Interrupts in the Arm Cortex-M33 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

Table 2. Interrupts in the Arm Cortex-M33 Processor (Continued)

Interrupt Enumeration Define	Enumeration Value	Vector Number	Description
MemoryManagement_IRQn	-12	4	Memory management interrupt
BusFault_IRQn	-11	5	Bus fault interrupt
UsageFault_IRQn	-10	6	Usage fault interrupt
SecureFault_IRQn	-9	7	Secure 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
LSAD_MONITOR_IRQn	3	19	LSAD and voltage monitoring 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
FIFO_IRQn	8	24	FIFO interrupt
GPIO0_IRQn	9	25	GPIO 0 interrupt
GPIO1_IRQn	10	26	GPIO 1 interrupt
GPIO2_IRQn	11	27	GPIO 2 interrupt
GPIO3_IRQn	12	28	GPIO 3 interrupt
WATCHDOG_IRQn	13	29	Watchdog interrupt
SPI0_RX_IRQn	14	30	SPI0 receive interrupt
SPI0_TX_IRQn	15	31	SPI0 transmit interrupt
SPI0_COM_IRQn	16	32	SPI0 common interrupt
SPI1_RX_IRQn	17	33	SPI1 receive interrupt
SPI1_TX_IRQn	18	34	SPI1 transmit interrupt
SPI1_COM_IRQn	19	35	SPI1 common interrupt
I2C0_IRQn	20	36	I^2C 0 interrupt

Table 2. Interrupts in the Arm Cortex-M33 Processor (Continued)

Interrupt Enumeration Define	Enumeration Value	Vector Number	Description
I2C1_IRQn	21	37	I^2C 1 interrupt
UART0_RX_IRQn	22	38	UART receive interrupt
UART0_TX_IRQn	23	39	UART transmit interrupt
UART0_ERROR_IRQn	24	40	UART common interrupt
PCM0_RX_TX_IRQn	25	41	PCM receive/transmit interrupt
PCM0_ERROR_IRQn	26	42	PCM error interrupt
LIN0_IRQn	27	43	LIN interrupt
FPU_IRQn	28	44	Floating point unit exception interrupt
CC312_IRQn	29	45	CryptoCell (CC312) interrupt
ASCC_PHASE_IRQn	30	46	Asynchronous clock counter phase interrupt
ASCC_PERIOD_IRQn	31	47	Asynchronous clock counter period interrupt
BLE_SW_IRQn	32	48	Bluetooth Low Energy SW triggered interrupt
BLE_FINETGT_IRQn	33	49	Bluetooth Low Energy fine timer target interrupt
BLE_TIMESTAMP_TGT1_IRQn	34	50	Bluetooth Low Energy time stamp target 1 interrupt (generated at a defined instant)
BLE_TIMESTAMP_TGT2_IRQn	35	51	Bluetooth Low Energy time stamp target 2 interrupt (generated at a defined instant)
BLE_CRYPT_IRQn	36	52	Bluetooth Low Energy AES ciphering complete interrupt
BLE_SLP_IRQn	37	53	Bluetooth Low Energy sleep mode interrupt (generated at the end of a sleep period)
BLE_HSLOT_IRQn	38	54	Bluetooth Low Energy half slot interrupt (generated every 312.5 µs)
BLE_FIFO_IRQn	39	55	Bluetooth Low Energy FIFO activity interrupt
BLE_ERROR_IRQn	40	56	Bluetooth Low Energy error interrupt
BLE_COEX_IN_PROCESS_IRQn	41	57	RF coexistence - Bluetooth Low Energy technology in process interrupt
BLE_COEX_RX_TX_IRQn	42	58	RF coexistence - Bluetooth Low Energy technology start/stop Rx or Tx interrupt
TOF_IRQn	43	59	Time of flight counter interrupt
RF_TX_IRQn	44	60	RF end of packet transmission transmit interrupt
RF_RXSTOP_IRQn	45	61	RF receive stop interrupt
RF_IRQ_RECEIVED_IRQn	46	62	RF received packet interrupt

Table 2. Interrupts in the Arm Cortex-M33 Processor (Continued)

Interrupt Enumeration Define	Enumeration Value	Vector Number	Description
RF_SYNC_IRQn	47	63	RF received sync word interrupt
RF_TXFIFO_IRQn	48	64	RF Tx FIFO near underflow detect interrupt
RF_RXFIFO_IRQn	49	64	RF Rx FIFO near overflow detect interrupt
FLASH0_COPY_IRQn	50	66	Flash copy interrupt
FLASH0_ECC_IRQn	51	67	Flash ECC event interrupt
ACCESS_ERROR_IRQn	52	68	Memory and peripheral error interrupt
DMA0_IRQn	53	69	DMA 0 interrupt
DMA1_IRQn	54	70	DMA 1 interrupt
DMA2_IRQn	55	71	DMA 2 interrupt
DMA3_IRQn	56	72	DMA 3 interrupt

3.3.2.1 Interrupt Priority Registers

Use the Interrupt Priority Registers (`NVIC_IP*`) 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 72) listed in Table 2.

NOTE: Configuration of the interrupt priorities for standard Arm Cortex-M33 processor exceptions (vectors 4 to 15) are set using the System Handler Priority registers. For more information, see the *ARMv8M Architecture Reference Manual*.

The NVIC for the Arm Cortex-M33 processor in the RSL15 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 interrupt priority settings are divided into interrupt priority groups, configured using the `SCB_AIRCR` register, with preemption in lower numbered groups with interrupts within a group sorted by interrupt priority settings and finally if required sorted by vector number. Interrupts can be further prioritized to deprioritize all non-secure interrupts when one or more secure interrupts are pending. 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.

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.

For this peripheral's registers, see Section 3.4.2 "NVIC Registers" on page 68.

3.3.3 System Control

The Arm Cortex-M33 processor is supported by a system control block (SCB) that supports configuration information and functionality defined by the RSL15 implementation. This includes configuration of:

`ICB_ICTR`
: The Interrupt Controller Type register (also referred to as the `SCB_ICTR` register), which indicates the number of interrupts supported by the NVIC. RSL15 uses 57 external interrupts (`NVIC_INTLINESNUM_33_64`, hex value 0x1)

`SCB_CPUID`
: The CPU ID base register, containing the version of the Arm Cortex-M33 processor used in RSL15

`SCB_ICSR`
: The Interrupt Control and State register, containing system state information

`SCB_VTOR`
: The Vector Table Offset register, containing a pointer to the active vector table; this register is banked, with separate copies maintained by the system for both secure and non-secure execution.

`SCB_AIRCR, SCB_SHPR*`
: The Application Interrupt and Reset Control register, used to configure the NVIC priority configuration (as described in Section 3.3.2 “Nested Vector Interrupt Controller (NVIC)” on page 58), to clear all active interrupts, and to apply requests for a software reset of the device. This register pairs with the System Handler Priority registers, which set the equivalent priority settings for system interrupts and exceptions.

`SCB_SCR`
: The System Control register, used to configure the Arm Cortex-M33 core sleep behaviors

`SCB_CCR`
: The Configuration Control register, used to configure other Arm Cortex-M33 core behaviors

`SCB_CFSR, SCB_HFSR, SCB_DFSR, SCB_MMFAR, SCB_BFAR`
: The fault status and address registers, used for handling exceptions encountered by the Arm Cortex-M33 processor

`SCB_NSACR`
: Non-Secure Access Control register, used to configure whether the floating point extension can be available to an application using a non-secure execution state.

For more information about the SCB, refer to the *Arm v8-M Architecture Reference Manual* and the *Arm Cortex-M33 Processor Technical Reference Manual*.

3.3.4 Memory Protection Unit

The Arm Cortex-M33 processor implementation in RSL15 is supported by a protected memory system architecture centered around a memory protection unit (MPU). When used, the MPU provides support for protected memory regions and access permission control. When active, the MPU invokes a MemManage fault when a memory access occurs outside of a valid address that is accessible to the processor in the current system configuration (and execution state).

For information on using the memory protection unit as part of a secure/non-secure application pair, see Section 4.4 “TrustZone” on page 119. For more general information on the MPU, refer to the *Arm v8-M Architecture Reference Manual* and the *Arm Cortex-M33 Processor Technical Reference Manual*.

IMPORTANT: For best practices in error, fault, and watchdog interrupt handling see Chapter 1 "Diagnostic Strategies" on page 1 from the *RSL15 Developer's Guide*.

For this peripheral’s registers, see Section 3.4.3 “Memory Protection Unit Registers” on page 104.

3.3.5 Security Attribution Unit

The Arm Cortex-M33 processor implementation in RSL15 is supported by a programmable security attribution unit (SAU), which is used for controlling accesses to items mapped into the Arm Cortex-M33 processor's memory-map. The SAU supports the definition of between 0 and 4 regions of memory that can be made accessible for use with TrustZone non-secure applications.

For instructions and data, the SAU returns the security attribute that is associated with the address.

- For instructions, the attribute determines the allowable security state of the processor when the instruction is executed. It can also identify whether code at a secure address can be called from a non-secure state.
- For data, the attribute determines whether a memory address can be accessed from the non-secure execution state, and whether the external memory request is marked as secure or non-secure.
- If a data access is made from the non-secure execution state to an address marked as secure, then a `SecureFault` exception is taken by the processor. If a data access is made from the secure execution state to an address marked as non-secure, then the associated memory access is marked as non-secure.

For information on using the security attribution unit as part of a secure/non-secure application pair, see Section 4.4 “TrustZone” on page 119. For more general information on the SAU, refer to the *Arm v8-M Architecture Reference Manual* and the *Arm Cortex-M33 Processor Technical Reference Manual*.

IMPORTANT: For best practices in error, fault, and watchdog interrupt handling see Chapter 1 "Diagnostic Strategies" on page 1 from the *RSL15 Developer's Guide*.

For this peripheral’s registers, see Section 3.4.4 “Security Attribution Unit Registers” on page 108.

3.3.6 Debug Controller

Access to the debug controller that underlies the debug port is restricted in many of the security lifecycle states. For more information, see Chapter 4 "Arm TrustZone CryptoCell-312 Security IP" on page 117.

3.3.6.1 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-M33 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 to 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 RSL15 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
- A bus fault
- Usage faults for:
    - No coprocessor errors
    - Unaligned accesses or division by 0
    - State errors
- A secure fault
- Errors when handling an interrupt or exception
- A Hard Fault

The DHCSR also provides a variety of debug-related status information, including:

- Whether the core has been reset or is resetting (CoreDebug_DHCSR_S_RESET_ST); this bit is cleared when read
- Whether an instruction has completed execution since this register has been last read; this bit is cleared when read
- Whether the core is in a locked state
- Whether the core is in Sleep Mode
- Whether the core has been halted
- Whether the most recent register read/write has completed; for more information, see Section 3.3.6.3 “Arm Cortex-M33 Processor Core Register Access” on page 65.

NOTE: The DHCSR and all DEMCR bits that are not related to the debug monitor are only reset if a POR or similar event occurs. These registers are not reset for a core reset. For more information about resets, see Section 8.4 “Resets” on page 427.

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-M33 processor debug port.

3.3.6.2 Debug Monitor Configuration

The NVIC from the Arm Cortex-M33 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 has been caused by a manual request or a debug event (including a debug trap).

3.3.6.3 Arm Cortex-M33 Processor Core Register Access

The Arm Cortex-M33 processor's debug port includes a pair of registers that the debug port uses to provide read and write access to the Arm Cortex-M33 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 from or written to, 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 the "Debug Core Register Data Register Special Register Mapping" table (Table 3).

Table 3. Debug Core Register Data Register Special Register Mapping

Bit Field	Arm Cortex-M33 Processor Register
31:24	CONTROL
23:16	FAULTMASK
15:8	BASEPRI
7:0	PRIMASK

3.3.6.4 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 3.3.3 "System Control" on page 62 for descriptions of the events) if those events are enabled in the Debug Exception and Monitor Control Register (see Section 3.3.6.1 "Halting Debug Configuration and Status" on page 63).

If not interpreted as a Hard Fault, the debug event is ignored.

This register is part of the Arm Cortex-M33 processor's SCB register block.

For this peripheral's registers, see Section 3.4.5 "Debug Control Block Registers" on page 110.

3.3.7 Software Interrupts

The Arm Cortex-M33 processor is supported by a software trigger interrupt register that can be used by software to trigger any external interrupt. Values to be written to this register are compatible with the `IRQn` enumeration provided by *rsl15_vectors.h* and the positive enumeration values listed in Section 3.3.2 "Nested Vector Interrupt Controller (NVIC)" on page 58.

For the software interrupt registers, see Section 3.4.6 "Software Interrupts Registers" on page 114.

3.3.8 Floating-Point Extension

The Arm Cortex-M33 processor's floating point unit is an extension to the Arm Cortex-M33 design that provides a single precision floating point implementation. This implementation includes floating-point computation functionality that is compliant with the ANSI/IEEE Std 754-2008, IEEE Standard for Binary Floating-Point Arithmetic, referred to as the IEEE 754 standard.

The floating point extension provides native support for single-precision add, subtract, multiply, divide, multiply and accumulate, and square root operations on floating point numbers. It also provides conversions between fixed-point and floating-point data formats, and floating-point constant instructions.

For more information on the floating point extension, refer to the *Arm v8-M Architecture Reference Manual* and the *Arm Cortex-M33 Processor Technical Reference Manual*.

For this peripheral's registers, see Section 3.4.7 "Floating Point Extension Registers" on page 114.

3.4 Private Peripherals Registers

For the user's convenience, the registers for the private peripherals are grouped in separate sections according to peripheral.

NOTE: Arm's NVIC private peripheral includes registers supporting up to 256 external interrupts (described in the *ARM Cortex-M33 Technical Reference Manual*). Registers that are explicitly used by RSL15 are shown in Section 3.4.2 "NVIC Registers" on page 68. Additional registers can be used in the NVIC support firmware, to enable compatibility with all of onsemi's Arm Cortex-M-series-based devices.

3.4.1 SysTick 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

3.4.1.1 SysTick_CTRL

Bit Field	Read/Write	Field Name	Description
16	R	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	RW	CLKSOURCE	SYSTICK timer clock source
1	RW	TICKINT	SYSTICK timer interrupt enable
0	RW	ENABLE	SYSTICK timer enable

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
16	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
2	CLKSOURCE	SYSTICK_CLKSOURCE_EXTREF_CLK	Use external reference clock (STCLK)	0x0*
		SYSTICK_CLKSOURCE_CORE_CLK	Use the core clock	0x1
1	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
0	ENABLE	SYSTICK_DISABLE	Disable SYSTICK timer	0x0*
		SYSTICK_ENABLE	Enable SYSTICK Timer	0x1

3.4.1.2 SysTick_LOAD

Bit Field	Read/Write	Field Name	Description
23:0	RW	RELOAD	Counter reload value for the SYSTICK timer when it reaches 0

3.4.1.3 SysTick_VAL

Bit Field	Read/Write	Field Name	Description
23:0	RW	CURRENT	Current value of the SYSTICK counter value. Write to clear counter.

3.4.1.4 SysTick_CALIB

Bit Field	Read/Write	Field Name	Description
31	R	NOREF	Indicates if a reference clock is available
30	R	SKEW	Indicates if calibration value is exactly 10 ms or not
23:0	R	TENMS	SYSTICK counter calibration value for 10 ms. A value of 0 means the calibration value is not available

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31	NOREF	SYSTICK_NOREF	No external reference clock available	0x1
		SYSTICK_REF	External reference clock available	0x0*
30	SKEW	SYSTICK_SKEW	Calibration value is not exactly 10 ms	0x1*
		SYSTICK_NOSKEW	Calibration value is exactly 10 ms	0x0

3.4.2 NVIC Registers

NOTE: Individual register tables are not included for the _NS registers listed here, as the registers are all identical except for their offsets. For more information, see Section 4.4 “TrustZone” on page 119.

Register Name	Register Description	Address
NVIC_ISER0	NVIC External Interrupt Set Enable Register 0	0xE000E100
NVIC_ISER1	NVIC External Interrupt Set Enable Register 1	0xE000E104
NVIC_ICER0	NVIC External Interrupt Clear Enable Register 0	0xE000E180
NVIC_ICER1	NVIC External Interrupt Clear Enable Register 1	0xE000E184
NVIC_ISPR0	NVIC External Interrupt Set Pending Register 0	0xE000E200
NVIC_ISPR1	NVIC External Interrupt Set Pending Register 1	0xE000E204
NVIC_ICPR0	NVIC External Interrupt Clear Pending Register 0	0xE000E280
NVIC_ICPR1	NVIC External Interrupt Clear Pending Register 1	0xE000E284
NVIC_IABR0	NVIC External Interrupt Active Register 0	0xE000E300
NVIC_IABR1	NVIC External Interrupt Active Register 1	0xE000E304
NVIC_ITNS0	NVIC External Interrupt Non-secure Register 0	0xE000E380
NVIC_ITNS1	NVIC External Interrupt Non-secure Register 1	0xE000E384
NVIC_IPR0	NVIC External Interrupt Priority Register 0	0xE000E400
NVIC_IPR1	NVIC External Interrupt Priority Register 1	0xE000E404
NVIC_IPR2	NVIC External Interrupt Priority Register 2	0xE000E408

Register Name	Register Description	Address
NVIC_IPR3	NVIC External Interrupt Priority Register 3	0xE000E40C
NVIC_IPR4	NVIC External Interrupt Priority Register 4	0xE000E410
NVIC_IPR5	NVIC External Interrupt Priority Register 5	0xE000E414
NVIC_IPR6	NVIC External Interrupt Priority Register 6	0xE000E418
NVIC_IPR7	NVIC External Interrupt Priority Register 7	0xE000E41C
NVIC_IPR8	NVIC External Interrupt Priority Register 8	0xE000E420
NVIC_IPR9	NVIC External Interrupt Priority Register 9	0xE000E424
NVIC_IPR10	NVIC External Interrupt Priority Register 10	0xE000E428
NVIC_IPR11	NVIC External Interrupt Priority Register 11	0xE000E42C
NVIC_IPR12	NVIC External Interrupt Priority Register 12	0xE000E430
NVIC_IPR13	NVIC External Interrupt Priority Register 13	0xE000E434
NVIC_IPR14	NVIC External Interrupt Priority Register 14	0xE000E438
NVIC_ISER0_NS	NVIC External Interrupt Set Enable Register 0	0xE002E100
NVIC_ISER1_NS	NVIC External Interrupt Set Enable Register 1	0xE002E104
NVIC_ICER0_NS	NVIC External Interrupt Clear Enable Register 0	0xE002E180
NVIC_ICER1_NS	NVIC External Interrupt Clear Enable Register 1	0xE002E184
NVIC_ISPR0_NS	NVIC External Interrupt Set Pending Register 0	0xE002E200
NVIC_ISPR1_NS	NVIC External Interrupt Set Pending Register 1	0xE002E204
NVIC_ICPR0_NS	NVIC External Interrupt Clear Pending Register 0	0xE002E280
NVIC_ICPR1_NS	NVIC External Interrupt Clear Pending Register 1	0xE002E284
NVIC_IABR0_NS	NVIC External Interrupt Active Register 0	0xE002E300
NVIC_IABR1_NS	NVIC External Interrupt Active Register 1	0xE002E304
NVIC_IPR0_NS	NVIC External Interrupt Priority Register 0	0xE002E400
NVIC_IPR1_NS	NVIC External Interrupt Priority Register 1	0xE002E404
NVIC_IPR2_NS	NVIC External Interrupt Priority Register 2	0xE002E408
NVIC_IPR3_NS	NVIC External Interrupt Priority Register 3	0xE002E40C
NVIC_IPR4_NS	NVIC External Interrupt Priority Register 4	0xE002E410
NVIC_IPR5_NS	NVIC External Interrupt Priority Register 5	0xE002E414
NVIC_IPR6_NS	NVIC External Interrupt Priority Register 6	0xE002E418
NVIC_IPR7_NS	NVIC External Interrupt Priority Register 7	0xE002E41C
NVIC_IPR8_NS	NVIC External Interrupt Priority Register 8	0xE002E420

Register Name	Register Description	Address
NVIC_IPR9_NS	NVIC External Interrupt Priority Register 9	0xE002E424
NVIC_IPR10_NS	NVIC External Interrupt Priority Register 10	0xE002E428
NVIC_IPR11_NS	NVIC External Interrupt Priority Register 11	0xE002E42C
NVIC_IPR12_NS	NVIC External Interrupt Priority Register 12	0xE002E430
NVIC_IPR13_NS	NVIC External Interrupt Priority Register 13	0xE002E434
NVIC_IPR14_NS	NVIC External Interrupt Priority Register 14	0xE002E538

3.4.2.1 NVIC_ISER0

Bit Field	Read/Write	Field Name	Description
31	RW	ASCC_PHASE	ASCC_PHASE interrupt set enable
30	RW	ASCC_PERIOD	ASCC_PERIOD interrupt set enable
29	RW	CC312	CC312 interrupt set enable
28	RW	FPU	FPU interrupt set enable
27	RW	LIN0	LIN0 interrupt set enable
26	RW	PCM0_ERROR	PCM0_ERROR interrupt set enable
25	RW	PCM0_RX_TX	PCM0_RX_TX interrupt set enable
24	RW	UART0_ERROR	UART0_ERROR interrupt set enable
23	RW	UART0_TX	UART0_TX interrupt set enable
22	RW	UART0_RX	UART0_RX interrupt set enable
21	RW	I2C1	I2C1 interrupt set enable
20	RW	I2C0	I2C0 interrupt set enable
19	RW	SPI1_COM	SPI1_COM interrupt set enable
18	RW	SPI1_TX	SPI1_TX interrupt set enable
17	RW	SPI1_RX	SPI1_RX interrupt set enable
16	RW	SPI0_COM	SPI0_COM interrupt set enable
15	RW	SPI0_TX	SPI0_TX interrupt set enable
14	RW	SPI0_RX	SPI0_RX interrupt set enable
13	RW	WATCHDOG	WATCHDOG interrupt set enable
12	RW	GPIO3	GPIO3 interrupt set enable
11	RW	GPIO2	GPIO2 interrupt set enable
10	RW	GPIO1	GPIO1 interrupt set enable
9	RW	GPIO0	GPIO0 interrupt set enable

Bit Field	Read/Write	Field Name	Description
8	RW	FIFO	FIFO interrupt set enable
7	RW	TIMER3	TIMER3 interrupt set enable
6	RW	TIMER2	TIMER2 interrupt set enable
5	RW	TIMER1	TIMER1 interrupt set enable
4	RW	TIMER0	TIMER0 interrupt set enable
3	RW	LSAD_MONITOR	LSAD_MONITOR interrupt set enable
2	RW	RTC_CLOCK	RTC_CLOCK interrupt set enable
1	RW	RTC_ALARM	RTC_ALARM interrupt set enable
0	RW	WAKEUP	WAKEUP interrupt set enable

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31	ASCC_PHASE	NVIC_ASCC_PHASE_INT_ENABLE	Enable the ASCC_PHASE interrupt	0x1
30	ASCC_PERIOD	NVIC_ASCC_PERIOD_INT_ENABLE	Enable the ASCC_PERIOD interrupt	0x1
29	CC312	NVIC_CC312_INT_ENABLE	Enable the CC312 interrupt	0x1
28	FPU	NVIC_FPU_INT_ENABLE	Enable the FPU interrupt	0x1
27	LIN0	NVIC_LIN0_INT_ENABLE	Enable the LIN0 interrupt	0x1
26	PCM0_ERROR	NVIC_PCM0_ERROR_INT_ENABLE	Enable the PCM0_ERROR interrupt	0x1
25	PCM0_RX_TX	NVIC_PCM0_RX_TX_INT_ENABLE	Enable the PCM0_RX_TX interrupt	0x1
24	UART0_ERROR	NVIC_UART0_ERROR_INT_ENABLE	Enable the UART0_ERROR interrupt	0x1
23	UART0_TX	NVIC_UART0_TX_INT_ENABLE	Enable the UART0_TX interrupt	0x1
22	UART0_RX	NVIC_UART0_RX_INT_ENABLE	Enable the UART0_RX interrupt	0x1
21	I2C1	NVIC_I2C1_INT_ENABLE	Enable the I2C1 interrupt	0x1
20	I2C0	NVIC_I2C0_INT_ENABLE	Enable the I2C0 interrupt	0x1
19	SPI1_COM	NVIC_SPI1_COM_INT_ENABLE	Enable the SPI1_COM interrupt	0x1
18	SPI1_TX	NVIC_SPI1_TX_INT_ENABLE	Enable the SPI1_TX interrupt	0x1
17	SPI1_RX	NVIC_SPI1_RX_INT_ENABLE	Enable the SPI1_RX interrupt	0x1
16	SPI0_COM	NVIC_SPI0_COM_INT_ENABLE	Enable the SPI0_COM interrupt	0x1
15	SPI0_TX	NVIC_SPI0_TX_INT_ENABLE	Enable the SPI0_TX interrupt	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
14	SPI0_RX	NVIC_SPI0_RX_INT_ENABLE	Enable the SPI0_RX interrupt	0x1
13	WATCHDOG	NVIC_WATCHDOG_INT_ENABLE	Enable the WATCHDOG interrupt	0x1
12	GPIO3	NVIC_GPIO3_INT_ENABLE	Enable the GPIO3 interrupt	0x1
11	GPIO2	NVIC_GPIO2_INT_ENABLE	Enable the GPIO2 interrupt	0x1
10	GPIO1	NVIC_GPIO1_INT_ENABLE	Enable the GPIO1 interrupt	0x1
9	GPIO0	NVIC_GPIO0_INT_ENABLE	Enable the GPIO0 interrupt	0x1
8	FIFO	NVIC_FIFO_INT_ENABLE	Enable the FIFO interrupt	0x1
7	TIMER3	NVIC_TIMER3_INT_ENABLE	Enable the TIMER3 interrupt	0x1
6	TIMER2	NVIC_TIMER2_INT_ENABLE	Enable the TIMER2 interrupt	0x1
5	TIMER1	NVIC_TIMER1_INT_ENABLE	Enable the TIMER1 interrupt	0x1
4	TIMER0	NVIC_TIMER0_INT_ENABLE	Enable the TIMER0 interrupt	0x1
3	LSAD_MONITOR	NVIC_LSAD_MONITOR_INT_ENABLE	Enable the LSAD_MONITOR interrupt	0x1
2	RTC_CLOCK	NVIC_RTC_CLOCK_INT_ENABLE	Enable the RTC_CLOCK interrupt	0x1
1	RTC_ALARM	NVIC_RTC_ALARM_INT_ENABLE	Enable the RTC_ALARM interrupt	0x1
0	WAKEUP	NVIC_WAKEUP_INT_ENABLE	Enable the WAKEUP interrupt	0x1

3.4.2.2 NVIC_ISER1

Bit Field	Read/Write	Field Name	Description
24	RW	DMA3	DMA3 interrupt set enable
23	RW	DMA2	DMA2 interrupt set enable
22	RW	DMA1	DMA1 interrupt set enable
21	RW	DMA0	DMA0 interrupt set enable
20	RW	ACCESS_ERROR	ACCESS_ERROR interrupt set enable
19	RW	FLASH0_ECC	FLASH0_ECC interrupt set enable
18	RW	FLASH0_COPY	FLASH0_COPY interrupt set enable
17	RW	RF_RXFIFO	RF_RXFIFO interrupt set enable
16	RW	RF_TXFIFO	RF_TXFIFO interrupt set enable
15	RW	RF_SYNC	RF_SYNC interrupt set enable
14	RW	RF_IRQ_RECEIVED	RF_IRQ_RECEIVED interrupt set enable
13	RW	RF_RXSTOP	RF_RXSTOP interrupt set enable

Bit Field	Read/Write	Field Name	Description
12	RW	RF_TX	RF_TX interrupt set enable
11	RW	TOF	TOF interrupt set enable
10	RW	BLE_COEX_RX_TX	BLE_COEX_RX_TX interrupt set enable
9	RW	BLE_COEX_IN_PROCESS	BLE_COEX_IN_PROCESS interrupt set enable
8	RW	BLE_ERROR	BLE_ERROR interrupt set enable
7	RW	BLE_FIFO	BLE_FIFO interrupt set enable
6	RW	BLE_HSLOT	BLE_HSLOT interrupt set enable
5	RW	BLE_SLP	BLE_SLP interrupt set enable
4	RW	BLE_CRYPT	BLE_CRYPT interrupt set enable
3	RW	BLE_TIMESTAMP_TGT2	BLE_TIMESTAMP_TGT2 interrupt set enable
2	RW	BLE_TIMESTAMP_TGT1	BLE_TIMESTAMP_TGT1 interrupt set enable
1	RW	BLE_FINETGT	BLE_FINETGT interrupt set enable
0	RW	BLE_SW	BLE_SW interrupt set enable

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
24	DMA3	NVIC_DMA3_INT_ENABLE	Enable the DMA3 interrupt	0x1
23	DMA2	NVIC_DMA2_INT_ENABLE	Enable the DMA2 interrupt	0x1
22	DMA1	NVIC_DMA1_INT_ENABLE	Enable the DMA1 interrupt	0x1
21	DMA0	NVIC_DMA0_INT_ENABLE	Enable the DMA0 interrupt	0x1
20	ACCESS_ERROR	NVIC_ACCESS_ERROR_INT_ENABLE	Enable the ACCESS_ERROR interrupt	0x1
19	FLASH0_ECC	NVIC_FLASH0_ECC_INT_ENABLE	Enable the FLASH0_ECC interrupt	0x1
18	FLASH0_COPY	NVIC_FLASH0_COPY_INT_ENABLE	Enable the FLASH0_COPY interrupt	0x1
17	RF_RXFIFO	NVIC_RF_RXFIFO_INT_ENABLE	Enable the RF_RXFIFO interrupt	0x1
16	RF_TXFIFO	NVIC_RF_TXFIFO_INT_ENABLE	Enable the RF_TXFIFO interrupt	0x1
15	RF_SYNC	NVIC_RF_SYNC_INT_ENABLE	Enable the RF_SYNC interrupt	0x1
14	RF_IRQ_RECEIVED	NVIC_RF_IRQ_RECEIVED_INT_ENABLE	Enable the RF_IRQ_RECEIVED interrupt	0x1
13	RF_RXSTOP	NVIC_RF_RXSTOP_INT_ENABLE	Enable the RF_RXSTOP interrupt	0x1
12	RF_TX	NVIC_RF_TX_INT_ENABLE	Enable the RF_TX interrupt	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
11	TOF	NVIC_TOF_INT_ENABLE	Enable the TOF interrupt	0x1
10	BLE_COEX_RX_TX	NVIC_BLE_COEX_RX_TX_INT_ENABLE	Enable the BLE_COEX_RX_TX interrupt	0x1
9	BLE_COEX_IN_PROCESS	NVIC_BLE_COEX_IN_PROCESS_INT_ENABLE	Enable the BLE_COEX_IN_PROCESS interrupt	0x1
8	BLE_ERROR	NVIC_BLE_ERROR_INT_ENABLE	Enable the BLE_ERROR interrupt	0x1
7	BLE_FIFO	NVIC_BLE_FIFO_INT_ENABLE	Enable the BLE_FIFO interrupt	0x1
6	BLE_HSLOT	NVIC_BLE_HSLOT_INT_ENABLE	Enable the BLE_HSLOT interrupt	0x1
5	BLE_SLP	NVIC_BLE_SLP_INT_ENABLE	Enable the BLE_SLP interrupt	0x1
4	BLE_CRYPT	NVIC_BLE_CRYPT_INT_ENABLE	Enable the BLE_CRYPT interrupt	0x1
3	BLE_TIMESTAMP_TGT2	NVIC_BLE_TIMESTAMP_TGT2_INT_ENABLE	Enable the BLE_TIMESTAMP_TGT2 interrupt	0x1
2	BLE_TIMESTAMP_TGT1	NVIC_BLE_TIMESTAMP_TGT1_INT_ENABLE	Enable the BLE_TIMESTAMP_TGT1 interrupt	0x1
1	BLE_FINETGT	NVIC_BLE_FINETGT_INT_ENABLE	Enable the BLE_FINETGT interrupt	0x1
0	BLE_SW	NVIC_BLE_SW_INT_ENABLE	Enable the BLE_SW interrupt	0x1

3.4.2.3 NVIC_ICER0

Bit Field	Read/Write	Field Name	Description
31	RW	ASCC_PHASE	ASCC_PHASE interrupt clear enable
30	RW	ASCC_PERIOD	ASCC_PERIOD interrupt clear enable
29	RW	CC312	CC312 interrupt clear enable
28	RW	FPU	FPU interrupt clear enable
27	RW	LIN0	LIN0 interrupt clear enable
26	RW	PCM0_ERROR	PCM0_ERROR interrupt clear enable
25	RW	PCM0_RX_TX	PCM0_RX_TX interrupt clear enable
24	RW	UART0_ERROR	UART0_ERROR interrupt clear enable
23	RW	UART0_TX	UART0_TX interrupt clear enable
22	RW	UART0_RX	UART0_RX interrupt clear enable
21	RW	I2C1	I2C1 interrupt clear enable
20	RW	I2C0	I2C0 interrupt clear enable
19	RW	SPI1_COM	SPI1_COM interrupt clear enable

Bit Field	Read/Write	Field Name	Description
18	RW	SPI1_TX	SPI1_TX interrupt clear enable
17	RW	SPI1_RX	SPI1_RX interrupt clear enable
16	RW	SPI0_COM	SPI0_COM interrupt clear enable
15	RW	SPI0_TX	SPI0_TX interrupt clear enable
14	RW	SPI0_RX	SPI0_RX interrupt clear enable
13	RW	WATCHDOG	WATCHDOG interrupt clear enable
12	RW	GPIO3	GPIO3 interrupt clear enable
11	RW	GPIO2	GPIO2 interrupt clear enable
10	RW	GPIO1	GPIO1 interrupt clear enable
9	RW	GPIO0	GPIO0 interrupt clear enable
8	RW	FIFO	FIFO interrupt clear enable
7	RW	TIMER3	TIMER3 interrupt clear enable
6	RW	TIMER2	TIMER2 interrupt clear enable
5	RW	TIMER1	TIMER1 interrupt clear enable
4	RW	TIMER0	TIMER0 interrupt clear enable
3	RW	LSAD_MONITOR	LSAD_MONITOR interrupt clear enable
2	RW	RTC_CLOCK	RTC_CLOCK interrupt clear enable
1	RW	RTC_ALARM	RTC_ALARM interrupt clear enable
0	RW	WAKEUP	WAKEUP interrupt clear enable

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31	ASCC_PHASE	NVIC_ASCC_PHASE_INT_DISABLE	Disable the ASCC_PHASE interrupt	0x1
30	ASCC_PERIOD	NVIC_ASCC_PERIOD_INT_DISABLE	Disable the ASCC_PERIOD interrupt	0x1
29	CC312	NVIC_CC312_INT_DISABLE	Disable the CC312 interrupt	0x1
28	FPU	NVIC_FPU_INT_DISABLE	Disable the FPU interrupt	0x1
27	LIN0	NVIC_LIN0_INT_DISABLE	Disable the LIN0 interrupt	0x1
26	PCM0_ERROR	NVIC_PCM0_ERROR_INT_DISABLE	Disable the PCM0_ERROR interrupt	0x1
25	PCM0_RX_TX	NVIC_PCM0_RX_TX_INT_DISABLE	Disable the PCM0_RX_TX interrupt	0x1
24	UART0_ERROR	NVIC_UART0_ERROR_INT_	Disable the UART0_ERROR interrupt	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		DISABLE		
23	UART0_TX	NVIC_UART0_TX_INT_DISABLE	Disable the UART0_TX interrupt	0x1
22	UART0_RX	NVIC_UART0_RX_INT_DISABLE	Disable the UART0_RX interrupt	0x1
21	I2C1	NVIC_I2C1_INT_DISABLE	Disable the I2C1 interrupt	0x1
20	I2C0	NVIC_I2C0_INT_DISABLE	Disable the I2C0 interrupt	0x1
19	SPI1_COM	NVIC_SPI1_COM_INT_DISABLE	Disable the SPI1_COM interrupt	0x1
18	SPI1_TX	NVIC_SPI1_TX_INT_DISABLE	Disable the SPI1_TX interrupt	0x1
17	SPI1_RX	NVIC_SPI1_RX_INT_DISABLE	Disable the SPI1_RX interrupt	0x1
16	SPI0_COM	NVIC_SPI0_COM_INT_DISABLE	Disable the SPI0_COM interrupt	0x1
15	SPI0_TX	NVIC_SPI0_TX_INT_DISABLE	Disable the SPI0_TX interrupt	0x1
14	SPI0_RX	NVIC_SPI0_RX_INT_DISABLE	Disable the SPI0_RX interrupt	0x1
13	WATCHDOG	NVIC_WATCHDOG_INT_DISABLE	Disable the WATCHDOG interrupt	0x1
12	GPIO3	NVIC_GPIO3_INT_DISABLE	Disable the GPIO3 interrupt	0x1
11	GPIO2	NVIC_GPIO2_INT_DISABLE	Disable the GPIO2 interrupt	0x1
10	GPIO1	NVIC_GPIO1_INT_DISABLE	Disable the GPIO1 interrupt	0x1
9	GPIO0	NVIC_GPIO0_INT_DISABLE	Disable the GPIO0 interrupt	0x1
8	FIFO	NVIC_FIFO_INT_DISABLE	Disable the FIFO interrupt	0x1
7	TIMER3	NVIC_TIMER3_INT_DISABLE	Disable the TIMER3 interrupt	0x1
6	TIMER2	NVIC_TIMER2_INT_DISABLE	Disable the TIMER2 interrupt	0x1
5	TIMER1	NVIC_TIMER1_INT_DISABLE	Disable the TIMER1 interrupt	0x1
4	TIMER0	NVIC_TIMER0_INT_DISABLE	Disable the TIMER0 interrupt	0x1
3	LSAD_MONITOR	NVIC_LSAD_MONITOR_INT_ DISABLE	Disable the LSAD_MONITOR interrupt	0x1
2	RTC_CLOCK	NVIC_RTC_CLOCK_INT_ DISABLE	Disable the RTC_CLOCK interrupt	0x1
1	RTC_ALARM	NVIC_RTC_ALARM_INT_ DISABLE	Disable the RTC_ALARM interrupt	0x1
0	WAKEUP	NVIC_WAKEUP_INT_DISABLE	Disable the WAKEUP interrupt	0x1

3.4.2.4 NVIC_ICER1

Bit Field	Read/Write	Field Name	Description
24	RW	DMA3	DMA3 interrupt clear enable
23	RW	DMA2	DMA2 interrupt clear enable
22	RW	DMA1	DMA1 interrupt clear enable
21	RW	DMA0	DMA0 interrupt clear enable
20	RW	ACCESS_ERROR	ACCESS_ERROR interrupt clear enable
19	RW	FLASH0_ECC	FLASH0_ECC interrupt clear enable
18	RW	FLASH0_COPY	FLASH0_COPY interrupt clear enable
17	RW	RF_RXFIFO	RF_RXFIFO interrupt clear enable
16	RW	RF_TXFIFO	RF_TXFIFO interrupt clear enable
15	RW	RF_SYNC	RF_SYNC interrupt clear enable
14	RW	RF_IRQ_RECEIVED	RF_IRQ_RECEIVED interrupt clear enable
13	RW	RF_RXSTOP	RF_RXSTOP interrupt clear enable
12	RW	RF_TX	RF_TX interrupt clear enable
11	RW	TOF	TOF interrupt clear enable
10	RW	BLE_COEX_RX_TX	BLE_COEX_RX_TX interrupt clear enable
9	RW	BLE_COEX_IN_PROCESS	BLE_COEX_IN_PROCESS interrupt clear enable
8	RW	BLE_ERROR	BLE_ERROR interrupt clear enable
7	RW	BLE_FIFO	BLE_FIFO interrupt clear enable
6	RW	BLE_HSLOT	BLE_HSLOT interrupt clear enable
5	RW	BLE_SLP	BLE_SLP interrupt clear enable
4	RW	BLE_CRYPT	BLE_CRYPT interrupt clear enable
3	RW	BLE_TIMESTAMP_TGT2	BLE_TIMESTAMP_TGT2 interrupt clear enable
2	RW	BLE_TIMESTAMP_TGT1	BLE_TIMESTAMP_TGT1 interrupt clear enable
1	RW	BLE_FINETGT	BLE_FINETGT interrupt clear enable
0	RW	BLE_SW	BLE_SW interrupt clear enable

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
24	DMA3	NVIC_DMA3_INT_DISABLE	Disable the DMA3 interrupt	0x0
23	DMA2	NVIC_DMA2_INT_DISABLE	Disable the DMA2 interrupt	0x1
22	DMA1	NVIC_DMA1_INT_DISABLE	Disable the DMA1 interrupt	0x1
21	DMA0	NVIC_DMA0_INT_DISABLE	Disable the DMA0 interrupt	0x1
20	ACCESS_ERROR	NVIC_ACCESS_ERROR_INT_DISABLE	Disable the ACCESS_ERROR interrupt	0x1
19	FLASH0_ECC	NVIC_FLASH0_ECC_INT_DISABLE	Disable the FLASH0_ECC interrupt	0x1
18	FLASH0_COPY	NVIC_FLASH0_COPY_INT_DISABLE	Disable the FLASH0_COPY interrupt	0x1
17	RF_RXFIFO	NVIC_RF_RXFIFO_INT_DISABLE	Disable the RF_RXFIFO interrupt	0x1
16	RF_TXFIFO	NVIC_RF_TXFIFO_INT_DISABLE	Disable the RF_TXFIFO interrupt	0x1
15	RF_SYNC	NVIC_RF_SYNC_INT_DISABLE	Disable the RF_SYNC interrupt	0x1
14	RF_IRQ_RECEIVED	NVIC_RF_IRQ_RECEIVED_INT_DISABLE	Disable the RF_IRQ_RECEIVED interrupt	0x1
13	RF_RXSTOP	NVIC_RF_RXSTOP_INT_DISABLE	Disable the RF_RXSTOP interrupt	0x1
12	RF_TX	NVIC_RF_TX_INT_DISABLE	Disable the RF_TX interrupt	0x1
11	TOF	NVIC_TOF_INT_DISABLE	Disable the TOF interrupt	0x1
10	BLE_COEX_RX_TX	NVIC_BLE_COEX_RX_TX_INT_DISABLE	Disable the BLE_COEX_RX_TX interrupt	0x1
9	BLE_COEX_IN_PROCESS	NVIC_BLE_COEX_IN_PROCESS_INT_DISABLE	Disable the BLE_COEX_IN_PROCESS interrupt	0x1
8	BLE_ERROR	NVIC_BLE_ERROR_INT_DISABLE	Disable the BLE_ERROR interrupt	0x1
7	BLE_FIFO	NVIC_BLE_FIFO_INT_DISABLE	Disable the BLE_FIFO interrupt	0x1
6	BLE_HSLOT	NVIC_BLE_HSLOT_INT_DISABLE	Disable the BLE_HSLOT interrupt	0x1
5	BLE_SLP	NVIC_BLE_SLP_INT_DISABLE	Disable the BLE_SLP interrupt	0x1
4	BLE_CRYPT	NVIC_BLE_CRYPT_INT_DISABLE	Disable the BLE_CRYPT interrupt	0x1
3	BLE_TIMESTAMP_TGT2	NVIC_BLE_TIMESTAMP_TGT2_INT_DISABLE	Disable the BLE_TIMESTAMP_TGT2 interrupt	0x1
2	BLE_TIMESTAMP_TGT1	NVIC_BLE_TIMESTAMP_TGT1_INT_DISABLE	Disable the BLE_TIMESTAMP_TGT1 interrupt	0x1
1	BLE_FINETGT	NVIC_BLE_FINETGT_INT_DISABLE	Disable the BLE_FINETGT interrupt	0x1
0	BLE_SW	NVIC_BLE_SW_INT_DISABLE	Disable the BLE_SW interrupt	0x1

3.4.2.5 NVIC_ISPR0

Bit Field	Read/Write	Field Name	Description
31	RW	ASCC_PHASE	ASCC_PHASE interrupt set pending
30	RW	ASCC_PERIOD	ASCC_PERIOD interrupt set pending
29	RW	CC312	CC312 interrupt set pending
28	RW	FPU	FPU interrupt set pending
27	RW	LIN0	LIN0 interrupt set pending
26	RW	PCM0_ERROR	PCM0_ERROR interrupt set pending
25	RW	PCM0_RX_TX	PCM0_RX_TX interrupt set pending
24	RW	UART0_ERROR	UART0_ERROR interrupt set pending
23	RW	UART0_TX	UART0_TX interrupt set pending
22	RW	UART0_RX	UART0_RX interrupt set pending
21	RW	I2C1	I2C1 interrupt set pending
20	RW	I2C0	I2C0 interrupt set pending
19	RW	SPI1_COM	SPI1_COM interrupt set pending
18	RW	SPI1_TX	SPI1_TX interrupt set pending
17	RW	SPI1_RX	SPI1_RX interrupt set pending
16	RW	SPI0_COM	SPI0_COM interrupt set pending
15	RW	SPI0_TX	SPI0_TX interrupt set pending
14	RW	SPI0_RX	SPI0_RX interrupt set pending
13	RW	WATCHDOG	WATCHDOG interrupt set pending
12	RW	GPIO3	GPIO3 interrupt set pending
11	RW	GPIO2	GPIO2 interrupt set pending
10	RW	GPIO1	GPIO1 interrupt set pending
9	RW	GPIO0	GPIO0 interrupt set pending
8	RW	FIFO	FIFO interrupt set pending
7	RW	TIMER3	TIMER3 interrupt set pending
6	RW	TIMER2	TIMER2 interrupt set pending
5	RW	TIMER1	TIMER1 interrupt set pending
4	RW	TIMER0	TIMER0 interrupt set pending
3	RW	LSAD_MONITOR	LSAD_MONITOR interrupt set pending

Bit Field	Read/Write	Field Name	Description
2	RW	RTC_CLOCK	RTC_CLOCK interrupt set pending
1	RW	RTC_ALARM	RTC_ALARM interrupt set pending
0	RW	WAKEUP	WAKEUP interrupt set pending

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31	ASCC_PHASE	NVIC_ASCC_PHASE_INT_SETPEND	Set the ASCC_PHASE interrupt pending	0x1
30	ASCC_PERIOD	NVIC_ASCC_PERIOD_INT_SETPEND	Set the ASCC_PERIOD interrupt pending	0x1
29	CC312	NVIC_CC312_INT_SETPEND	Set the CC312 interrupt pending	0x1
28	FPU	NVIC_FPU_INT_SETPEND	Set the FPU interrupt pending	0x1
27	LIN0	NVIC_LIN0_INT_SETPEND	Set the LIN0 interrupt pending	0x1
26	PCM0_ERROR	NVIC_PCM0_ERROR_INT_SETPEND	Set the PCM0_ERROR interrupt pending	0x1
25	PCM0_RX_TX	NVIC_PCM0_RX_TX_INT_SETPEND	Set the PCM0_RX_TX interrupt pending	0x1
24	UART0_ERROR	NVIC_UART0_ERROR_INT_SETPEND	Set the UART0_ERROR interrupt pending	0x1
23	UART0_TX	NVIC_UART0_TX_INT_SETPEND	Set the UART0_TX interrupt pending	0x1
22	UART0_RX	NVIC_UART0_RX_INT_SETPEND	Set the UART0_RX interrupt pending	0x1
21	I2C1	NVIC_I2C1_INT_SETPEND	Set the I2C1 interrupt pending	0x1
20	I2C0	NVIC_I2C0_INT_SETPEND	Set the I2C0 interrupt pending	0x1
19	SPI1_COM	NVIC_SPI1_COM_INT_SETPEND	Set the SPI1_COM interrupt pending	0x1
18	SPI1_TX	NVIC_SPI1_TX_INT_SETPEND	Set the SPI1_TX interrupt pending	0x1
17	SPI1_RX	NVIC_SPI1_RX_INT_SETPEND	Set the SPI1_RX interrupt pending	0x1
16	SPI0_COM	NVIC_SPI0_COM_INT_SETPEND	Set the SPI0_COM interrupt pending	0x1
15	SPI0_TX	NVIC_SPI0_TX_INT_SETPEND	Set the SPI0_TX interrupt pending	0x1
14	SPI0_RX	NVIC_SPI0_RX_INT_SETPEND	Set the SPI0_RX interrupt pending	0x1
13	WATCHDOG	NVIC_WATCHDOG_INT_SETPEND	Set the WATCHDOG interrupt pending	0x1
12	GPIO3	NVIC_GPIO3_INT_SETPEND	Set the GPIO3 interrupt pending	0x1
11	GPIO2	NVIC_GPIO2_INT_SETPEND	Set the GPIO2 interrupt pending	0x1
10	GPIO1	NVIC_GPIO1_INT_SETPEND	Set the GPIO1 interrupt pending	0x1
9	GPIO0	NVIC_GPIO0_INT_SETPEND	Set the GPIO0 interrupt pending	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
8	FIFO	NVIC_FIFO_INT_SETPEND	Set the FIFO interrupt pending	0x1
7	TIMER3	NVIC_TIMER3_INT_SETPEND	Set the TIMER3 interrupt pending	0x1
6	TIMER2	NVIC_TIMER2_INT_SETPEND	Set the TIMER2 interrupt pending	0x1
5	TIMER1	NVIC_TIMER1_INT_SETPEND	Set the TIMER1 interrupt pending	0x1
4	TIMER0	NVIC_TIMER0_INT_SETPEND	Set the TIMER0 interrupt pending	0x1
3	LSAD_MONITOR	NVIC_LSAD_MONITOR_INT_ SETPEND	Set the LSAD_MONITOR interrupt pending	0x1
2	RTC_CLOCK	NVIC_RTC_CLOCK_INT_ SETPEND	Set the RTC_CLOCK interrupt pending	0x1
1	RTC_ALARM	NVIC_RTC_ALARM_INT_ SETPEND	Set the RTC_ALARM interrupt pending	0x1
0	WAKEUP	NVIC_WAKEUP_INT_SETPEND	Set the WAKEUP interrupt pending	0x1

3.4.2.6 NVIC_ISPR1

Bit Field	Read/Write	Field Name	Description
24	RW	DMA3	DMA3 interrupt set pending
23	RW	DMA2	DMA2 interrupt set pending
22	RW	DMA1	DMA1 interrupt set pending
21	RW	DMA0	DMA0 interrupt set pending
20	RW	ACCESS_ERROR	ACCESS_ERROR interrupt set pending
19	RW	FLASH0_ECC	FLASH0_ECC interrupt set pending
18	RW	FLASH0_COPY	FLASH0_COPY interrupt set pending
17	RW	RF_RXFIFO	RF_RXFIFO interrupt set pending
16	RW	RF_TXFIFO	RF_TXFIFO interrupt set pending
15	RW	RF_SYNC	RF_SYNC interrupt set pending
14	RW	RF_IRQ_RECEIVED	RF_IRQ_RECEIVED interrupt set pending
13	RW	RF_RXSTOP	RF_RXSTOP interrupt set pending
12	RW	RF_TX	RF_TX interrupt set pending
11	RW	TOF	TOF interrupt set pending
10	RW	BLE_COEX_RX_TX	BLE_COEX_RX_TX interrupt set pending
9	RW	BLE_COEX_IN_PROCESS	BLE_COEX_IN_PROCESS interrupt set pending
8	RW	BLE_ERROR	BLE_ERROR interrupt set pending
7	RW	BLE_FIFO	BLE_FIFO interrupt set pending

Bit Field	Read/Write	Field Name	Description
6	RW	BLE_HSLOT	BLE_HSLOT interrupt set pending
5	RW	BLE_SLP	BLE_SLP interrupt set pending
4	RW	BLE_CRYPT	BLE_CRYPT interrupt set pending
3	RW	BLE_TIMESTAMP_TGT2	BLE_TIMESTAMP_TGT2 interrupt set pending
2	RW	BLE_TIMESTAMP_TGT1	BLE_TIMESTAMP_TGT1 interrupt set pending
1	RW	BLE_FINETGT	BLE_FINETGT interrupt set pending
0	RW	BLE_SW	BLE_SW interrupt set pending

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
24	DMA3	NVIC_DMA3_INT_SETPEND	Set the DMA3 interrupt pending	0x1
23	DMA2	NVIC_DMA2_INT_SETPEND	Set the DMA2 interrupt pending	0x1
22	DMA1	NVIC_DMA1_INT_SETPEND	Set the DMA1 interrupt pending	0x1
21	DMA0	NVIC_DMA0_INT_SETPEND	Set the DMA0 interrupt pending	0x1
20	ACCESS_ERROR	NVIC_ACCESS_ERROR_INT_SETPEND	Set the ACCESS_ERROR interrupt pending	0x1
19	FLASH0_ECC	NVIC_FLASH0_ECC_INT_SETPEND	Set the FLASH0_ECC interrupt pending	0x1
18	FLASH0_COPY	NVIC_FLASH0_COPY_INT_SETPEND	Set the FLASH0_COPY interrupt pending	0x1
17	RF_RXFIFO	NVIC_RF_RXFIFO_INT_SETPEND	Set the RF_RXFIFO interrupt pending	0x1
16	RF_TXFIFO	NVIC_RF_TXFIFO_INT_SETPEND	Set the RF_TXFIFO interrupt pending	0x1
15	RF_SYNC	NVIC_RF_SYNC_INT_SETPEND	Set the RF_SYNC interrupt pending	0x1
14	RF_IRQ_RECEIVED	NVIC_RF_IRQ_RECEIVED_INT_SETPEND	Set the RF_IRQ_RECEIVED interrupt pending	0x1
13	RF_RXSTOP	NVIC_RF_RXSTOP_INT_SETPEND	Set the RF_RXSTOP interrupt pending	0x1
12	RF_TX	NVIC_RF_TX_INT_SETPEND	Set the RF_TX interrupt pending	0x1
11	TOF	NVIC_TOF_INT_SETPEND	Set the TOF interrupt pending	0x1
10	BLE_COEX_RX_TX	NVIC_BLE_COEX_RX_TX_INT_SETPEND	Set the BLE_COEX_RX_TX interrupt pending	0x1
9	BLE_COEX_IN_PROCESS	NVIC_BLE_COEX_IN_PROCESS_INT_SETPEND	Set the BLE_COEX_IN_PROCESS interrupt pending	0x1
8	BLE_ERROR	NVIC_BLE_ERROR_INT_SETPEND	Set the BLE_ERROR interrupt pending	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
7	BLE_FIFO	NVIC_BLE_FIFO_INT_SETPEND	Set the BLE_FIFO interrupt pending	0x1
6	BLE_HSLOT	NVIC_BLE_HSLOT_INT_SETPEND	Set the BLE_HSLOT interrupt pending	0x1
5	BLE_SLP	NVIC_BLE_SLP_INT_SETPEND	Set the BLE_SLP interrupt pending	0x1
4	BLE_CRYPT	NVIC_BLE_CRYPT_INT_SETPEND	Set the BLE_CRYPT interrupt pending	0x1
3	BLE_TIMESTAMP_TGT2	NVIC_BLE_TIMESTAMP_TGT2_INT_SETPEND	Set the BLE_TIMESTAMP_TGT2 interrupt pending	0x1
2	BLE_TIMESTAMP_TGT1	NVIC_BLE_TIMESTAMP_TGT1_INT_SETPEND	Set the BLE_TIMESTAMP_TGT1 interrupt pending	0x1
1	BLE_FINETGT	NVIC_BLE_FINETGT_INT_SETPEND	Set the BLE_FINETGT interrupt pending	0x1
0	BLE_SW	NVIC_BLE_SW_INT_SETPEND	Set the BLE_SW interrupt pending	0x1

3.4.2.7 NVIC_ICPR0

Bit Field	Read/Write	Field Name	Description
31	RW	ASCC_PHASE	ASCC_PHASE interrupt clear pending
30	RW	ASCC_PERIOD	ASCC_PERIOD interrupt clear pending
29	RW	CC312	CC312 interrupt clear pending
28	RW	FPU	FPU interrupt clear pending
27	RW	LIN0	LIN0 interrupt clear pending
26	RW	PCM0_ERROR	PCM0_ERROR interrupt clear pending
25	RW	PCM0_RX_TX	PCM0_RX_TX interrupt clear pending
24	RW	UART0_ERROR	UART0_ERROR interrupt clear pending
23	RW	UART0_TX	UART0_TX interrupt clear pending
22	RW	UART0_RX	UART0_RX interrupt clear pending
21	RW	I2C1	I2C1 interrupt clear pending
20	RW	I2C0	I2C0 interrupt clear pending
19	RW	SPI1_COM	SPI1_COM interrupt clear pending
18	RW	SPI1_TX	SPI1_TX interrupt clear pending
17	RW	SPI1_RX	SPI1_RX interrupt clear pending
16	RW	SPI0_COM	SPI0_COM interrupt clear pending
15	RW	SPI0_TX	SPI0_TX interrupt clear pending

Bit Field	Read/Write	Field Name	Description
14	RW	SPI0_RX	SPI0_RX interrupt clear pending
13	RW	WATCHDOG	WATCHDOG interrupt clear pending
12	RW	GPIO3	GPIO3 interrupt clear pending
11	RW	GPIO2	GPIO2 interrupt clear pending
10	RW	GPIO1	GPIO1 interrupt clear pending
9	RW	GPIO0	GPIO0 interrupt clear pending
8	RW	FIFO	FIFO interrupt clear pending
7	RW	TIMER3	TIMER3 interrupt clear pending
6	RW	TIMER2	TIMER2 interrupt clear pending
5	RW	TIMER1	TIMER1 interrupt clear pending
4	RW	TIMER0	TIMER0 interrupt clear pending
3	RW	LSAD_MONITOR	LSAD_MONITOR interrupt clear pending
2	RW	RTC_CLOCK	RTC_CLOCK interrupt clear pending
1	RW	RTC_ALARM	RTC_ALARM interrupt clear pending
0	RW	WAKEUP	WAKEUP interrupt clear pending

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31	ASCC_PHASE	NVIC_ASCC_PHASE_INT_CLRPEND	Clear the NVIC_ASCC_PHASE_INT_CLRPEND interrupt status bit	0x1
30	ASCC_PERIOD	NVIC_ASCC_PERIOD_INT_CLRPEND	Clear the NVIC_ASCC_PERIOD_INT_CLRPEND interrupt status bit	0x1
29	CC312	NVIC_CC312_INT_CLRPEND	Clear the NVIC_CC312_INT_CLRPEND interrupt status bit	0x1
28	FPU	NVIC_FPU_INT_CLRPEND	Clear the NVIC_FPU_INT_CLRPEND interrupt status bit	0x1
27	LIN0	NVIC_LIN0_INT_CLRPEND	Clear the NVIC_LIN0_INT_CLRPEND interrupt status bit	0x1
26	PCM0_ERROR	NVIC_PCM0_ERROR_INT_CLRPEND	Clear the NVIC_PCM0_ERROR_INT_CLRPEND interrupt status bit	0x1
25	PCM0_RX_TX	NVIC_PCM0_RX_TX_INT_CLRPEND	Clear the NVIC_PCM0_RX_TX_INT_CLRPEND interrupt status bit	0x1
24	UART0_ERROR	NVIC_UART0_ERROR_INT_CLRPEND	Clear the NVIC_UART0_ERROR_INT_CLRPEND interrupt status bit	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
23	UART0_TX	NVIC_UART0_TX_INT_CLRPEND	Clear the NVIC_UART0_TX_INT_ CLRPEND interrupt status bit	0x1
22	UART0_RX	NVIC_UART0_RX_INT_CLRPEND	Clear the NVIC_UART0_RX_INT_ CLRPEND interrupt status bit	0x1
21	I2C1	NVIC_I2C1_INT_CLRPEND	Clear the NVIC_I2C1_INT_CLRPEND interrupt status bit	0x1
20	I2C0	NVIC_I2C0_INT_CLRPEND	Clear the NVIC_I2C0_INT_CLRPEND interrupt status bit	0x1
19	SPI1_COM	NVIC_SPI1_COM_INT_CLRPEND	Clear the NVIC_SPI1_COM_INT_ CLRPEND interrupt status bit	0x1
18	SPI1_TX	NVIC_SPI1_TX_INT_CLRPEND	Clear the NVIC_SPI1_TX_INT_ CLRPEND interrupt status bit	0x1
17	SPI1_RX	NVIC_SPI1_RX_INT_CLRPEND	Clear the NVIC_SPI1_RX_INT_ CLRPEND interrupt status bit	0x1
16	SPI0_COM	NVIC_SPI0_COM_INT_CLRPEND	Clear the NVIC_SPI0_COM_INT_ CLRPEND interrupt status bit	0x1
15	SPI0_TX	NVIC_SPI0_TX_INT_CLRPEND	Clear the NVIC_SPI0_TX_INT_ CLRPEND interrupt status bit	0x1
14	SPI0_RX	NVIC_SPI0_RX_INT_CLRPEND	Clear the NVIC_SPI0_RX_INT_ CLRPEND interrupt status bit	0x1
13	WATCHDOG	NVIC_WATCHDOG_INT_CLRPEND	Clear the NVIC_WATCHDOG_INT_ CLRPEND interrupt status bit	0x1
12	GPIO3	NVIC_GPIO3_INT_CLRPEND	Clear the NVIC_GPIO3_INT_CLRPEND interrupt status bit	0x1
11	GPIO2	NVIC_GPIO2_INT_CLRPEND	Clear the NVIC_GPIO2_INT_CLRPEND interrupt status bit	0x1
10	GPIO1	NVIC_GPIO1_INT_CLRPEND	Clear the NVIC_GPIO1_INT_CLRPEND interrupt status bit	0x1
9	GPIO0	NVIC_GPIO0_INT_CLRPEND	Clear the NVIC_GPIO0_INT_CLRPEND interrupt status bit	0x1
8	FIFO	NVIC_FIFO_INT_CLRPEND	Clear the NVIC_FIFO_INT_CLRPEND interrupt status bit	0x1
7	TIMER3	NVIC_TIMER3_INT_CLRPEND	Clear the NVIC_TIMER3_INT_ CLRPEND interrupt status bit	0x1
6	TIMER2	NVIC_TIMER2_INT_CLRPEND	Clear the NVIC_TIMER2_INT_ CLRPEND interrupt status bit	0x1
5	TIMER1	NVIC_TIMER1_INT_CLRPEND	Clear the NVIC_TIMER1_INT_	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
			CLRPEND interrupt status bit	
4	TIMER0	NVIC_TIMER0_INT_CLRPEND	Clear the NVIC_TIMER0_INT_ CLRPEND interrupt status bit	0x1
3	LSAD_MONITOR	NVIC_LSAD_MONITOR_INT_ CLRPEND	Clear the NVIC_LSAD_MONITOR_INT_ CLRPEND interrupt status bit	0x1
2	RTC_CLOCK	NVIC_RTC_CLOCK_INT_CLRPEND	Clear the NVIC_RTC_CLOCK_INT_ CLRPEND interrupt status bit	0x1
1	RTC_ALARM	NVIC_RTC_ALARM_INT_CLRPEND	Clear the NVIC_RTC_ALARM_INT_ CLRPEND interrupt status bit	0x1
0	WAKEUP	NVIC_WAKEUP_INT_CLRPEND	Clear the NVIC_WAKEUP_INT_ CLRPEND interrupt status bit	0x1

3.4.2.8 NVIC_ICPR1

Bit Field	Read/Write	Field Name	Description
24	RW	DMA3	DMA3 interrupt clear pending
23	RW	DMA2	DMA2 interrupt clear pending
22	RW	DMA1	DMA1 interrupt clear pending
21	RW	DMA0	DMA0 interrupt clear pending
20	RW	ACCESS_ERROR	ACCESS_ERROR interrupt clear pending
19	RW	FLASH0_ECC	FLASH0_ECC interrupt clear pending
18	RW	FLASH0_COPY	FLASH0_COPY interrupt clear pending
17	RW	RF_RXFIFO	RF_RXFIFO interrupt clear pending
16	RW	RF_TXFIFO	RF_TXFIFO interrupt clear pending
15	RW	RF_SYNC	RF_SYNC interrupt clear pending
14	RW	RF_IRQ_RECEIVED	RF_IRQ_RECEIVED interrupt clear pending
13	RW	RF_RXSTOP	RF_RXSTOP interrupt clear pending
12	RW	RF_TX	RF_TX interrupt clear pending
11	RW	TOF	TOF interrupt clear pending
10	RW	BLE_COEX_RX_TX	BLE_COEX_RX_TX interrupt clear pending
9	RW	BLE_COEX_IN_PROCESS	BLE_COEX_IN_PROCESS interrupt clear pending
8	RW	BLE_ERROR	BLE_ERROR interrupt clear pending
7	RW	BLE_FIFO	BLE_FIFO interrupt clear pending
6	RW	BLE_HSLOT	BLE_HSLOT interrupt clear pending

Bit Field	Read/Write	Field Name	Description
5	RW	BLE_SLP	BLE_SLP interrupt clear pending
4	RW	BLE_CRYPT	BLE_CRYPT interrupt clear pending
3	RW	BLE_TIMESTAMP_TGT2	BLE_TIMESTAMP_TGT2 interrupt clear pending
2	RW	BLE_TIMESTAMP_TGT1	BLE_TIMESTAMP_TGT1 interrupt clear pending
1	RW	BLE_FINETGT	BLE_FINETGT interrupt clear pending
0	RW	BLE_SW	BLE_SW interrupt clear pending

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
24	DMA3	NVIC_DMA3_INT_CLRPEND	Clear the NVIC_DMA3_INT_CLRPEND interrupt status bit	0x1
23	DMA2	NVIC_DMA2_INT_CLRPEND	Clear the NVIC_DMA2_INT_CLRPEND interrupt status bit	0x1
22	DMA1	NVIC_DMA1_INT_CLRPEND	Clear the NVIC_DMA1_INT_CLRPEND interrupt status bit	0x1
21	DMA0	NVIC_DMA0_INT_CLRPEND	Clear the NVIC_DMA0_INT_CLRPEND interrupt status bit	0x1
20	ACCESS_ERROR	NVIC_ACCESS_ERROR_INT_CLRPEND	Clear the NVIC_ACCESS_ERROR_INT_CLRPEND interrupt status bit	0x1
19	FLASH0_ECC	NVIC_FLASH0_ECC_INT_CLRPEND	Clear the NVIC_FLASH0_ECC_INT_CLRPEND interrupt status bit	0x1
18	FLASH0_COPY	NVIC_FLASH0_COPY_INT_CLRPEND	Clear the NVIC_FLASH0_COPY_INT_CLRPEND interrupt status bit	0x1
17	RF_RXFIFO	NVIC_RF_RXFIFO_INT_CLRPEND	Clear the NVIC_RF_RXFIFO_INT_CLRPEND interrupt status bit	0x1
16	RF_TXFIFO	NVIC_RF_TXFIFO_INT_CLRPEND	Clear the NVIC_RF_TXFIFO_INT_CLRPEND interrupt status bit	0x1
15	RF_SYNC	NVIC_RF_SYNC_INT_CLRPEND	Clear the NVIC_RF_SYNC_INT_CLRPEND interrupt status bit	0x1
14	RF_IRQ_RECEIVED	NVIC_RF_IRQ_RECEIVED_INT_CLRPEND	Clear the NVIC_RF_IRQ_RECEIVED_INT_CLRPEND interrupt status bit	0x1
13	RF_RXSTOP	NVIC_RF_RXSTOP_INT_CLRPEND	Clear the NVIC_RF_RXSTOP_INT_CLRPEND interrupt status bit	0x1
12	RF_TX	NVIC_RF_TX_INT_CLRPEND	Clear the NVIC_RF_TX_INT_CLRPEND interrupt status bit	0x1
11	TOF	NVIC_TOF_INT_CLRPEND	Clear the NVIC_TOF_INT_CLRPEND	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
			interrupt status bit	
10	BLE_COEX_RX_TX	NVIC_BLE_COEX_RX_TX_INT_CLRPEND	Clear the NVIC_BLE_COEX_RX_TX_INT_CLRPEND interrupt status bit	0x1
9	BLE_COEX_IN_PROCESS	NVIC_BLE_COEX_IN_PROCESS_INT_CLRPEND	Clear the NVIC_BLE_COEX_IN_PROCESS_INT_CLRPEND interrupt status bit	0x1
8	BLE_ERROR	NVIC_BLE_ERROR_INT_CLRPEND	Clear the NVIC_BLE_ERROR_INT_CLRPEND interrupt status bit	0x1
7	BLE_FIFO	NVIC_BLE_FIFO_INT_CLRPEND	Clear the NVIC_BLE_FIFO_INT_CLRPEND interrupt status bit	0x1
6	BLE_HSLOT	NVIC_BLE_HSLOT_INT_CLRPEND	Clear the NVIC_BLE_HSLOT_INT_CLRPEND interrupt status bit	0x1
5	BLE_SLP	NVIC_BLE_SLP_INT_CLRPEND	Clear the NVIC_BLE_SLP_INT_CLRPEND interrupt status bit	0x1
4	BLE_CRYPT	NVIC_BLE_CRYPT_INT_CLRPEND	Clear the NVIC_BLE_CRYPT_INT_CLRPEND interrupt status bit	0x1
3	BLE_TIMESTAMP_TGT2	NVIC_BLE_TIMESTAMP_TGT2_INT_CLRPEND	Clear the NVIC_BLE_TIMESTAMP_TGT2_INT_CLRPEND interrupt status bit	0x1
2	BLE_TIMESTAMP_TGT1	NVIC_BLE_TIMESTAMP_TGT1_INT_CLRPEND	Clear the NVIC_BLE_TIMESTAMP_TGT1_INT_CLRPEND interrupt status bit	0x1
1	BLE_FINETGT	NVIC_BLE_FINETGT_INT_CLRPEND	Clear the NVIC_BLE_FINETGT_INT_CLRPEND interrupt status bit	0x1
0	BLE_SW	NVIC_BLE_SW_INT_CLRPEND	Clear the NVIC_BLE_SW_INT_CLRPEND interrupt status bit	0x1

3.4.2.9 NVIC_IABR0

Bit Field	Read/Write	Field Name	Description
31	R	ASCC_PHASE	Set the ASCC_PHASE interrupt as active
30	R	ASCC_PERIOD	Set the ASCC_PERIOD interrupt as active
29	R	CC312	Set the CC312 interrupt as active
28	R	FPU	Set the FPU interrupt as active
27	R	LIN0	Set the LIN0 interrupt as active
26	R	PCM0_ERROR	Set the PCM0_ERROR interrupt as active
25	R	PCM0_RX_TX	Set the PCM0_RX_TX interrupt as active

Bit Field	Read/Write	Field Name	Description
24	R	UART0_ERROR	Set the UART0_ERROR interrupt as active
23	R	UART0_TX	Set the UART0_TX interrupt as active
22	R	UART0_RX	Set the UART0_RX interrupt as active
21	R	I2C1	Set the I2C1 interrupt as active
20	R	I2C0	Set the I2C0 interrupt as active
19	R	SPI1_COM	Set the SPI1_COM interrupt as active
18	R	SPI1_TX	Set the SPI1_TX interrupt as active
17	R	SPI1_RX	Set the SPI1_RX interrupt as active
16	R	SPI0_COM	Set the SPI0_COM interrupt as active
15	R	SPI0_TX	Set the SPI0_TX interrupt as active
14	R	SPI0_RX	Set the SPI0_RX interrupt as active
13	R	WATCHDOG	Set the WATCHDOG interrupt as active
12	R	GPIO3	Set the GPIO3 interrupt as active
11	R	GPIO2	Set the GPIO2 interrupt as active
10	R	GPIO1	Set the GPIO1 interrupt as active
9	R	GPIO0	Set the GPIO0 interrupt as active
8	R	FIFO	Set the FIFO interrupt as active
7	R	TIMER3	Set the TIMER3 interrupt as active
6	R	TIMER2	Set the TIMER2 interrupt as active
5	R	TIMER1	Set the TIMER1 interrupt as active
4	R	TIMER0	Set the TIMER0 interrupt as active
3	R	LSAD_MONITOR	Set the LSAD_MONITOR interrupt as active
2	R	RTC_CLOCK	Set the RTC_CLOCK interrupt as active
1	R	RTC_ALARM	Set the RTC_ALARM interrupt as active
0	R	WAKEUP	Set the WAKEUP interrupt as active

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31	ASCC_PHASE	NVIC_ASCC_PHASE_INT_ACTIVE	The NVIC_ASCC_PHASE_INT_ ACTIVE interrupt is active	0x1
30	ASCC_PERIOD	NVIC_ASCC_PERIOD_INT_ ACTIVE	The NVIC_ASCC_PERIOD_INT_ ACTIVE interrupt is active	0x1
29	CC312	NVIC_CC312_INT_ACTIVE	The NVIC_CC312_INT_ACTIVE interrupt is active	0x1
28	FPU	NVIC_FPU_INT_ACTIVE	The NVIC_FPU_INT_ACTIVE interrupt is active	0x1
27	LIN0	NVIC_LIN0_INT_ACTIVE	The NVIC_LIN0_INT_ACTIVE interrupt is active	0x1
26	PCM0_ERROR	NVIC_PCM0_ERROR_INT_ACTIVE	The NVIC_PCM0_ERROR_INT_ ACTIVE interrupt is active	0x1
25	PCM0_RX_TX	NVIC_PCM0_RX_TX_INT_ACTIVE	The NVIC_PCM0_RX_TX_INT_ ACTIVE interrupt is active	0x1
24	UART0_ERROR	NVIC_UART0_ERROR_INT_ ACTIVE	The NVIC_UART0_ERROR_INT_ ACTIVE interrupt is active	0x1
23	UART0_TX	NVIC_UART0_TX_INT_ACTIVE	The NVIC_UART0_TX_INT_ACTIVE interrupt is active	0x1
22	UART0_RX	NVIC_UART0_RX_INT_ACTIVE	The NVIC_UART0_RX_INT_ACTIVE interrupt is active	0x1
21	I2C1	NVIC_I2C1_INT_ACTIVE	The NVIC_I2C1_INT_ACTIVE interrupt is active	0x1
20	I2C0	NVIC_I2C0_INT_ACTIVE	The NVIC_I2C0_INT_ACTIVE interrupt is active	0x1
19	SPI1_COM	NVIC_SPI1_COM_INT_ACTIVE	The NVIC_SPI1_COM_INT_ACTIVE interrupt is active	0x1
18	SPI1_TX	NVIC_SPI1_TX_INT_ACTIVE	The NVIC_SPI1_TX_INT_ACTIVE interrupt is active	0x1
17	SPI1_RX	NVIC_SPI1_RX_INT_ACTIVE	The NVIC_SPI1_RX_INT_ACTIVE interrupt is active	0x1
16	SPI0_COM	NVIC_SPI0_COM_INT_ACTIVE	The NVIC_SPI0_COM_INT_ACTIVE interrupt is active	0x1
15	SPI0_TX	NVIC_SPI0_TX_INT_ACTIVE	The NVIC_SPI0_TX_INT_ACTIVE interrupt is active	0x1
14	SPI0_RX	NVIC_SPI0_RX_INT_ACTIVE	The NVIC_SPI0_RX_INT_ACTIVE interrupt is active	0x1
13	WATCHDOG	NVIC_WATCHDOG_INT_ACTIVE	The NVIC_WATCHDOG_INT_ACTIVE	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
			interrupt is active	
12	GPIO3	NVIC_GPIO3_INT_ACTIVE	The NVIC_GPIO3_INT_ACTIVE interrupt is active	0x1
11	GPIO2	NVIC_GPIO2_INT_ACTIVE	The NVIC_GPIO2_INT_ACTIVE interrupt is active	0x1
10	GPIO1	NVIC_GPIO1_INT_ACTIVE	The NVIC_GPIO1_INT_ACTIVE interrupt is active	0x1
9	GPIO0	NVIC_GPIO0_INT_ACTIVE	The NVIC_GPIO0_INT_ACTIVE interrupt is active	0x1
8	FIFO	NVIC_FIFO_INT_ACTIVE	The NVIC_FIFO_INT_ACTIVE interrupt is active	0x1
7	TIMER3	NVIC_TIMER3_INT_ACTIVE	The NVIC_TIMER3_INT_ACTIVE interrupt is active	0x1
6	TIMER2	NVIC_TIMER2_INT_ACTIVE	The NVIC_TIMER2_INT_ACTIVE interrupt is active	0x1
5	TIMER1	NVIC_TIMER1_INT_ACTIVE	The NVIC_TIMER1_INT_ACTIVE interrupt is active	0x1
4	TIMER0	NVIC_TIMER0_INT_ACTIVE	The NVIC_TIMER0_INT_ACTIVE interrupt is active	0x1
3	LSAD_MONITOR	NVIC_LSAD_MONITOR_INT_ ACTIVE	The NVIC_LSAD_MONITOR_INT_ ACTIVE interrupt is active	0x1
2	RTC_CLOCK	NVIC_RTC_CLOCK_INT_ACTIVE	The NVIC_RTC_CLOCK_INT_ACTIVE interrupt is active	0x1
1	RTC_ALARM	NVIC_RTC_ALARM_INT_ACTIVE	The NVIC_RTC_ALARM_INT_ACTIVE interrupt is active	0x1
0	WAKEUP	NVIC_WAKEUP_INT_ACTIVE	The NVIC_WAKEUP_INT_ACTIVE interrupt is active	0x1

3.4.2.10 NVIC_IABR1

Bit Field	Read/Write	Field Name	Description
24	R	DMA3	Set the DMA3 interrupt as active
23	R	DMA2	Set the DMA2 interrupt as active
22	R	DMA1	Set the DMA1 interrupt as active
21	R	DMA0	Set the DMA0 interrupt as active
20	R	ACCESS_ERROR	Set the ACCESS_ERROR interrupt as active

Bit Field	Read/Write	Field Name	Description
19	R	FLASH0_ECC	Set the FLASH0_ECC interrupt as active
18	R	FLASH0_COPY	Set the FLASH0_COPY interrupt as active
17	R	RF_RXFIFO	Set the RF_RXFIFO interrupt as active
16	R	RF_TXFIFO	Set the RF_TXFIFO interrupt as active
15	R	RF_SYNC	Set the RF_SYNC interrupt as active
14	R	RF_IRQ_RECEIVED	Set the RF_IRQ_RECEIVED interrupt as active
13	R	RF_RXSTOP	Set the RF_RXSTOP interrupt as active
12	R	RF_TX	Set the RF_TX interrupt as active
11	R	TOF	Set the TOF interrupt as active
10	R	BLE_COEX_RX_TX	Set the BLE_COEX_RX_TX interrupt as active
9	R	BLE_COEX_IN_PROCESS	Set the BLE_COEX_IN_PROCESS interrupt as active
8	R	BLE_ERROR	Set the BLE_ERROR interrupt as active
7	R	BLE_FIFO	Set the BLE_FIFO interrupt as active
6	R	BLE_HSLOT	Set the BLE_HSLOT interrupt as active
5	R	BLE_SLP	Set the BLE_SLP interrupt as active
4	R	BLE_CRYPT	Set the BLE_CRYPT interrupt as active
3	R	BLE_TIMESTAMP_TGT2	Set the BLE_TIMESTAMP_TGT2 interrupt as active
2	R	BLE_TIMESTAMP_TGT1	Set the BLE_TIMESTAMP_TGT1 interrupt as active
1	R	BLE_FINETGT	Set the BLE_FINETGT interrupt as active
0	R	BLE_SW	Set the BLE_SW interrupt as active

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
24	DMA3	NVIC_DMA3_INT_ACTIVE	The NVIC_DMA3_INT_ACTIVE interrupt is active	0x1
23	DMA2	NVIC_DMA2_INT_ACTIVE	The NVIC_DMA2_INT_ACTIVE interrupt is active	0x1
22	DMA1	NVIC_DMA1_INT_ACTIVE	The NVIC_DMA1_INT_ACTIVE interrupt is active	0x1
21	DMA0	NVIC_DMA0_INT_ACTIVE	The NVIC_DMA0_INT_ACTIVE interrupt is active	0x1
20	ACCESS_ERROR	NVIC_ACCESS_ERROR_INT_ACTIVE	The NVIC_ACCESS_ERROR_INT_ACTIVE interrupt is active	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
19	FLASH0_ECC	NVIC_FLASH0_ECC_INT_ACTIVE	The NVIC_FLASH0_ECC_INT_ACTIVE interrupt is active	0x1
18	FLASH0_COPY	NVIC_FLASH0_COPY_INT_ ACTIVE	The NVIC_FLASH0_COPY_INT_ ACTIVE interrupt is active	0x1
17	RF_RXFIFO	NVIC_RF_RXFIFO_INT_ACTIVE	The NVIC_RF_RXFIFO_INT_ACTIVE interrupt is active	0x1
16	RF_TXFIFO	NVIC_RF_TXFIFO_INT_ACTIVE	The NVIC_RF_TXFIFO_INT_ACTIVE interrupt is active	0x1
15	RF_SYNC	NVIC_RF_SYNC_INT_ACTIVE	The NVIC_RF_SYNC_INT_ACTIVE interrupt is active	0x1
14	RF_IRQ_RECEIVED	NVIC_RF_IRQ_RECEIVED_INT_ ACTIVE	The NVIC_RF_IRQ_RECEIVED_INT_ ACTIVE interrupt is active	0x1
13	RF_RXSTOP	NVIC_RF_RXSTOP_INT_ACTIVE	The NVIC_RF_RXSTOP_INT_ACTIVE interrupt is active	0x1
12	RF_TX	NVIC_RF_TX_INT_ACTIVE	The NVIC_RF_TX_INT_ACTIVE interrupt is active	0x1
11	TOF	NVIC_TOF_INT_ACTIVE	The NVIC_TOF_INT_ACTIVE interrupt is active	0x1
10	BLE_COEX_RX_TX	NVIC_BLE_COEX_RX_TX_INT_ ACTIVE	The NVIC_BLE_COEX_RX_TX_INT_ ACTIVE interrupt is active	0x1
9	BLE_COEX_IN_ PROCESS	NVIC_BLE_COEX_IN_PROCESS_ INT_ACTIVE	The NVIC_BLE_COEX_IN_PROCESS_ INT_ACTIVE interrupt is active	0x1
8	BLE_ERROR	NVIC_BLE_ERROR_INT_ACTIVE	The NVIC_BLE_ERROR_INT_ACTIVE interrupt is active	0x1
7	BLE_FIFO	NVIC_BLE_FIFO_INT_ACTIVE	The NVIC_BLE_FIFO_INT_ACTIVE interrupt is active	0x1
6	BLE_HSLOT	NVIC_BLE_HSLOT_INT_ACTIVE	The NVIC_BLE_HSLOT_INT_ACTIVE interrupt is active	0x1
5	BLE_SLP	NVIC_BLE_SLP_INT_ACTIVE	The NVIC_BLE_SLP_INT_ACTIVE interrupt is active	0x1
4	BLE_CRYPT	NVIC_BLE_CRYPT_INT_ACTIVE	The NVIC_BLE_CRYPT_INT_ACTIVE interrupt is active	0x1
3	BLE_TIMESTAMP_TGT2	NVIC_BLE_TIMESTAMP_TGT2_ INT_ACTIVE	The NVIC_BLE_TIMESTAMP_TGT2_ INT_ACTIVE interrupt is active	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
2	BLE_TIMESTAMP_TGT1	NVIC_BLE_TIMESTAMP_TGT1_INT_ACTIVE	The NVIC_BLE_TIMESTAMP_TGT1_INT_ACTIVE interrupt is active	0x1
1	BLE_FINETGT	NVIC_BLE_FINETGT_INT_ACTIVE	The NVIC_BLE_FINETGT_INT_ACTIVE interrupt is active	0x1
0	BLE_SW	NVIC_BLE_SW_INT_ACTIVE	The NVIC_BLE_SW_INT_ACTIVE interrupt is active	0x1

3.4.2.11 NVIC_ITNS0

Bit Field	Read/Write	Field Name	Description
31	RW	ASCC_PHASE	Set the ASCC_PHASE interrupt as non-secure
30	RW	ASCC_PERIOD	Set the ASCC_PERIOD interrupt as non-secure
29	RW	CC312	Set the CC312 interrupt as non-secure
28	RW	FPU	Set the FPU interrupt as non-secure
27	RW	LIN0	Set the LIN0 interrupt as non-secure
26	RW	PCM0_ERROR	Set the PCM0_ERROR interrupt as non-secure
25	RW	PCM0_RX_TX	Set the PCM0_RX_TX interrupt as non-secure
24	RW	UART0_ERROR	Set the UART0_ERROR interrupt as non-secure
23	RW	UART0_TX	Set the UART0_TX interrupt as non-secure
22	RW	UART0_RX	Set the UART0_RX interrupt as non-secure
21	RW	I2C1	Set the I2C1 interrupt as non-secure
20	RW	I2C0	Set the I2C0 interrupt as non-secure
19	RW	SPI1_COM	Set the SPI1_COM interrupt as non-secure
18	RW	SPI1_TX	Set the SPI1_TX interrupt as non-secure
17	RW	SPI1_RX	Set the SPI1_RX interrupt as non-secure
16	RW	SPI0_COM	Set the SPI0_COM interrupt as non-secure
15	RW	SPI0_TX	Set the SPI0_TX interrupt as non-secure
14	RW	SPI0_RX	Set the SPI0_RX interrupt as non-secure
13	RW	WATCHDOG	Set the WATCHDOG interrupt as non-secure
12	RW	GPIO3	Set the GPIO3 interrupt as non-secure
11	RW	GPIO2	Set the GPIO2 interrupt as non-secure
10	RW	GPIO1	Set the GPIO1 interrupt as non-secure

Bit Field	Read/Write	Field Name	Description
9	RW	GPIO0	Set the GPIO0 interrupt as non-secure
8	RW	FIFO	Set the FIFO interrupt as non-secure
7	RW	TIMER3	Set the TIMER3 interrupt as non-secure
6	RW	TIMER2	Set the TIMER2 interrupt as non-secure
5	RW	TIMER1	Set the TIMER1 interrupt as non-secure
4	RW	TIMER0	Set the TIMER0 interrupt as non-secure
3	RW	LSAD_MONITOR	Set the LSAD_MONITOR interrupt as non-secure
2	RW	RTC_CLOCK	Set the RTC_CLOCK interrupt as non-secure
1	RW	RTC_ALARM	Set the RTC_ALARM interrupt as non-secure
0	RW	WAKEUP	Set the WAKEUP interrupt as non-secure

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31	ASCC_PHASE	NVIC_ASCC_PHASE_INT_NON_SECURE	The NVIC_ASCC_PHASE_INT_NON_SECURE interrupt is non-secure	0x1
30	ASCC_PERIOD	NVIC_ASCC_PERIOD_INT_NON_SECURE	The NVIC_ASCC_PERIOD_INT_NON_SECURE interrupt is non-secure	0x1
29	CC312	NVIC_CC312_INT_NON_SECURE	The NVIC_CC312_INT_NON_SECURE interrupt is non-secure	0x1
28	FPU	NVIC_FPU_INT_NON_SECURE	The NVIC_FPU_INT_NON_SECURE interrupt is non-secure	0x1
27	LIN0	NVIC_LIN0_INT_NON_SECURE	The NVIC_LIN0_INT_NON_SECURE interrupt is non-secure	0x1
26	PCM0_ERROR	NVIC_PCM0_ERROR_INT_NON_SECURE	The NVIC_PCM0_ERROR_INT_NON_SECURE interrupt is non-secure	0x1
25	PCM0_RX_TX	NVIC_PCM0_RX_TX_INT_NON_SECURE	The NVIC_PCM0_RX_TX_INT_NON_SECURE interrupt is non-secure	0x1
24	UART0_ERROR	NVIC_UART0_ERROR_INT_NON_SECURE	The NVIC_UART0_ERROR_INT_NON_SECURE interrupt is non-secure	0x1
23	UART0_TX	NVIC_UART0_TX_INT_NON_SECURE	The NVIC_UART0_TX_INT_NON_SECURE interrupt is non-secure	0x1
22	UART0_RX	NVIC_UART0_RX_INT_NON_SECURE	The NVIC_UART0_RX_INT_NON_SECURE interrupt is non-secure	0x1
21	I2C1	NVIC_I2C1_INT_NON_SECURE	The NVIC_I2C1_INT_NON_SECURE interrupt is non-secure	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
20	I2C0	NVIC_I2C0_INT_NON_SECURE	The NVIC_I2C0_INT_NON_SECURE interrupt is non-secure	0x1
19	SPI1_COM	NVIC_SPI1_COM_INT_NON_SECURE	The NVIC_SPI1_COM_INT_NON_SECURE interrupt is non-secure	0x1
18	SPI1_TX	NVIC_SPI1_TX_INT_NON_SECURE	The NVIC_SPI1_TX_INT_NON_SECURE interrupt is non-secure	0x1
17	SPI1_RX	NVIC_SPI1_RX_INT_NON_SECURE	The NVIC_SPI1_RX_INT_NON_SECURE interrupt is non-secure	0x1
16	SPI0_COM	NVIC_SPI0_COM_INT_NON_SECURE	The NVIC_SPI0_COM_INT_NON_SECURE interrupt is non-secure	0x1
15	SPI0_TX	NVIC_SPI0_TX_INT_NON_SECURE	The NVIC_SPI0_TX_INT_NON_SECURE interrupt is non-secure	0x1
14	SPI0_RX	NVIC_SPI0_RX_INT_NON_SECURE	The NVIC_SPI0_RX_INT_NON_SECURE interrupt is non-secure	0x1
13	WATCHDOG	NVIC_WATCHDOG_INT_NON_SECURE	The NVIC_WATCHDOG_INT_NON_SECURE interrupt is non-secure	0x1
12	GPIO3	NVIC_GPIO3_INT_NON_SECURE	The NVIC_GPIO3_INT_NON_SECURE interrupt is non-secure	0x1
11	GPIO2	NVIC_GPIO2_INT_NON_SECURE	The NVIC_GPIO2_INT_NON_SECURE interrupt is non-secure	0x1
10	GPIO1	NVIC_GPIO1_INT_NON_SECURE	The NVIC_GPIO1_INT_NON_SECURE interrupt is non-secure	0x1
9	GPIO0	NVIC_GPIO0_INT_NON_SECURE	The NVIC_GPIO0_INT_NON_SECURE interrupt is non-secure	0x1
8	FIFO	NVIC_FIFO_INT_NON_SECURE	The NVIC_FIFO_INT_NON_SECURE interrupt is non-secure	0x1
7	TIMER3	NVIC_TIMER3_INT_NON_SECURE	The NVIC_TIMER3_INT_NON_SECURE interrupt is non-secure	0x1
6	TIMER2	NVIC_TIMER2_INT_NON_SECURE	The NVIC_TIMER2_INT_NON_SECURE interrupt is non-secure	0x1
5	TIMER1	NVIC_TIMER1_INT_NON_SECURE	The NVIC_TIMER1_INT_NON_SECURE interrupt is non-secure	0x1
4	TIMER0	NVIC_TIMER0_INT_NON_SECURE	The NVIC_TIMER0_INT_NON_SECURE interrupt is non-secure	0x1
3	LSAD_MONITOR	NVIC_LSAD_MONITOR_INT_NON_SECURE	The NVIC_LSAD_MONITOR_INT_NON_SECURE interrupt is non-secure	0x1
2	RTC_CLOCK	NVIC_RTC_CLOCK_INT_NON_	The NVIC_RTC_CLOCK_INT_NON_	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		SECURE	SECURE interrupt is non-secure	
1	RTC_ALARM	NVIC_RTC_ALARM_INT_NON_SECURE	The NVIC_RTC_ALARM_INT_NON_SECURE interrupt is non-secure	0x1
0	WAKEUP	NVIC_WAKEUP_INT_NON_SECURE	The NVIC_WAKEUP_INT_NON_SECURE interrupt is non-secure	0x1

3.4.2.12 NVIC_ITNS1

Bit Field	Read/Write	Field Name	Description
24	RW	DMA3	Set the DMA3 interrupt as non-secure
23	RW	DMA2	Set the DMA2 interrupt as non-secure
22	RW	DMA1	Set the DMA1 interrupt as non-secure
21	RW	DMA0	Set the DMA0 interrupt as non-secure
20	RW	ACCESS_ERROR	Set the ACCESS_ERROR interrupt as non-secure
19	RW	FLASH0_ECC	Set the FLASH0_ECC interrupt as non-secure
18	RW	FLASH0_COPY	Set the FLASH0_COPY interrupt as non-secure
17	RW	RF_RXFIFO	Set the RF_RXFIFO interrupt as non-secure
16	RW	RF_TXFIFO	Set the RF_TXFIFO interrupt as non-secure
15	RW	RF_SYNC	Set the RF_SYNC interrupt as non-secure
14	RW	RF_IRQ_RECEIVED	Set the RF_IRQ_RECEIVED interrupt as non-secure
13	RW	RF_RXSTOP	Set the RF_RXSTOP interrupt as non-secure
12	RW	RF_TX	Set the RF_TX interrupt as non-secure
11	RW	TOF	Set the TOF interrupt as non-secure
10	RW	BLE_COEX_RX_TX	Set the BLE_COEX_RX_TX interrupt as non-secure
9	RW	BLE_COEX_IN_PROCESS	Set the BLE_COEX_IN_PROCESS interrupt as non-secure
8	RW	BLE_ERROR	Set the BLE_ERROR interrupt as non-secure
7	RW	BLE_FIFO	Set the BLE_FIFO interrupt as non-secure
6	RW	BLE_HSLOT	Set the BLE_HSLOT interrupt as non-secure
5	RW	BLE_SLP	Set the BLE_SLP interrupt as non-secure
4	RW	BLE_CRYPT	Set the BLE_CRYPT interrupt as non-secure
3	RW	BLE_TIMESTAMP_TGT2	Set the BLE_TIMESTAMP_TGT2 interrupt as non-secure

Bit Field	Read/Write	Field Name	Description
2	RW	BLE_TIMESTAMP_TGT1	Set the BLE_TIMESTAMP_TGT1 interrupt as non-secure
1	RW	BLE_FINETGT	Set the BLE_FINETGT interrupt as non-secure
0	RW	BLE_SW	Set the BLE_SW interrupt as non-secure

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
24	DMA3	NVIC_DMA3_INT_NON_SECURE	The NVIC_DMA3_INT_NON_SECURE interrupt is active	0x1
23	DMA2	NVIC_DMA2_INT_NON_SECURE	The NVIC_DMA2_INT_NON_SECURE interrupt is active	0x1
22	DMA1	NVIC_DMA1_INT_NON_SECURE	The NVIC_DMA1_INT_NON_SECURE interrupt is active	0x1
21	DMA0	NVIC_DMA0_INT_NON_SECURE	The NVIC_DMA0_INT_NON_SECURE interrupt is active	0x1
20	ACCESS_ERROR	NVIC_ACCESS_ERROR_INT_NON_SECURE	The NVIC_ACCESS_ERROR_INT_NON_SECURE interrupt is active	0x1
19	FLASH0_ECC	NVIC_FLASH0_ECC_INT_NON_SECURE	The NVIC_FLASH0_ECC_INT_NON_SECURE interrupt is active	0x1
18	FLASH0_COPY	NVIC_FLASH0_COPY_INT_NON_SECURE	The NVIC_FLASH0_COPY_INT_NON_SECURE interrupt is active	0x1
17	RF_RXFIFO	NVIC_RF_RXFIFO_INT_NON_SECURE	The NVIC_RF_RXFIFO_INT_NON_SECURE interrupt is active	0x1
16	RF_TXFIFO	NVIC_RF_TXFIFO_INT_NON_SECURE	The NVIC_RF_TXFIFO_INT_NON_SECURE interrupt is active	0x1
15	RF_SYNC	NVIC_RF_SYNC_INT_NON_SECURE	The NVIC_RF_SYNC_INT_NON_SECURE interrupt is active	0x1
14	RF_IRQ_RECEIVED	NVIC_RF_IRQ_RECEIVED_INT_NON_SECURE	The NVIC_RF_IRQ_RECEIVED_INT_NON_SECURE interrupt is active	0x1
13	RF_RXSTOP	NVIC_RF_RXSTOP_INT_NON_SECURE	The NVIC_RF_RXSTOP_INT_NON_SECURE interrupt is active	0x1
12	RF_TX	NVIC_RF_TX_INT_NON_SECURE	The NVIC_RF_TX_INT_NON_SECURE interrupt is active	0x1
11	TOF	NVIC_TOF_INT_NON_SECURE	The NVIC_TOF_INT_NON_SECURE interrupt is active	0x1
10	BLE_COEX_RX_TX	NVIC_BLE_COEX_RX_TX_INT_NON_SECURE	The NVIC_BLE_COEX_RX_TX_INT_NON_SECURE interrupt is active	0x1
9	BLE_COEX_IN_	NVIC_BLE_COEX_IN_PROCESS_	The NVIC_BLE_COEX_IN_PROCESS_	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
	PROCESS	INT_NON_SECURE	INT_NON_SECURE interrupt is active	
8	BLE_ERROR	NVIC_BLE_ERROR_INT_NON_SECURE	The NVIC_BLE_ERROR_INT_NON_SECURE interrupt is active	0x1
7	BLE_FIFO	NVIC_BLE_FIFO_INT_NON_SECURE	The NVIC_BLE_FIFO_INT_NON_SECURE interrupt is active	0x1
6	BLE_HSLOT	NVIC_BLE_HSLOT_INT_NON_SECURE	The NVIC_BLE_HSLOT_INT_NON_SECURE interrupt is active	0x1
5	BLE_SLP	NVIC_BLE_SLP_INT_NON_SECURE	The NVIC_BLE_SLP_INT_NON_SECURE interrupt is active	0x1
4	BLE_CRYPT	NVIC_BLE_CRYPT_INT_NON_SECURE	The NVIC_BLE_CRYPT_INT_NON_SECURE interrupt is active	0x1
3	BLE_TIMESTAMP_TGT2	NVIC_BLE_TIMESTAMP_TGT2_INT_NON_SECURE	The NVIC_BLE_TIMESTAMP_TGT2_INT_NON_SECURE interrupt is active	0x1
2	BLE_TIMESTAMP_TGT1	NVIC_BLE_TIMESTAMP_TGT1_INT_NON_SECURE	The NVIC_BLE_TIMESTAMP_TGT1_INT_NON_SECURE interrupt is active	0x1
1	BLE_FINETGT	NVIC_BLE_FINETGT_INT_NON_SECURE	The NVIC_BLE_FINETGT_INT_NON_SECURE interrupt is active	0x1
0	BLE_SW	NVIC_BLE_SW_INT_NON_SECURE	The NVIC_BLE_SW_INT_NON_SECURE interrupt is active	0x1

3.4.2.13 NVIC_IPR0

Bit Field	Read/Write	Field Name	Description
31:29	RW	LSAD_MONITOR	Configure the LSAD_MONITOR interrupt priority
23:21	RW	RTC_CLOCK	Configure the RTC_CLOCK interrupt priority
15:13	RW	RTC_ALARM	Configure the RTC_ALARM interrupt priority
7:5	RW	WAKEUP	Configure the WAKEUP interrupt priority

3.4.2.14 NVIC_IPR1

Bit Field	Read/Write	Field Name	Description
31:29	RW	TIMER3	Configure the TIMER3 interrupt priority
23:21	RW	TIMER2	Configure the TIMER2 interrupt priority
15:13	RW	TIMER1	Configure the TIMER1 interrupt priority
7:5	RW	TIMER0	Configure the TIMER0 interrupt priority

3.4.2.15 NVIC_IPR2

Bit Field	Read/Write	Field Name	Description
31:29	RW	GPIO2	Configure the GPIO2 interrupt priority
23:21	RW	GPIO1	Configure the GPIO1 interrupt priority
15:13	RW	GPIO0	Configure the GPIO0 interrupt priority
7:5	RW	FIFO	Configure the FIFO interrupt priority

3.4.2.16 NVIC_IPR3

Bit Field	Read/Write	Field Name	Description
31:29	RW	SPI0_TX	Configure the SPI0_TX interrupt priority
23:21	RW	SPI0_RX	Configure the SPI0_RX interrupt priority
15:13	RW	WATCHDOG	Configure the WATCHDOG interrupt priority
7:5	RW	GPIO3	Configure the GPIO3 interrupt priority

3.4.2.17 NVIC_IPR4

Bit Field	Read/Write	Field Name	Description
31:29	RW	SPI1_COM	Configure the SPI1_COM interrupt priority
23:21	RW	SPI1_TX	Configure the SPI1_TX interrupt priority
15:13	RW	SPI1_RX	Configure the SPI1_RX interrupt priority
7:5	RW	SPI0_COM	Configure the SPI0_COM interrupt priority

3.4.2.18 NVIC_IPR5

Bit Field	Read/Write	Field Name	Description
31:29	RW	UART0_TX	Configure the UART0_TX interrupt priority
23:21	RW	UART0_RX	Configure the UART0_RX interrupt priority
15:13	RW	I2C1	Configure the I2C1 interrupt priority
7:5	RW	I2C0	Configure the I2C0 interrupt priority

3.4.2.19 NVIC_IPR6

Bit Field	Read/Write	Field Name	Description
31:29	RW	LIN0	Configure the LIN0 interrupt priority
23:21	RW	PCM0_ERROR	Configure the PCM0_ERROR interrupt priority
15:13	RW	PCM0_RX_TX	Configure the PCM0_RX_TX interrupt priority
7:5	RW	UART0_ERROR	Configure the UART0_ERROR interrupt priority

3.4.2.20 NVIC_IPR7

Bit Field	Read/Write	Field Name	Description
31:29	RW	ASCC_PHASE	Configure the ASCC_PHASE interrupt priority
23:21	RW	ASCC_PERIOD	Configure the ASCC_PERIOD interrupt priority
15:13	RW	CC312	Configure the CC312 interrupt priority
7:5	RW	FPU	Configure the FPU interrupt priority

3.4.2.21 NVIC_IPR8

Bit Field	Read/Write	Field Name	Description
31:29	RW	BLE_TIMESTAMP_TGT2	Configure the BLE_TIMESTAMP_TGT2 interrupt priority
23:21	RW	BLE_TIMESTAMP_TGT1	Configure the BLE_TIMESTAMP_TGT1 interrupt priority
15:13	RW	BLE_FINETGT	Configure the BLE_FINETGT interrupt priority
7:5	RW	BLE_SW	Configure the BLE_SW interrupt priority

3.4.2.22 NVIC_IPR9

Bit Field	Read/Write	Field Name	Description
31:29	RW	BLE_FIFO	Configure the BLE_FIFO interrupt priority
23:21	RW	BLE_HSLOT	Configure the BLE_HSLOT interrupt priority
15:13	RW	BLE_SLP	Configure the BLE_SLP interrupt priority
7:5	RW	BLE_CRYPT	Configure the BLE_CRYPT interrupt priority

3.4.2.23 NVIC_IPR10

Bit Field	Read/Write	Field Name	Description
31:29	RW	TOF	Configure the TOF interrupt priority
23:21	RW	BLE_COEX_RX_TX	Configure the BLE_COEX_RX_TX interrupt priority
15:13	RW	BLE_COEX_IN_PROCESS	Configure the BLE_COEX_IN_PROCESS interrupt priority
7:5	RW	BLE_ERROR	Configure the BLE_ERROR interrupt priority

3.4.2.24 NVIC_IPR11

Bit Field	Read/Write	Field Name	Description
31:29	RW	RF_SYNC	Configure the RF_SYNC interrupt priority
23:21	RW	RF_IRQ_RECEIVED	Configure the RF_IRQ_RECEIVED interrupt priority
15:13	RW	RF_RXSTOP	Configure the RF_RXSTOP interrupt priority
7:5	RW	RF_TX	Configure the RF_TX interrupt priority

3.4.2.25 NVIC_IPR12

Bit Field	Read/Write	Field Name	Description
31:29	RW	FLASH0_ECC	Configure the FLASH0_ECC interrupt priority
23:21	RW	FLASH0_COPY	Configure the FLASH0_COPY interrupt priority
15:13	RW	RF_RXFIFO	Configure the RF_RXFIFO interrupt priority
7:5	RW	RF_TXFIFO	Configure the RF_TXFIFO interrupt priority

3.4.2.26 NVIC_IPR13

Bit Field	Read/Write	Field Name	Description
31:29	RW	ACCESS_ERROR	Configure the ACCESS_ERROR interrupt priority
23:21	RW	DMA0	Configure the DMA0 interrupt priority
15:13	RW	DMA1	Configure the DMA1 interrupt priority
7:5	RW	DMA2	Configure the DMA2 interrupt priority

3.4.2.27 NVIC_IPR14

Bit Field	Read/Write	Field Name	Description
7:5	RW	DMA3	Configure the DMA3 interrupt priority

3.4.2.28 NVIC_ISER0_NS

Bit Field	Read/Write	Field Name	Description

3.4.2.29 NVIC_ISER1_NS

Bit Field	Read/Write	Field Name	Description

3.4.2.30 NVIC_ICER0_NS

Bit Field	Read/Write	Field Name	Description

3.4.2.31 NVIC_ICER1_NS

Bit Field	Read/Write	Field Name	Description

3.4.2.32 NVIC_ISPR0_NS

Bit Field	Read/Write	Field Name	Description

3.4.2.33 NVIC_ISPR1_NS

Bit Field	Read/Write	Field Name	Description

3.4.2.34 NVIC_ICPR0_NS

Bit Field	Read/Write	Field Name	Description

3.4.2.35 NVIC_ICPR1_NS

Bit Field	Read/Write	Field Name	Description

3.4.2.36 NVIC_IABR0_NS

Bit Field	Read/Write	Field Name	Description

3.4.2.37 NVIC_IABR1_NS

Bit Field	Read/Write	Field Name	Description

3.4.2.38 NVIC_IPR0_NS

Bit Field	Read/Write	Field Name	Description

3.4.2.39 NVIC_ICPR1_NS

Bit Field	Read/Write	Field Name	Description

3.4.2.40 NVIC_IPR2_NS

Bit Field	Read/Write	Field Name	Description

3.4.2.41 NVIC_IPR3_NS

Bit Field	Read/Write	Field Name	Description

3.4.2.42 NVIC_IPR4_NS

Bit Field	Read/Write	Field Name	Description

3.4.2.43 NVIC_IPR5_NS

Bit Field	Read/Write	Field Name	Description

3.4.2.44 NVIC_IPR6_NS

Bit Field	Read/Write	Field Name	Description

3.4.2.45 NVIC_IPR7_NS

Bit Field	Read/Write	Field Name	Description

3.4.2.46 NVIC_IPR8_NS

Bit Field	Read/Write	Field Name	Description

3.4.2.47 NVIC_IPR9_NS

Bit Field	Read/Write	Field Name	Description

3.4.2.48 NVIC_IPR10_NS

Bit Field	Read/Write	Field Name	Description

3.4.2.49 NVIC_IPR11_NS

Bit Field	Read/Write	Field Name	Description

3.4.2.50 NVIC_IPR12_NS

Bit Field	Read/Write	Field Name	Description

3.4.2.51 NVIC_IPR13_NS

Bit Field	Read/Write	Field Name	Description

3.4.2.52 NVIC_IPR14_NS

Bit Field	Read/Write	Field Name	Description

3.4.3 Memory Protection Unit Registers

Register Name	Register Description	Address
MPU_TYPE	Indicates how many regions the MPU for the selected security state supports	0xE000ED90
MPU_CTRL	MPU control register	0xE000ED94
MPU_RNR	MPU Region Number Register	0xE000ED98
MPU_RBAR	MPU Region Base Address Register	0xE000ED9C
MPU_RLAR	MPU Region Limit Address Register	0xE000EDA0
MPU_RBAR1	MPU Region 1 Base Address Register	0xE000EDA4
MPU_RLAR1	MPU Region 1 Limit Address Register	0xE000EDA8
MPU_RBAR2	MPU Region 2 Base Address Register	0xE000EDAC
MPU_RLAR2	MPU Region 2 Limit Address Register	0xE000EDB0

Register Name	Register Description	Address
MPU_RBAR3	MPU Region 3 Base Address Register	0xE000EDB4
MPU_RLAR3	MPU Region 3 Limit Address Register	0xE000EDB8
MPU_MAIR0	MPU Memoryattribute Indirection Register 0	0xE000EDC0

3.4.3.1 MPU_TYPE

Bit Field	Read/Write	Field Name	Description
15:8	R	NBR_OF_REGION	Number of regions

3.4.3.2 MPU_CTRL

Bit Field	Read/Write	Field Name	Description
2	RW	PRIVDEFENA	Privileged default enable
1	RW	HFNMIENA	Hard fault and NMI enable
0	RW	ENABLE	Enable the MPU

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
2	PRIVDEFENA	DEFAULT_MEM_MAP_DISABLED	Disables use of the default memory map. Any memory access to a location that is not covered by any enabled region causes a fault	0x0*
		DEFAULT_MEM_MAP_ENABLED	Enables use of the default memory map as a background region for privileged software accesses	0x1
1	HFNMIENA	MPU_DISABLED_FOR_HF_NMI	MPU is disabled during HardFault and NMI handlers, regardless of the value of the ENABLE bit	0x0*
		MPU_ENABLED_FOR_HF_NMI	The MPU is enabled during HardFault and NMI handlers	0x1
0	ENABLE	MPU_DISABLED	MPU is disabled	0x0*
		MPU_ENABLED	MPU is enabled	0x1

3.4.3.3 MPU_RNR

Bit Field	Read/Write	Field Name	Description
7:0	R	REGION	Region number

3.4.3.4 MPU_RBAR

Bit Field	Read/Write	Field Name	Description
31:5	RW	BASE	Base address
4:3	RW	SH	Shareability
2:1	RW	AP	Access permissions
0	RW	XN	Execute never

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
4:3	SH	NON_SHAREABLE	Non-shareable	0x0*
		OUTER_SHAREABLE	Outer shareable	0x2
		INNER_SHAREABLE	Inner Shareable	0x3
2:1	AP	RW_PRIVILEGED_CODE_ONLY	Read/write by privileged code only	0x0*
		RW_PRIVILEGED_LEVEL	Read/write by any privilege level	0x1
		R_PRIVILEGED_CODE_ONLY	Read-only by privileged code only	0x2
		R_PRIVILEGED_LEVEL	Read-only by any privilege level	0x3
0	XN	EXEC_PERMITED	Execution only permitted if read permitted	0x0*
		EXEC_NOT_PERMITTED	Execution not permitted	0x1

3.4.3.5 MPU_RLAR

Bit Field	Read/Write	Field Name	Description
31:5	RW	LIMIT	Limit address
3:1	RW	ATTRINDX	Attribute index
0	RW	EN	Region enable

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
0	EN	REGION_DISABLED	Region disabled	0x0*
		REGION_ENABLED	Region enabled	0x1

3.4.3.6 MPU_RBAR1

Bit Field	Read/Write	Field Name	Description
31:5	RW	BASE	Base address
4:3	RW	SH	Shareability
2:1	RW	AP	Access permissions
0	RW	XN	Execute never

3.4.3.7 MPU_RLAR1

Bit Field	Read/Write	Field Name	Description
31:5	RW	LIMIT	Limit address
3:1	RW	ATTRINDX	Attribute index
0	RW	EN	Region enable

3.4.3.8 MPU_RBAR2

Bit Field	Read/Write	Field Name	Description
31:5	RW	BASE	Base address
4:3	RW	SH	Shareability
2:1	RW	AP	Access permissions
0	RW	XN	Execute never

3.4.3.9 MPU_RLAR2

Bit Field	Read/Write	Field Name	Description
31:5	RW	LIMIT	Limit address
3:1	RW	ATTRINDX	Attribute index
0	RW	EN	Region enable

3.4.3.10 MPU_RBAR3

Bit Field	Read/Write	Field Name	Description
31:5	RW	BASE	Base address
4:3	RW	SH	Shareability
2:1	RW	AP	Access permissions
0	RW	XN	Execute never

3.4.3.11 MPU_RLAR3

Bit Field	Read/Write	Field Name	Description
31:5	RW	LIMIT	Limit address
3:1	RW	ATTRINDX	Attribute index
0	RW	EN	Region enable

3.4.3.12 MPU_MAIR0

Bit Field	Read/Write	Field Name	Description
31:24	RW	REGION3	Memory attribute encoding for MPU regions 3
23:16	RW	REGION2	Memory attribute encoding for MPU regions 2
15:8	RW	REGION1	Memory attribute encoding for MPU regions 1
7:0	RW	REGION0	Memory attribute encoding for MPU regions 0

3.4.4 Security Attribution Unit Registers

Register Name	Register Description	Address
SAU_CTRL	SAU Control Register	0xE000EDD0
SAU_TYPE	SAU Type Register	0xE000EDD4
SAU_RNR	SAU Region Number Register	0xE000EDD8
SAU_RBAR	SAU Region Base Address Register	0xE000EDDC
SAU_RLAR	SAU Region Limit Address Register	0xE000EDE0
SAU_SFSR	Secure Fault Status Register	0xE000EDE4
SAU_SFAR	Secure Fault Address Register	0xE000EDE8

3.4.4.1 SAU_CTRL

Bit Field	Read/Write	Field Name	Description
1	RW	ALLNS	All non-secure
0	RW	ENABLE	Enable the SAU

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
1	ALLNS	MEM_AS_SECURE	Memory is marked as Secure and is not Non-secure callable	0x0*
		MEM_AS_NON_SECURE	Memory is marked as Non-secure	0x1
0	ENABLE	SAU_DISABLED	The SAU is disabled	0x0*
		SAU_ENABLED	The SAU is enabled	0x1

3.4.4.2 SAU_TYPE

Bit Field	Read/Write	Field Name	Description
7:0	R	NBR_OF_REGION	Number of region implemented by the SAU

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
7:0	NBR_OF_REGION	FOUR_SAU_REGIONS	The number of implemented SAU regions	0x4*

3.4.4.3 SAU_RNR

Bit Field	Read/Write	Field Name	Description
7:0	RW	SEL_REGION	Selects the region currently accessed by SAU_RBAR and SAU_RLAR

3.4.4.4 SAU_RBAR

Bit Field	Read/Write	Field Name	Description
31:5	RW	BASE_ADDRESS	Base address

3.4.4.5 SAU_RLAR

Bit Field	Read/Write	Field Name	Description
31:5	RW	LIMIT_ADDRESS	Limit address

3.4.4.6 SAU_SFSR

Bit Field	Read/Write	Field Name	Description
7	R	LSERR	Lazy state error flag
6	R	SFARVALID	Secure fault address valid
5	R	LSPERR	Lazy state preservation error flag
4	R	INVTRAN	Invalid transition flag
3	R	AUVIOL	Attribution unit violation flag
2	R	INVER	Invalid exception return flag
1	R	INVIS	Invalid integrity signature flag
0	R	INVEP	Invalid entry point

3.4.4.7 SAU_SFAR

Bit Field	Read/Write	Field Name	Description
31:0	R	ADDRESS	Address

3.4.5 Debug Control Block Registers

Register Name	Register Description	Address
DEBUG_DHCSR	Debug Halting Control and Status Register	0xE000EDF0
DEBUG_DCRSR	Debug Core Register Selector Register	0xE000EDF4
DEBUG_DCRDR	Debug Core Register Data Register	0xE000EDF8
DEBUG_DEMCR	Debug Exception and Monitor Control Register	0xE000EDFC
DEBUG_DAUTHCTRL	Debug Authentication Control Regiser	0xE000EE04
DEBUG_DSCSR	Debug Security Control and Status Register	0xE000EE08

3.4.5.1 DEBUG_DHCSR

Bit Field	Read/Write	Field Name	Description
31:16	W	DBGKEY	Debug key must be written to this field in order to write the rest of the register
26	R	S_RESART_S	Restart sticky status
25	R	S_RESET_ST	Core has been reset or is being rest. Bit is cleared on read.
24	R	S_RETIRE_ST	Retire sticky status
20	R	S_SDE	Secure debug enable. Indicates whether Secure invasive debug is allowed
19	R	S_LOCKUP	Indicates if core is in lockup state
18	R	S_SLEEP	Indicates if core is in sleep mode
17	R	S_HALT	Indicates is core is halted
16	R	S_REGRDY	Indicates register read/write operation is completed
5	RW	C_SNAPSTALL	Set to break a stalled memory access
3	RW	C_MASKINTS	Mask interrupts while stepping
2	RW	C_STEP	Single step the processor
1	RW	C_HALT	Halt the processor
0	RW	C_DEBUGEN	Enable halt mode debugging

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:16	DBGKEY	DEBUG_HALT_KEY	Key to unlock DEBUG_DHCSR register	0xA05F

3.4.5.2 DEBUG_DCRSR

Bit Field	Read/Write	Field Name	Description
16	W	REGWNR	Indicates direction of register transfer
6:0	W	REGSEL	Indicates register to be accessed

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
16	REGWNR	REGWNR_READ	Indicates register read	0x0
		REGWNR_WRITE	Indicates register write	0x1
6:0	REGSEL	REGSEL_R0	Select R0	0x0
		REGSEL_R1	Select R1	0x1
		REGSEL_R2	Select R2	0x2
		REGSEL_R3	Select R3	0x3
		REGSEL_R4	Select R4	0x4
		REGSEL_R5	Select R5	0x5
		REGSEL_R6	Select R6	0x6
		REGSEL_R7	Select R7	0x7
		REGSEL_R8	Select R8	0x8
		REGSEL_R9	Select R9	0x9
		REGSEL_R10	Select R10	0xA
		REGSEL_R11	Select R11	0xB
		REGSEL_R12	Select R12	0xC
		REGSEL_SP	Select current stack pointer	0xD
		REGSEL_LR	Select LR	0xE
		REGSEL_DRA	Select Debug Return Address	0xF
		REGSEL_XPSR	Select xPSR/flags	0x10
		REGSEL_MSP	Select main stack pointer	0x11
		REGSEL_PSP	Select process stack pointer	0x12
		REGSEL_SPECREG	Access other registers including control, FAULTMASK, BASEPRI and PRIMASK	0x14
		REGSEL_MSP_NS	Select non-secure main stack pointer	0x18
		REGSEL_PSP_NS	Select non-secure process stack pointer	0x19
		REGSEL_MPS_S	Select secure main stack pointer	0x1A

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		REGSEL_PSP_S	Select secure process stack pointer	0x1B
		REGSEL_MSPLIM_S	Select secure main stack limit	0x1C
		REGSEL_PSPLIM_S	Select secure process stack limit	0x1D
		REGSEL_MSPLIM_NS	Select non-secure main stack limit	0x1E
		REGSEL_PSPLIM_NS	Select non-secure process stack limit	0x1F
		REGSEL_FPSCR	Select Floating Point register	0x21
		REGSEL_SPECREG_S	Access other registers including control, FAULTMASK, BASEPRI and PRIMASK when S_SDE = 1	0x22
		REGSEL_S0	Select Floating point register S0	0x40
		REGSEL_S1	Select Floating point register S1	0x41
		REGSEL_S2	Select Floating point register S2	0x42
		REGSEL_S3	Select Floating point register S3	0x43
		REGSEL_S4	Select Floating point register S4	0x44
		REGSEL_S5	Select Floating point register S5	0x45
		REGSEL_S6	Select Floating point register S6	0x46
		REGSEL_S7	Select Floating point register S7	0x47
		REGSEL_S8	Select Floating point register S8	0x48
		REGSEL_S9	Select Floating point register S9	0x49
		REGSEL_S10	Select Floating point register S10	0x4A
		REGSEL_S11	Select Floating point register S11	0x4B
		REGSEL_S12	Select Floating point register S12	0x4C
		REGSEL_S13	Select Floating point register S13	0x4D
		REGSEL_S14	Select Floating point register S14	0x4E
		REGSEL_S15	Select Floating point register S15	0x4F
		REGSEL_S16	Select Floating point register S16	0x50
		REGSEL_S17	Select Floating point register S17	0x51
		REGSEL_S18	Select Floating point register S18	0x52
		REGSEL_S19	Select Floating point register S19	0x53
		REGSEL_S20	Select Floating point register S20	0x54
		REGSEL_S21	Select Floating point register S21	0x55

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		REGSEL_S22	Select Floating point register S22	0x56
		REGSEL_S23	Select Floating point register S23	0x57
		REGSEL_S24	Select Floating point register S24	0x58
		REGSEL_S25	Select Floating point register S25	0x59
		REGSEL_S26	Select Floating point register S26	0x5A
		REGSEL_S27	Select Floating point register S27	0x5B
		REGSEL_S28	Select Floating point register S28	0x5C
		REGSEL_S29	Select Floating point register S29	0x5D
		REGSEL_S30	Select Floating point register S30	0x5E
		REGSEL_S31	Select Floating point register S31	0x5F

3.4.5.3 DEBUG_DCRDR

Bit Field	Read/Write	Field Name	Description
31:0	RW	DEBUG_REGDATA	Register read/write data for debugging

3.4.5.4 DEBUG_DEMCR

Bit Field	Read/Write	Field Name	Description

3.4.5.5 DEBUG_DAUTHCTRL

Bit Field	Read/Write	Field Name	Description
3	R	INTSPNIDEN	Internal Secure non-invasive debug enable
2	R	SPNIDENSEL	Secure non-invasive debug enable select
1	R	INTSPIDEN	Internal Secure invasive debug enable
0	R	SPIDENSEL	Secure invasive debug enable select

3.4.5.6 DEBUG_DSCSR

Bit Field	Read/Write	Field Name	Description
17	W	CDSKEY	CDS write-enable key
16	RW	CDS	Current domain secure
1	RW	SBRSEL	Secure banked register select
0	RW	SBRSELEN	Secure banked register select enable

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
17	CDSKEY	CDS_NOT_IGNORED	Concurrent write to CDS not ignored	0x0
		CDS_IGNORED	Concurrent write to CDS ignored	0x1
16	CDS	PE_IS_IN_NON_SECURE	PE is in Non-secure state	0x0*
		PE_IS_IN_SECURE	PE is in Secure state	0x1

3.4.6 Software Interrupts Registers

Register Name	Register Description	Address
SIG_STIR	Software Triggered Interrupt Register	0xE000EF00

3.4.6.1 SIG_STIR

Bit Field	Read/Write	Field Name	Description
8:0	W	INTID	Indicates the interrupt to be pended

3.4.7 Floating Point Extension Registers

Register Name	Register Description	Address
FPE_FPCCR	Floating-Point Context Control Register	0xE000EF34
FPE_FPCAR	Floating-Point Context Address Register	0xE000EF38
FPE_FPDSCR	Floating-Point Default Status Control Register	0xE000EF3C
FPE_MVFR0	Media and VFP Feature Register 0	0xE000EF40
FPE_MVFR1	Media and VFP Feature Register 1	0xE000EF44
FPE_MVFR2	Media and VFP Feature Register 2	0xE000EF48

3.4.7.1 FPE_FPCCR

Bit Field	Read/Write	Field Name	Description
31	RW	ASPEN	When this bit is set to 1, execution of a floating-point instruction sets the CONTROL.FPCA bit to 1
30	RW	LSPEN	Enable lazy contect save of FP state
29	RW	LSPENS	This bit controls whether the LSPEN bit is writable from the Non-secure state
28	RW	CLRONRET	Clear floating point caller saved register on exception return
27	RW	CLRONRETS	CLRONRET secure only
26	RW	TS	Threat FP registers as Secure enable

Bit Field	Read/Write	Field Name	Description
10	RW	UFRDY	Indicates whether the software executing when the processor allocated the floating-point stack frame was able to set the Usage Fault exeption to pending.
9	RW	SPLIMVIOL	This bit is banked between the Security states and indicates whether the floating-point context violates the stack pointer limit that was active when lazy state preservation was activated.
8	RW	MONRDY	Indicates whether the software executing when the processor allocated the floating-point stack frame was able to set the debug monitor execption to pending.
7	RW	SFRDY	Indicates whether the software executing when the processor allocated the floating-point stack frame was able to set the secure fault exception to pending.
6	RW	BRDY	Indicates whether the software executing when the processor allocated the floating-point stack frame was able to set the bus fault exception to pending.
5	RW	MMRDY	Indicates whether the software executing when the processor allocated the floating-point stack frame was able to set the Mem Manage exception to pending.
4	RW	HFRDY	Indicates whether the software executing when the processor allocated the floating-point stack frame was able to set the Hard fault exception to pending.
3	RW	THREAD	Indicates the processor mode when it allocated the floating-point stack frame.
2	RW	S	Security status of the floating-point context
1	RW	USER	Indicates the privilege level of the software executing when the processor allocated the floating-point stack frame
0	RW	LSPACT	Indicates whether lazy preservation of the floating-point state is active.

3.4.7.2 FPE_FPCAR

Bit Field	Read/Write	Field Name	Description
31:3	RW	ADDRESS	The location of the unpopulated floating-point register space allocated on an exception stack frame.

3.4.7.3 FPE_FPDSCR

Bit Field	Read/Write	Field Name	Description
26	RW	AHP	Alternative half-precision.
25	RW	DN	Default NaN
24	RW	FZ	Flush-to-zero
23:22	RW	RMODE	Rounding mode

3.4.7.4 FPE_MVFR0

Bit Field	Read/Write	Field Name	Description
31:28	RW	FPROUND	All rounding modes supported
23:20	RW	FPSQRT	Floating-point square root
19:16	RW	FPDIVIDE	Floating-point divide
11:8	RW	FPDP	Floating-point double-precision
7:4	RW	FPSP	Floating-point single-precision
3:0	RW	SIMDREG	SIMD registers. Indicates size of floating-point extension register file. (15*64 bit registers)

3.4.7.5 FPE_MVFR1

Bit Field	Read/Write	Field Name	Description
31:28	RW	FMAC	Fused multiply accumulate
27:24	RW	FPHP	Floating-point half-precision
7:4	RW	FPDNAN	Floating-point default NaN
3:0	RW	FPFTZ	Floating-point flush-to-zero

3.4.7.6 FPE_MVFR2

Bit Field	Read/Write	Field Name	Description
7:4	RW	FPMISC	Floating-point miscellaneous

CHAPTER 4

Arm TrustZone CryptoCell-312 Security IP

The RSL15 contains an Arm CryptoCell-312, which supports the following functional features:

- Symmetric and asymmetric cryptography
- True Random Number Generation (TRNG)
- Device lifecycle state management
- Secure boot including Root of Trust (RoT), providing software image validation and optional decryption at both boot time and update time
- Provisioning, management and isolation for keys and assets
- Provides secure debug and test facilities

These supported elements are briefly described here, and additional material can be found in the following manuals:

- *RSL15 Firmware Reference Manual*
- *RSL15 Security User's Guide*
- *RSL15 Developer's Guide*
- *Arm TrustZone CryptoCell-312 Software Developers Manual*

4.1 Cryptographic Features

User available cryptographic services that are provided by the Arm CryptoCell-312 include the following:

- Encryption and decryption schemes, including:
  - Advanced Encryption Standard (AES-128, AES-192, AES-256)
  - RSA (RSA-2048, RSA-3072, RSA-4096)
  - ChaCha (ChaCha20, ChaCha20-Poly1305)
  - Elliptic Curve Integrated Encryption Scheme (ECIES)
  - Galois Counter Mode (GCM) support, typically using AES
- Hash schemes, including:
  - Secure Hash Algorithms (SHA-1, SHA-224, SHA-256, SHA-384, SHA-512)
- Message Authentication Coding, including:
  - Hash Message Authentication Code (HMAC), typically using SHA-256
  - Cipher-based Message Authentication Code (CMAC), typically using AES
  - Elliptic Curve Digital Signature Algorithm (ECDSA)
  - Secure Remote Password (SRP)
- Key generation and exchange algorithms, including:
  - Elliptic-Curve Diffie-Hellman (ECDH)
  - DHM key exchange (DHM)
- CCM (CRT CBC-Mac, or Counter mode cipher block chaining message authentication codes calculated using a counter-based initialization vector)
- CTR-DRBG (Counter mode deterministic random bit/byte generator)
- Key Derivation

These services allow for development of custom proprietary security solutions.

4.2 True Random Number Generator (TRNG)

The Arm CryptoCell-312 implementation includes a hardware True Random Number Generator that conforms to the following standards:

- NIST SP800-90B
- NIST SP800-22
- NIST SP800-90C
- NIST SP800-90A
- FIPS 14-2
- BSI AIS-31

4.3 Power Configuration

The SYSCTRL_CRYPTOCELL_PWR_CFG register is responsible for configuring power to the Arm CryptoCell-312 module.

IMPORTANT: The SYSCTRL_CRYPTOCELL_PWR_CFG_PWR_KEY bit field in the SYSCTRL_CRYPTOCELL_PWR_CFG register must be written with the correct key at the same time that the desired operation is written to the register.

The SYSCTRL_CRYPTOCELL_PWR_CFG_CC_POWER_STARTUP and SYSCTRL_CRYPTOCELL_PWR_CFG_CC_POWER_ENABLE bit fields, in the same register, control the enabling and disabling of the power to the Arm CryptoCell-312 module. The module can be powered down when not in use, to save power in Low Power Modes, particularly in the lowest-powered configuration of Sleep Mode.

Powering down the Arm CryptoCell-312 module during Run Mode results in a restart of the device. At this point the debug port does not function but the device otherwise operates as normal. The CryptoCell Always-On feature (CCAO) is not reset in this case, so the Arm CryptoCell-312 block remains powered down, as the power bits are stuck as inactive. Only a power-on-reset can reset the CCAO to its default enabled state, allowing the user to connect to the debug port.

The following procedure is used to power up the Arm CryptoCell-312 module:

```
 ACS->PWR_CTRL = ((ACS->PWR_CTRL) & ~(1 << ACS_PWR_CTRL_CCAO_PWR_EN_Pos))
            | ACS_PWR_KEY | ACS_CCAO_POWERED;
 ACS->PWR_CTRL = ((ACS->PWR_CTRL) & ~(1 << ACS_PWR_CTRL_CCAO_ISOLATE_Pos))
            | ACS_PWR_KEY | ACS_CCAO_NOT_ISOLATE;

 /* Note that there is a 1.3usec delay automatically added when writing these registers
*/

 SYSCTRL->CRYPTOCELL_PWR_CFG = CC_WRITE_KEY | CC_POWER_STARTUP | CC_POWER_DISABLE | CC_
ISOLATE;
 SYSCTRL->CRYPTOCELL_PWR_CFG = CC_WRITE_KEY | CC_POWER_STARTUP | CC_POWER_ENABLE | CC_
ISOLATE;
 SYSCTRL->CRYPTOCELL_PWR_CFG = CC_WRITE_KEY | CC_POWER_STARTUP | CC_POWER_ENABLE | CC_
NOT_ISOLATE;
```

The following procedure is used to power down the Arm CryptoCell-312 module:

```
 SYSCTRL->CRYPTOCELL_PWR_CFG = CC_WRITE_KEY | CC_POWER_STARTUP | CC_POWER_ENABLE | CC_
ISOLATE;
 SYSCTRL->CRYPTOCELL_PWR_CFG = CC_WRITE_KEY | CC_POWER_STARTUP | CC_POWER_DISABLE | CC_
ISOLATE;
 SYSCTRL->CRYPTOCELL_PWR_CFG = CC_WRITE_KEY | CC_POWER_OFF | CC_POWER_DISABLE | CC_
ISOLATE;
```

```
ACS->PWR_CTRL = ((ACS->PWR_CTRL) & ~(1 << ACS_PWR_CTRL_CCAO_ISOLATE_Pos))
                | ACS_PWR_KEY | ACS_CCAO_ISOLATE;
ACS->PWR_CTRL = ((ACS->PWR_CTRL) & ~(1 << ACS_PWR_CTRL_CCAO_PWR_EN_Pos))
                | ACS_PWR_KEY | ACS_CCAO_SHUTDOWN;
```

4.4 TrustZone

TrustZone is a security hardware extension to the ARMv8-M architecture that has been included with the Arm Cortex-M33 processor in the RSL15 device. This extension provides hardware isolation between secure and non-secure components, and can separate safe applications from unsafe ones, protecting underlying system components from being accessed or changed by untrusted code. For more information about Arm TrustZone, refer to the *TrustZone technology for the ARMv8-M architecture* manual from Arm.

IMPORTANT: Developers creating TrustZone applications need to be aware of best practices for security when using TrustZone, including Stack Sealing and other recommendations from Arm.

4.4.1 Secure Execution

System execution starts in the secure execution state. Secure execution has access to all components of the RSL15 system, including the banked instances of several system control, GPIO, and private peripheral registers used for non-secure execution at an offset of 0x20000 from their secure forms.

System configuration, including configuration of the non-secure execution environment, must be implemented in secure execution. Items that the can only be configured in a secure application include:

- All system control elements controlled by registers in the SYSCTRL group, except the activity counters controlled by the SYSCTRL_CNT_CTRL register and their values stored to the SYSCTRL_*_CNT registers
- GPIO configuration using the GPIO_CFG registers (the configuration of each GPIO can be read by non-secure code), and configuration of the GPIO pads for JTAG mode using the GPIO_JTAG_SW_PAD_CFG register as described in Section 10.2 “Functional Configuration” on page 512
- GPIO interrupt debounce filter configuration (GPIO_INT_DEBOUNCE) and the GPIO interrupt summary (GPIO_INT_STATUS_S)

Most other system components can be configured to allow or deny access in non-secure code. Secure application configuration of the components that non-secure firmware may access includes:

- The RAM memories using the SYSCTRL_NS_ACCESS_RAM_CFG* registers
- Interfaces and peripherals that may be accessed by non-secure firmware, using the SYSCTRL_NS_PERIPH_CFG* registers
- The GPIOs that can be accessed by non-secure firmware, using the GPIO_CFG_NS_ACCESS_GPIO bits from the GPIO_CFG registers
- The GPIO interrupt sources that can be accessed by non-secure firmware, using the GPIO_INT_CFG_NS_ACCESS bits from the GPIO_CFG registers
- The interrupts that can be accessed by the non-secure firmware, using the NVIC_ITNS* registers

NOTE: The default configuration for all of these components denies access in non-secure mode.

4.4.2 Non-Secure Execution

Care must be taken in a non-secure application to ensure that no elements that remain secure are accessed in the non-secure application, as the non-secure execution has limited access to many system components as configured by the secure application. Any attempts to access a component that remains secure results in control reverting to the secure application's secure exception handler. For more information about secure exceptions, refer to Section 3.3.2 “Nested Vector Interrupt Controller (NVIC)” on page 58 and the *Arm Cortex-M33 Processor Technical Reference Manual.*

The non-secure portion of an application has separate:

- Register header files (use *rsl15_hw_cid*_ns.h*, *rsl15_hw_flat_cid*_ns.h*)
  - These register files contain the updated mappings for the GPIO registers and remove the protected SYSCTRL registers.
- Vector table defining the non-secure application's own interrupt handlers

4.4.3 Non-Secure Use of Secured Features

Secure and non-secure firmware applications can be built to work together. However, non-secure application components cannot access secure resources directly. Instead, any access to secure resources can go through APIs provided by secure software, and these APIs can implement authentications to decide if the access to the secured components is permitted. By having this arrangement, even if there are vulnerabilities in the non-secure applications, the secure components of the device are not compromised.

To create a function that is accessible to the non-secure components of your application, assign the `cmse_nonsecure_entry` attribute to your function, as shown in the following example:

```
int __attribute__ ((cmse_nonsecure_entry)) NonSecureCallback_ExampleFunction(int x)
```

4.5 Arm TrustZone CryptoCell-312 Registers

Register Name	Register Description	Address
SYSCTRL_CRYPTOCELL_PWR_CFG	CryptoCell Power Configuration	0x40000040
SYSCTRL_NS_ACCESS_PERIPH_CFG0	Non-Secure code access peripherals configuration	0x40000044
SYSCTRL_NS_ACCESS_PERIPH_CFG1	Non-Secure code access peripherals configuration	0x40000048
SYSCTRL_NS_ACCESS_RAM_CFG0	Non-Secure code access RAM configuration	0x4000004C
SYSCTRL_NS_ACCESS_RAM_CFG1	Non-Secure code access RAM configuration	0x40000050
SYSCTRL_DEU_STATUS	Data Exchange Unit Status	0x4000005C
SYSCTRL_DEU_DATA	Data Exchange Unit Data	0x40000060
SYSCTRL_PROD_STATUS	Production Test Status	0x40000064
SYSCTRL_CC_DCU_EN0	CryptoCell Always On block ICV owned DCU_EN [31:0] state	0x40000094
SYSCTRL_CC_DCU_EN1	CryptoCell Always On block OEM owned DCU_EN [63:32] state	0x40000098
SYSCTRL_CC_DCU_EN2	CryptoCell Always On block ICV owned DCU_EN [95:64] state	0x4000009C
SYSCTRL_CC_DCU_EN3	CryptoCell Always On block OEM owned DCU_EN [127:96]	0x400000A0

Register Name	Register Description	Address
	state	
SYSCTRL_CC_DCU_LOCK0	CryptoCell Always On block ICV owned DCU_LOCK [31:0] state	0x400000A4
SYSCTRL_CC_DCU_LOCK1	CryptoCell Always On block OEM owned DCU_LOCK [63:32] state	0x400000A8
SYSCTRL_CC_DCU_LOCK2	CryptoCell Always On block ICV owned DCU_LOCK [95:64] state	0x400000AC
SYSCTRL_CC_DCU_LOCK3	CryptoCell Always On block OEM owned DCU_LOCK [127:96] state	0x400000B0
SYSCTRL_CC_STATUS	CryptoCell Always On block various status fields	0x400000B4
SYSCTRL_CC_FEATURES_CTRL	CryptoCell always on block debug features control	0x400000B8

4.5.0.1 SYSCTRL_CRYPTOCELL_PWR_CFG

Bit Field	Read/Write	Field Name	Description
31:16	W	PWR_KEY	Key to enable write to this register
4	R	CC_CG_STATE	CryptoCell central clock gating state
3	RW	CC_POWER_STARTUP	Power control for CryptoCell
2	RW	CC_POWER_ENABLE	Power control for CryptoCell
1	RW	CC_ISOLATE_N	CryptoCell isolate control signal
0	R	CC_POWERDOWN_RDY	Indicates that CryptoCell can be powered down

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:16	PWR_KEY	CC_WRITE_KEY	Key to enable write to this register	0x4363
4	CC_CG_STATE	CC_CG_IS_IDLE	The CryptoCell is not receiving any clock	0x0*
		CC_CG_IS_ACTIVE	The CryptoCell is receiving clock	0x1
3	CC_POWER_STARTUP	CC_POWER_OFF	CryptoCell shut-down	0x0*
		CC_POWER_STARTUP	CryptoCell power startup control	0x1
2	CC_POWER_ENABLE	CC_POWER_DISABLE	CryptoCell power disable	0x0*
		CC_POWER_ENABLE	CryptoCell power enable	0x1
1	CC_ISOLATE_N	CC_ISOLATE	CryptoCell isolated	0x0*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		CC_NOT_ISOLATE	CryptoCell accessible	0x1
0	CC_POWERDOWN_RDY	CC_PWR_DOWN_NOT_RDY	CryptoCell could not be shut-down	0x0*
		CC_PWR_DOWN_RDY	CryptoCell can be shut-down	0x1

4.5.0.2 SYSCTRL_NS_ACCESS_PERIPH_CFG0

Bit Field	Read/Write	Field Name	Description
28	RW	PWM_ACCESS	Allow Non-Secure code to access the PWM
27	RW	LIN_ACCESS	Allow Non-Secure code to access the LIN[0:0]
26	RW	TOF_ACCESS	Allow Non-Secure code to access the TOF
25	RW	PCM_ACCESS	Allow Non-Secure code to access the PCM[0:0]
24	RW	UART_ACCESS	Allow Non-Secure code to access the UART[0:0]
23:22	RW	I2C_ACCESS	Allow Non-Secure code to access the I2C[1:0]
21:20	RW	SPI_ACCESS	Allow Non-Secure code to access the SPI[1:0]
19	RW	GPIO_SRC_ACCESS	Allow Non-Secure code to access the GPIO_SRC
18	RW	GPIO_ACCESS	Allow Non-Secure code to access the GPIO
17	RW	CC312_ACCESS	Allow Non-Secure code to access the CC312
16	RW	ASCC_ACCESS	Allow Non-Secure code to access the ASCC
15:12	RW	TIMER_ACCESS	Allow Non-Secure code to access the Timer[3:0]
10	RW	CRC_ACCESS	Allow Non-Secure code to access the CRC
8	RW	NMI_ACCESS	Allow Non-Secure code to access the NMI
7	RW	WATCHDOG_ACCESS	Allow Non-Secure code to access the Watchdog
6	RW	SYSCTRL_ACCESS	Allow Non-Secure code to access the SYSCTRL
5	RW	SENSOR_ACCESS	Allow Non-Secure code to access the Sensor
4	RW	ACS_ACCESS	Allow Non-Secure code to access the ACS
3	RW	LSAD_ACCESS	Allow Non-Secure code to access the LSAD config
2	RW	TEST_CTRL_ACCESS	Allow Non-Secure code to access the TEST_CTRL config
1	RW	CLK_ACCESS	Allow Non-Secure code to access the CLK config
0	RW	RESET_ACCESS	Allow Non-Secure code to access the Reset

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
28	PWM_ACCESS	NS_CANNOT_ACCESS_PWM	Non-Secure code cannot access the PWM	0x0*
		NS_CAN_ACCESS_PWM	Non-Secure code can access the PWM	0x1
27	LIN_ACCESS	NS_CANNOT_ACCESS_LIN	Non-Secure code cannot access the LIN	0x0*
		NS_CAN_ACCESS_LIN	Non-Secure code can access the LIN	0x1
26	TOF_ACCESS	NS_CANNOT_ACCESS_TOF	Non-Secure code cannot access the TOF	0x0*
		NS_CAN_ACCESS_TOF	Non-Secure code can access the TOF	0x1
25	PCM_ACCESS	NS_CANNOT_ACCESS_PCM0	Non-Secure code cannot access the PCM0	0x0*
		NS_CAN_ACCESS_PCM0	Non-Secure code can access the PCM0	0x1
24	UART_ACCESS	NS_CANNOT_ACCESS_UART0	Non-Secure code cannot access the UART0	0x0*
		NS_CAN_ACCESS_UART0	Non-Secure code can access the UART0	0x1
23:22	I2C_ACCESS	NS_CANNOT_ACCESS_I2C0	Non-Secure code cannot access the I2C0	0x0*
		NS_CANNOT_ACCESS_I2C1	Non-Secure code cannot access the I2C1	0x0*
		NS_CAN_ACCESS_I2C0	Non-Secure code can access the I2C0	0x1
		NS_CAN_ACCESS_I2C1	Non-Secure code can access the I2C1	0x2
21:20	SPI_ACCESS	NS_CANNOT_ACCESS_SPI0	Non-Secure code cannot access the SPI0	0x0*
		NS_CANNOT_ACCESS_SPI1	Non-Secure code cannot access the SPI1	0x0*
		NS_CAN_ACCESS_SPI0	Non-Secure code can access the SPI0	0x1
		NS_CAN_ACCESS_SPI1	Non-Secure code can access the SPI1	0x2
19	GPIO_SRC_ACCESS	NS_CANNOT_ACCESS_GPIO_SRC	Non-Secure code cannot access the GPIO_SRC	0x0*
		NS_CAN_ACCESS_GPIO_SRC	Non-Secure code can access the GPIO_SRC	0x1
18	GPIO_ACCESS	NS_CANNOT_ACCESS_GPIO	Non-Secure code cannot access the GPIO	0x0*
		NS_CAN_ACCESS_GPIO	Non-Secure code can access the GPIO	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
17	CC312_ACCESS	NS_CANNOT_ACCESS_CC312	Non-Secure code cannot access the CC312	0x0*
		NS_CAN_ACCESS_CC312	Non-Secure code can access the CC312	0x1
16	ASCC_ACCESS	NS_CANNOT_ACCESS_ASCC	Non-Secure code cannot access the ASCC	0x0*
		NS_CAN_ACCESS_ASCC	Non-Secure code can access the ASCC	0x1
15:12	TIMER_ACCESS	NS_CANNOT_ACCESS_TIMER0	Non-Secure code cannot access the Timer0	0x0*
		NS_CANNOT_ACCESS_TIMER1	Non-Secure code cannot access the Timer1	0x0*
		NS_CANNOT_ACCESS_TIMER2	Non-Secure code cannot access the Timer2	0x0*
		NS_CANNOT_ACCESS_TIMER3	Non-Secure code cannot access the Timer3	0x0*
		NS_CAN_ACCESS_TIMER0	Non-Secure code can access the Timer0	0x1
		NS_CAN_ACCESS_TIMER1	Non-Secure code can access the Timer1	0x2
		NS_CAN_ACCESS_TIMER2	Non-Secure code can access the Timer2	0x4
		NS_CAN_ACCESS_TIMER3	Non-Secure code can access the Timer3	0x8
10	CRC_ACCESS	NS_CANNOT_ACCESS_CRC	Non-Secure code cannot access the CRC	0x0*
		NS_CAN_ACCESS_CRC	Non-Secure code can access the CRC	0x1
8	NMI_ACCESS	NS_CANNOT_ACCESS_NMI	Non-Secure code cannot access the NMI	0x0*
		NS_CAN_ACCESS_NMI	Non-Secure code can access the NMI	0x1
7	WATCHDOG_ACCESS	NS_CANNOT_ACCESS_WATCHDOG	Non-Secure code cannot access the Watchdog	0x0*
		NS_CAN_ACCESS_WATCHDOG	Non-Secure code can access the Watchdog	0x1
6	SYSCTRL_ACCESS	NS_CANNOT_ACCESS_SYSCTRL	Non-Secure code cannot access the SYSCTRL	0x0*
		NS_CAN_ACCESS_SYSCTRL	Non-Secure code can access the	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
			SYSCTRL	
5	SENSOR_ACCESS	NS_CANNOT_ACCESS_SENSOR	Non-Secure code cannot access the Sensor	0x0*
		NS_CAN_ACCESS_SENSOR	Non-Secure code can access the Sensor	0x1
4	ACS_ACCESS	NS_CANNOT_ACCESS_ACS	Non-Secure code cannot access the ACS	0x0*
		NS_CAN_ACCESS_ACS	Non-Secure code can access the ACS	0x1
3	LSAD_ACCESS	NS_CANNOT_ACCESS_LSAD	Non-Secure code cannot access the LSAD	0x0*
		NS_CAN_ACCESS_LSAD	Non-Secure code can access the LSAD	0x1
2	TEST_CTRL_ACCESS	NS_CANNOT_ACCESS_TEST_CTRL	Non-Secure code cannot access the TEST_CTRL	0x0*
		NS_CAN_ACCESS_TEST_CTRL	Non-Secure code can access the TEST_CTRL	0x1
1	CLK_ACCESS	NS_CANNOT_ACCESS_CLK	Non-Secure code cannot access the CLK config	0x0*
		NS_CAN_ACCESS_CLK	Non-Secure code can access the CLK config	0x1
0	RESET_ACCESS	NS_CANNOT_ACCESS_RESET	Non-Secure code cannot access the Reset	0x0*
		NS_CAN_ACCESS_RESET	Non-Secure code can access the Reset	0x1

4.5.0.3 SYSCTRL_NS_ACCESS_PERIPH_CFG1

Bit Field	Read/Write	Field Name	Description
11	RW	FLASH_IF_ACCESS	Allow Non-Secure code to access the FLASH_IF[0:0]
10	RW	RF_ACCESS	Allow Non-Secure code to access the RF
8	RW	BB_ACCESS	Allow Non-Secure code to access the BB
3:0	RW	DMA_ACCESS	Allow Non-Secure code to access the DMA[3:0]

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
11	FLASH_IF_ACCESS	NS_CANNOT_ACCESS_FLASH_IF0	Non-Secure code cannot access the FLASH_IF0	0x0*
		NS_CAN_ACCESS_FLASH_IF0	Non-Secure code can access the FLASH_IF0	0x1
10	RF_ACCESS	NS_CANNOT_ACCESS_RF	Non-Secure code cannot access the RF	0x0*
		NS_CAN_ACCESS_RF	Non-Secure code can access the RF	0x1
8	BB_ACCESS	NS_CANNOT_ACCESS_BB	Non-Secure code cannot access the BB	0x0*
		NS_CAN_ACCESS_BB	Non-Secure code can access the BB	0x1
3:0	DMA_ACCESS	NS_CANNOT_ACCESS_DMA0	Non-Secure code cannot access the DMA0	0x0*
		NS_CANNOT_ACCESS_DMA1	Non-Secure code cannot access the DMA1	0x0*
		NS_CANNOT_ACCESS_DMA2	Non-Secure code cannot access the DMA2	0x0*
		NS_CANNOT_ACCESS_DMA3	Non-Secure code cannot access the DMA3	0x0*
		NS_CAN_ACCESS_DMA0	Non-Secure code can access the DMA0	0x1
		NS_CAN_ACCESS_DMA1	Non-Secure code can access the DMA1	0x2
		NS_CAN_ACCESS_DMA2	Non-Secure code can access the DMA2	0x4
		NS_CAN_ACCESS_DMA3	Non-Secure code can access the DMA3	0x8

4.5.0.4 SYSCTRL_NS_ACCESS_RAM_CFG0

Bit Field	Read/Write	Field Name	Description
7:0	RW	DRAM_ACCESS	Allow Non-Secure to access DRAM[7:0]

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
7:0	DRAM_ACCESS	NS_CANNOT_ACCESS_DRAM0	Non-Secure code cannot access the DRAM0	0x0*
		NS_CANNOT_ACCESS_DRAM1	Non-Secure code cannot access the DRAM1	0x0*
		NS_CANNOT_ACCESS_DRAM2	Non-Secure code cannot access the DRAM2	0x0*
		NS_CANNOT_ACCESS_DRAM3	Non-Secure code cannot access the DRAM3	0x0*
		NS_CANNOT_ACCESS_DRAM4	Non-Secure code cannot access the	0x0*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
			DRAM4	
		NS_CANNOT_ACCESS_DRAM5	Non-Secure code cannot access the DRAM5	0x0*
		NS_CANNOT_ACCESS_DRAM6	Non-Secure code cannot access the DRAM6	0x0*
		NS_CANNOT_ACCESS_DRAM7	Non-Secure code cannot access the DRAM7	0x0*
		NS_CAN_ACCESS_DRAM0	Non-Secure code can access the DRAM0	0x1
		NS_CAN_ACCESS_DRAM1	Non-Secure code can access the DRAM1	0x2
		NS_CAN_ACCESS_DRAM2	Non-Secure code can access the DRAM2	0x4
		NS_CAN_ACCESS_DRAM3	Non-Secure code can access the DRAM3	0x8
		NS_CAN_ACCESS_DRAM4	Non-Secure code can access the DRAM4	0x10
		NS_CAN_ACCESS_DRAM5	Non-Secure code can access the DRAM5	0x20
		NS_CAN_ACCESS_DRAM6	Non-Secure code can access the DRAM6	0x40
		NS_CAN_ACCESS_DRAM7	Non-Secure code can access the DRAM7	0x80

4.5.0.5 SYSCTRL_NS_ACCESS_RAM_CFG1

Bit Field	Read/Write	Field Name	Description
1:0	RW	BB_DRAM_ACCESS	Allow Non-Secure code to access BB_DRAM[1:0]

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
1:0	BB_DRAM_ACCESS	NS_CANNOT_ACCESS_BB_DRAM0	Non-Secure code cannot access the BB_DRAM0	0x0*
		NS_CANNOT_ACCESS_BB_DRAM1	Non-Secure code cannot access the BB_DRAM1	0x0*
		NS_CAN_ACCESS_BB_DRAM0	Non-Secure code can access the BB_ DRAM0	0x1
		NS_CAN_ACCESS_BB_DRAM1	Non-Secure code can access the BB_ DRAM1	0x2

4.5.0.6 SYSCTRL_DEU_STATUS

Bit Field	Read/Write	Field Name	Description
17	W	OVERFLOW_CLEAR	Clear OVERFLOW flag
16	W	FIRST_DAP_W_FLAG_CLEAR	Clear the FIRST_DAP_W_FLAG
12	R	FIRST_DAP_W_FLAG	First Debug access port write flag
8	R	OVERFLOW	High when DEU_DATA register is written before been read by any bus. Clear via CLR_OVERFLOW action bit.
4	R	SBUS_W	Set when SBus writes the DEU_DATA, clear when the debug access port read the DEU_DATA register
0	R	DAP_W	Set when the debug access port writes the DEU_DATA, clear when the SBus reads the DEU_DATA register.

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
17	OVERFLOW_CLEAR	DEU_OVERFLOW_CLEAR	Clear the flag OVERFLOW	0x1
16	FIRST_DAP_W_FLAG_CLEAR	DEU_FIRST_DAP_W_FLAG_CLEAR	Clear the flag FIRST_DAP_W_FLAG	0x1
12	FIRST_DAP_W_FLAG	FIRST_DAP_W_FLAG_LOW	First DAP write flag low	0x0*
		FIRST_DAP_W_FLAG_HIGH	First DAP write flag high	0x1
8	OVERFLOW	NO_OVERFLOW	No overflow on DEU_DATA register	0x0*
		OVERFLOW	Overflow on DEU_DATA register	0x1
4	SBUS_W	DAP_R_DATA	Debug access port has read the DEU_DATA register.	0x0*
		SBUS_W_DATA	SBus has written DEU_DATA register.	0x1
0	DAP_W	SBUS_R_DATA	SBus has read the DEU_DATA register.	0x0*
		DAP_W_DATA	Debug access port has written DEU_DATA register.	0x1

4.5.0.7 SYSCTRL_DEU_DATA

Bit Field	Read/Write	Field Name	Description
31:0	RW	DEU_DATA	Data exchange unit data

4.5.0.8 SYSCTRL_PROD_STATUS

Bit Field	Read/Write	Field Name	Description
31:0	W	PROD_STATUS	Production Status

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:0	PROD_STATUS	PROD_DONE	Indicate the production test is done.	0x5072446F

4.5.0.9 SYSCTRL_CC_DCU_EN0

Bit Field	Read/Write	Field Name	Description
31	R	CC_DCU_EN_ICV_GP	Always On ICV governed dcu_en0 general purpose bits
30:28	R	CC_DCU_EN_ICV_EH	Always On ICV governed energy harversting signature. Majority of the bis must be set to confirm the state.
27:25	R	CC_DCU_EN_ICV_PRDSTATE	Always On ICV governed production state identifier
24:22	R	CC_DCU_EN_ICV_TCTRL_ACC	Always On ICV governed test control configuration access control. Majority of the bits must be set to enable the feature
21:19	R	CC_DCU_EN_ICV_TRIM_ACC	Always On ICV governed MNVR and chip trim access control. Majority of the bits must be set to enable the feature
18:16	R	CC_DCU_EN_ICV_NVM_ACC	Always On ICV governed NVM access control. Majority of the bits must be set to enable the feature
15:13	R	CC_DCU_EN_ICV_SEC_RST	Always On ICV governed secure reset enable. If any of these bits are set and the part is in SE state, the cc312_top is reset
12:10	R	CC_DCU_EN_ICV_SPINDEN	Always On ICV governed CM33 secure non-intrusive debug enable control. Majority of the bits must be set to enable the feature
9:7	R	CC_DCU_EN_ICV_SPIDEN	Always On ICV governed CM33 secure intrusive debug enable control. Majority of the bits must be set to enable the feature
6:4	R	CC_DCU_EN_ICV_NIDEN	Always On ICV governed CM33 non-intrusive debug enable control. Majority of the bits must be set to enable the feature
3:1	R	CC_DCU_EN_ICV_DBGEN	Always On ICV governed CM33 debug enable control. Majority of the bits must be set to enable the feature
0	R	CC_DCU_EN_ICV_SEC_RST_MASK	Always On ICV governed secure reset mask used in FW

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
0	CC_DCU_EN_ICV_SEC_RST_MASK	CC_SECURE_RST_FW_EN	The FW does not mask the secure debug reset bits	0x0*
		CC_SECURE_RST_FW_DIS	The FW masks the secure debug reset bits, no reset can be issued	0x1

4.5.0.10 SYSCTRL_CC_DCU_EN1

Bit Field	Read/Write	Field Name	Description
31:16	R	CC_DCU_EN_OEM_GP	Always On OEM governed dcu_en1 general purpose bits
15:13	R	CC_DCU_EN_OEM_SEC_RST	Always On OEM governed secure reset enable. If any of these bits are set and the part is in SE state, the cc312_top is reset
12:10	R	CC_DCU_EN_OEM_SPINDEN	Always On OEM governed CM33 secure non-intrusive debug enable control. Majority of the bits must be set to enable the feature
9:7	R	CC_DCU_EN_OEM_SPIDEN	Always On OEM governed CM33 secure intrusive debug enable control. Majority of the bits must be set to enable the feature
6:4	R	CC_DCU_EN_OEM_NIDEN	Always On OEM governed CM33 non-intrusive debug enable control. Majority of the bits must be set to enable the feature
3:1	R	CC_DCU_EN_OEM_DBGEN	Always On OEM governed CM33 debug enable control. Majority of the bits must be set to enable the feature
0	R	CC_DCU_EN_OEM_RESERVED	Always On OEM allocated reserved bit (unused)

4.5.0.11 SYSCTRL_CC_DCU_EN2

Bit Field	Read/Write	Field Name	Description
31:0	R	CC_DCU_EN2	Always On block DCU_EN2 state

4.5.0.12 SYSCTRL_CC_DCU_EN3

Bit Field	Read/Write	Field Name	Description
31:0	R	CC_DCU_EN3	Always On block DCU_EN3 state

4.5.0.13 SYSCTRL_CC_DCU_LOCK0

Bit Field	Read/Write	Field Name	Description
31	R	CC_DCU_LOCK_ICV_GP	Always On dcu_en0 general purpose bits lock. All bits must be locked to assure that the state is locked
30:28	R	CC_DCU_LOCK_ICV_EH	Always On energy harvesting signature lock. All bits must be locked to assure that the state is locked
27:25	R	CC_DCU_LOCK_ICV_PRDSTATE	Always On production state identifier lock. All bits must be locked to assure that the state is locked
24:22	R	CC_DCU_LOCK_ICV_TCTRL_ACC	Always On test control configuration access control lock. All bits must be locked to assure that the state is locked
21:19	R	CC_DCU_LOCK_ICV_TRIM_ACC	Always On MNVR and chip trim access control lock. All bits must be locked to assure that the state is locked
18:16	R	CC_DCU_LOCK_ICV_NVM_ACC	Always On NVM access control lock. All bits must be locked to assure that the state is locked

Bit Field	Read/Write	Field Name	Description
15:13	R	CC_DCU_LOCK_ICV_SEC_RST	Always On ICV secure reset enable lock. All bits must be locked to assure that the state is locked
12:10	R	CC_DCU_LOCK_ICV_SPINDEN	Always On ICV CM33 secure non-intrusive debug enable control lock. All bits must be locked to assure that the state is locked
9:7	R	CC_DCU_LOCK_ICV_SPIDEN	Always On ICV CM33 secure intrusive debug enable control lock. All bits must be locked to assure that the state is locked
6:4	R	CC_DCU_LOCK_ICV_NIDEN	Always On ICV CM33 non-intrusive debug enable control lock. All bits must be locked to assure that the state is locked
3:1	R	CC_DCU_LOCK_ICV_DBGEN	Always On ICV CM33 debug enable control lock. All bits must be locked to assure that the state is locked
0	R	CC_DCU_LOCK_ICV_SEC_RST_MASK	Always On ICV secure reset mask lock

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
0	CC_DCU_LOCK_ICV_SEC_RST_MASK	CC_SECURE_RST_MASK_UNLOCKED	The FW does not lock the secure debug reset mask bit	0x0*
		CC_SECURE_RST_MASK_LOCKED	The FW locks the secure debug reset mask bit	0x1

4.5.0.14 SYSCTRL_CC_DCU_LOCK1

Bit Field	Read/Write	Field Name	Description
31:16	R	CC_DCU_LOCK_OEM_GP	Always On OEM governed dcu_en1 general purpose bits lock
15:13	R	CC_DCU_LOCK_OEM_SEC_RST	Always On OEM governed secure reset lock. All bits must be locked to assure that the state is locked
12:10	R	CC_DCU_LOCK_OEM_SPINDEN	Always On OEM governed CM33 secure non-intrusive debug enable control lock. All bits must be locked to assure that the state is locked
9:7	R	CC_DCU_LOCK_OEM_SPIDEN	Always On OEM governed CM33 secure intrusive debug enable control lock. All bits must be locked to assure that the state is locked
6:4	R	CC_DCU_LOCK_OEM_NIDEN	Always On OEM governed CM33 non-intrusive debug enable control lock. All bits must be locked to assure that the state is locked
3:1	R	CC_DCU_LOCK_OEM_DBGEN	Always On OEM governed CM33 debug enable control lock. All bits must be locked to assure that the state is locked
0	R	CC_DCU_LOCK_OEM_RESERVED	Always On OEM allocated reserved bit (unused)

4.5.0.15 SYSCTRL_CC_DCU_LOCK2

Bit Field	Read/Write	Field Name	Description
31:0	R	CC_DCU_LOCK2	Always On block DCU_LOCK2 state

4.5.0.16 SYSCTRL_CC_DCU_LOCK3

Bit Field	Read/Write	Field Name	Description
31:0	R	CC_DCU_LOCK3	Always On block DCU_LOCK3 state

4.5.0.17 SYSCTRL_CC_STATUS

Bit Field	Read/Write	Field Name	Description
25	R	CC_SEC_DEBUG_RESET	Always On block secure debug reset status
24	R	CC_HOST_DFA_ENABLE_LOCK	Always On block host AES DFA lock status
23	R	CC_HOST_FORCE_DFA_ENABLE	Always On block host AES DFA status
22	R	CC_RESET_UPON_DEBUG_ DISABLE	Always On block HUK reset mechanism configuration status
21	R	CC_HOST_ICV_RMA_LOCK	Always On block host icv rma bit in NVM lock status
20	R	CC_HOST_KCE_LOCK	Always On block host OEM code encryption key lock status
19	R	CC_HOST_KCP_LOCK	Always On block host OEM provisioning key lock status
18	R	CC_HOST_KCEICV_LOCK	Always On block host ICV code encryption key lock status
17	R	CC_HOST_KPICV_LOCK	Always On block host ICV provisioning key lock status
16	R	CC_HOST_FATAL_ERR	Always On block host fatal error flag
15	R	CC_APB_ONLY_PRIV_ACCESS_ LOCK	Always On block APB filtering privileged access configuration lock
14	R	CC_APB_ONLY_PRIV_ACCESS	Always On block APB filtering privileged access configuration
13	R	CC_APB_ONLY_SEC_ACCESS_ LOCK	Always On block APB filtering secure access configuration lock
12	R	CC_APB_ONLY_SEC_ACCESS	Always On block APB filtering secure access configuration
11:4	R	CC_GPPC	Always On block GPPC register
3	R	CC_LCS_VALID	Always On block life cycle state valid
2:0	R	CC_LCS	Always On block life cycle state

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
25	CC_SEC_DEBUG_ RESET	CC_SEC_DBG_RST_IDLE	The CC312 does not have the secure debug reset activated	0x0*
		CC_SEV_DBG_RST_ACTIVATED	The CC312 has secure debug reset fired, can't fire again	0x1
24	CC_HOST_DFA_ ENABLE_LOCK	CC_DFA_CFG_UNLOCKED	The CC312 AES DFA configuration register is unlocked	0x0*
		CC_DFA_CFG_LOCKED	The CC312 AES DFA configuration register is locked	0x1
23	CC_HOST_FORCE_ DFA_ENABLE	CC_DFA_DIS	The CC312 AES DFA counter measures are not enabled	0x0*
		CC_DFA_EN	The CC312 AES DFA counter measures are enabled	0x1
22	CC_RESET_UPON_ DEBUG_DISABLE	CC_RESET_UPON_DEBUG_EN	The CC312 HUK reset mechanism is enabled	0x0*
		CC_RESET_UPON_DEBUG_DIS	The CC312 HUK reset mechanism is disabled	0x1
21	CC_HOST_ICV_RMA_ LOCK	CC_ICV_RMA_UNLOCKED	The CC312 icv_rma flag in NVM can be written	0x0*
		CC_ICV_RMA_LOCKED	The CC312 icv_rma flag in NVM is locked, can't be written	0x1
20	CC_HOST_KCE_LOCK	CC_KCE_UNLOCKED	The CC312 KCE is unlocked	0x0*
		CC_KCE_LOCKED	The CC312 KCE is masked, locked by the SW	0x1
19	CC_HOST_KCP_LOCK	CC_KCP_UNLOCKED	The CC312 KCP is unlocked	0x0*
		CC_KCP_LOCKED	The CC312 KCP is masked, locked by the SW	0x1
18	CC_HOST_KCEICV_ LOCK	CC_KCEICV_UNLOCKED	The CC312 KCEICV is unlocked	0x0*
		CC_KCEICV_LOCKED	The CC312 KCEICV is masked, locked by the SW	0x1
17	CC_HOST_KPICV_LOCK	CC_KPICV_UNLOCKED	The CC312 KPICV is unlocked	0x0*
		CC_KPICV_LOCKED	The CC312 KPICV is masked, locked by the SW	0x1
16	CC_HOST_FATAL_ERR	CC_HOST_NO_FATAL_ERROR	The CC312 does not have any fatal error	0x0*
		CC_HOST_FATAL_ERROR	The CC312 reports a fatal error	0x1
15	CC_APB_ONLY_PRIV_	CC_APB_PRIV_CFG_UNLOCK	The CC312 privileged access	0x0*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
	ACCESS_LOCK		configuration field unlocked	
		CC_APB_PRIV_CFG_LOCK	The CC312 privileged access configuration field locked	0x1
14	CC_APB_ONLY_PRIV_ACCESS	CC_APB_NORMAL_ACCESS	The CC312 serve both privileged and normal access	0x0*
		CC_APB_PRIV_ACCESS	The CC312 serve only privileged access	0x1
13	CC_APB_ONLY_SEC_ACCESS_LOCK	CC_APB_SEC_CFG_UNLOCK	The CC312 secure access configuration field is unlocked	0x0*
		CC_APB_SEC_CFG_LOCKED	The CC312 secure access configuration field is locked	0x1
12	CC_APB_ONLY_SEC_ACCESS	CC_APB_NON_SEC_ACCESS	The CC312 allow both secure and non-secure access	0x0*
		CC_APB_ONLY_SEC_ACCESS	The CC312 allow only secure access	0x1
3	CC_LCS_VALID	CC_LCS_IS_INVALID	The CC312 does not have LCS defined after reset	0x0*
		CC_LCS_IS_VALID	The CC312 has LCS defined after reset	0x1
2:0	CC_LCS	CC_LCS_CM	The CC312 is in CM lifecycle state	0x0*
		CC_LCS_DM	The CC312 is in DM lifecycle state	0x1
		CC_LCS_SECURE	The CC312 is in Secure lifecycle state	0x5
		CC_LCS_RMA	The CC312 is in RMA lifecycle state	0x7

4.5.0.18 SYSCTRL_CC_FEATURES_CTRL

Bit Field	Read/Write	Field Name	Description
30	R	CC_OEM_SEC_RST_ALL	Always On block OEM governed DCU_EN bits fault status allocated for enabling SEC_RST operation
29	R	CC_OEM_SPINDEN_ALL	Always On block OEM governed DCU_EN bits fault status allocated for enabling SPINDEN operation
28	R	CC_OEM_SPIDEN_ALL	Always On block OEM governed DCU_EN bits fault status allocated for enabling SPIDEN operation
27	R	CC_OEM_NIDEN_ALL	Always On block OEM governed DCU_EN bits fault status allocated for enabling NIDEN operation
26	R	CC_OEM_DBGEN_ALL	Always On block OEM governed DCU_EN bits fault status allocated for enabling DBGEN operation
25	R	CC_ENERGY_HARVESTING_ALL	Always On block DCU_EN bits fault status allocated for enabling ENERGY_HARVESTING operation

Bit Field	Read/Write	Field Name	Description
24	R	CC_PROD_STATUS_ALL	Always On block DCU_EN bits fault status allocated for enabling PROD_STATUS operation
23	R	CC_TCTRL_ACC_ALL	Always On block DCU_EN bits fault status allocated for enabling TCTRL_ACC operation
22	R	CC_TRIM_ACC_ALL	Always On block DCU_EN bits fault status allocated for enabling TRIM_ACC operation
21	R	CC_NVM_ACC_ALL	Always On block DCU_EN bits fault status allocated for enabling NVM_ACC operation
20	R	CC_ICV_SEC_RST_ALL	Always On block ICV governed DCU_EN bits fault status allocated for enabling SEC_RST operation
19	R	CC_ICV_SPINDEN_ALL	Always On block ICV governed DCU_EN bits fault status allocated for enabling SPINDEN operation
18	R	CC_ICV_SPIDEN_ALL	Always On block ICV governed DCU_EN bits fault status allocated for enabling SPIDEN operation
17	R	CC_ICV_NIDEN_ALL	Always On block ICV governed DCU_EN bits fault status allocated for enabling NIDEN operation
16	R	CC_ICV_DBGEN_ALL	Always On block ICV governed DCU_EN bits fault status allocated for enabling DBGEN operation
13	R	CC_OEM_SPINDEN	Always On block OEM SPINDEN status as the result of the equality check and production state confirmation
12	R	CC_OEM_SPIDEN	Always On block OEM SPIDEN status as the result of the equality check and production state confirmation
11	R	CC_OEM_NIDEN	Always On block OEM NIDEN status as the result of the equality check and production state confirmation
10	R	CC_OEM_DBGEN	Always On block OEM DBGEN status as the result of the equality check and production state confirmation
9	R	CC_ENERGY_HARVESTING	Always On block Energy Harvesting status as the result of equality check
8	R	CC_PROD_STATUS	Always On block Production Status as the result of equality check
7	R	CC_TCTRL_ACC	Always On block TCTRL_ACC status as the result of the equality check and production state confirmation
6	R	CC_TRIM_ACC	Always On block TRIM_ACC status as the result of the equality check and production state confirmation
5	R	CC_NVM_ACC	Always On block NVM_ACC status as the result of the equality check and production state confirmation
4	R	CC_SEC_RST	Always On block ICV or OEM SEC_RST status as the result of any bit being set.

Bit Field	Read/Write	Field Name	Description
3	R	CC_ICV_SPINDEN	Always On block ICV SPINDEN status as the result of the equality check and production state confirmation
2	R	CC_ICV_SPIDEN	Always On block ICV SPIDEN status as the result of the equality check and production state confirmation
1	R	CC_ICV_NIDEN	Always On block ICV NIDEN status as the result of the equality check and production state confirmation
0	R	CC_ICV_DBGEN	Always On block ICV DBGEN status as the result of the equality check and production state confirmation

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
30	CC_OEM_SEC_RST_ ALL	CC_OEM_SEC_RST_DCUEN_MATCH	The DCU_EN bits allocated for the feature are all equal	0x0*
		CC_OEM_SEC_RST_DCUEN_ MISMATCH	The DCU_EN bits allocated for the feature have mismatch	0x1
29	CC_OEM_SPINDEN_ALL	CC_OEM_SPINDEN_DCUEN_MATCH	The DCU_EN bits allocated for the feature are all equal	0x0*
		CC_OEM_SPINDEN_DCUEN_ MISMATCH	The DCU_EN bits allocated for the feature have mismatch	0x1
28	CC_OEM_SPIDEN_ALL	CC_OEM_SPIDEN_DCUEN_MATCH	The DCU_EN bits allocated for the feature are all equal	0x0*
		CC_OEM_SPIDEN_DCUEN_ MISMATCH	The DCU_EN bits allocated for the feature have mismatch	0x1
27	CC_OEM_NIDEN_ALL	CC_OEM_NIDEN_DCUEN_MATCH	The DCU_EN bits allocated for the feature are all equal	0x0*
		CC_OEM_NIDEN_DCUEN_ MISMATCH	The DCU_EN bits allocated for the feature have mismatch	0x1
26	CC_OEM_DBGEN_ALL	CC_OEM_DBGEN_DCUEN_MATCH	The DCU_EN bits allocated for the feature are all equal	0x0*
		CC_OEM_DBGEN_DCUEN_ MISMATCH	The DCU_EN bits allocated for the feature have mismatch	0x1
25	CC_ENERGY_ HARVESTING_ALL	CC_ENERGY_HARVESTING_ DCUEN_MATCH	The DCU_EN bits allocated for the feature are all equal	0x0*
		CC_ENERGY_HARVESTING_ DCUEN_MISMATCH	The DCU_EN bits allocated for the feature have mismatch	0x1
24	CC_PROD_STATUS_ALL	CC_PROD_STATUS_DCUEN_MATCH	The DCU_EN bits allocated for the feature are all equal	0x0*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		CC_PROD_STATUS_DCUEN_MISMATCH	The DCU_EN bits allocated for the feature have mismatch	0x1
23	CC_TCTRL_ACC_ALL	CC_TCTRL_ACC_DCUEN_MATCH	The DCU_EN bits allocated for the feature are all equal	0x0*
		CC_TCTRL_ACC_DCUEN_MISMATCH	The DCU_EN bits allocated for the feature have mismatch	0x1
22	CC_TRIM_ACC_ALL	CC_TRIM_ACC_DCUEN_MATCH	The DCU_EN bits allocated for the feature are all equal	0x0*
		CC_TRIM_ACC_DCUEN_MISMATCH	The DCU_EN bits allocated for the feature have mismatch	0x1
21	CC_NVM_ACC_ALL	CC_NVM_ACC_DCUEN_MATCH	The DCU_EN bits allocated for the feature are all equal	0x0*
		CC_NVM_ACC_DCUEN_MISMATCH	The DCU_EN bits allocated for the feature have mismatch	0x1
20	CC_ICV_SEC_RST_ALL	CC_ICV_SEC_RST_DCUEN_MATCH	The DCU_EN bits allocated for the feature are all equal	0x0*
		CC_ICV_SEC_RST_DCUEN_MISMATCH	The DCU_EN bits allocated for the feature have mismatch	0x1
19	CC_ICV_SPINDEN_ALL	CC_ICV_SPINDEN_DCUEN_MATCH	The DCU_EN bits allocated for the feature are all equal	0x0*
		CC_ICV_SPINDEN_DCUEN_MISMATCH	The DCU_EN bits allocated for the feature have mismatch	0x1
18	CC_ICV_SPIDEN_ALL	CC_ICV_SPIDEN_DCUEN_MATCH	The DCU_EN bits allocated for the feature are all equal	0x0*
		CC_ICV_SPIDEN_DCUEN_MISMATCH	The DCU_EN bits allocated for the feature have mismatch	0x1
17	CC_ICV_NIDEN_ALL	CC_ICV_NIDEN_DCUEN_MATCH	The DCU_EN bits allocated for the feature are all equal	0x0*
		CC_ICV_NIDEN_DCUEN_MISMATCH	The DCU_EN bits allocated for the feature have mismatch	0x1
16	CC_ICV_DBGEN_ALL	CC_ICV_DBGEN_DCUEN_MATCH	The DCU_EN bits allocated for the feature are all equal	0x0*
		CC_ICV_DBGEN_DCUEN_MISMATCH	The DCU_EN bits allocated for the feature have mismatch	0x1
13	CC_OEM_SPINDEN	CC_OEM_SPINDEN_INACTIVE	Secure non-invasive CM33 debug disabled by OEM	0x0*
		CC_OEM_SPINDEN_ACTIVE	Secure non-invasive CM33 debug	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
			enabled by OEM, all bits set	
12	CC_OEM_SPIDEN	CC_OEM_SPIDEN_INACTIVE	Secure invasive CM33 debug disabled by OEM	0x0*
		CC_OEM_SPIDEN_ACTIVE	Secure invasive CM33 debug enabled by OEM, all bits set	0x1
11	CC_OEM_NIDEN	CC_OEM_NIDEN_INACTIVE	Non-invasive CM33 debug disabled by OEM	0x0*
		CC_OEM_NIDEN_ACTIVE	Non-invasive CM33 debug enabled by OEM, all bits set	0x1
10	CC_OEM_DBGEN	CC_OEM_DBGEN_INACTIVE	CM33 debug disabled by OEM	0x0*
		CC_OEM_DBGEN_ACTIVE	CM33 debug enabled by OEM, all bits set	0x1
9	CC_ENERGY_ HARVESTING	CC_ENERGY_HARVESTING_ INACTIVE	Energy harvesting is inactive.	0x0*
		CC_ENERGY_HARVESTING_ ACTIVE	Energy harvesting is active, all bits set	0x1
8	CC_PROD_STATUS	CC_PROD_NOT_DONE	Production is not done.	0x0*
		CC_PROD_DONE	Production screening done, all bits set	0x1
7	CC_TCTRL_ACC	CC_TCTRL_ACC_INACTIVE	Test_ctrl register access disabled	0x0*
		CC_TCTRL_ACC_ACTIVE	Test_ctrl register access enabled, all bits set	0x1
6	CC_TRIM_ACC	CC_TRIM_ACC_INACTIVE	MNVR and chip trim access disabled	0x0*
		CC_TRIM_ACC_ACTIVE	MNVR and chip trim access enabled, all bits set	0x1
5	CC_NVM_ACC	CC_NVM_ACC_INACTIVE	NVM access limited	0x0*
		CC_NVM_ACC_ACTIVE	NVM access available for ERASE and WRITE command, all bits set	0x1
4	CC_SEC_RST	CC_SEC_RST_INACTIVE	None of the bits are set, the feature is deactivated by ICV or OEM	0x0*
		CC_SEC_RST_ACTIVE	Some of the bits are set by ICV or OEM to activate the feature	0x1
3	CC_ICV_SPINDEN	CC_ICV_SPINDEN_INACTIVE	Secure non-invasive CM33 debug disabled by ICV	0x0*
		CC_ICV_SPINDEN_ACTIVE	Secure non-invasive CM33 debug enabled by ICV, all bits set	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
2	CC_ICV_SPIDEN	CC_ICV_SPIDEN_INACTIVE	Secure invasive CM33 debug disabled by ICV	0x0*
		CC_ICV_SPIDEN_ACTIVE	Secure invasive CM33 debug enabled by ICV, all bits set	0x1
1	CC_ICV_NIDEN	CC_ICV_NIDEN_INACTIVE	Non-invasive CM33 debug disabled by ICV	0x0*
		CC_ICV_NIDEN_ACTIVE	Non-invasive CM33 debug enabled by ICV, all bits set	0x1
0	CC_ICV_DBGEN	CC_ICV_DBGEN_INACTIVE	CM33 debug disabled by ICV	0x0*
		CC_ICV_DBGEN_ACTIVE	CM33 debug enabled by ICV, all bits set	0x1

CHAPTER 5

RF Front-End

The RF front-end (RFFE) transceiver is an ultra low-power 2.4 GHz radio. It can support several wireless protocols, such as Bluetooth Low Energy, and custom and proprietary protocols.

5.1 Overview

The RF front-end communicates with:

- The CPU and the DMA, through a dedicated bridge that accesses the RF front-end internal configuration registers. The CPU 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 depending on various factors, and additional wait states are inserted when an SPI operation is active. Note that the SPI interface has priority over the APB interface.
- The CPU via 6 interrupts.
- The Arm Cortex-M33 processor (CPU), which uses a simple baseband to provide packet handling, and data transfers (supported by interrupts and GPIOs as proprietary debug resources). This mode of operation can be used to implement custom protocols.
- The baseband controller, through the internal SPI interface and dedicated CLK and DATA signals. All these signals are multiplexed through the GPIO block. This mode of operation is used along with provided SW Bluetooth Low Energy stack as part of the RSL15 SDK.

The Arm Cortex-M33 processor, the DMA (through the APB bus), and the internal SPI interface all have access to the RFFE registers. However, the user application must ensure that no SPI transaction is ongoing when accessing the RFFE over the APB bus. The RF is powered by the VDDRF and VDDC regulators. Digital part of RFFE is powered through VDDC when a power switch is closed through the `SYSCTRL_RF_POWER_CFG` register, and access is granted through the `SYSCTRL_RF_ACCESS_CFG` register. The analog part is powered when VDDRF is enabled.

The simplified block diagram of RFFE is shown in the "RFFE Block Diagram" figure (Figure 3).

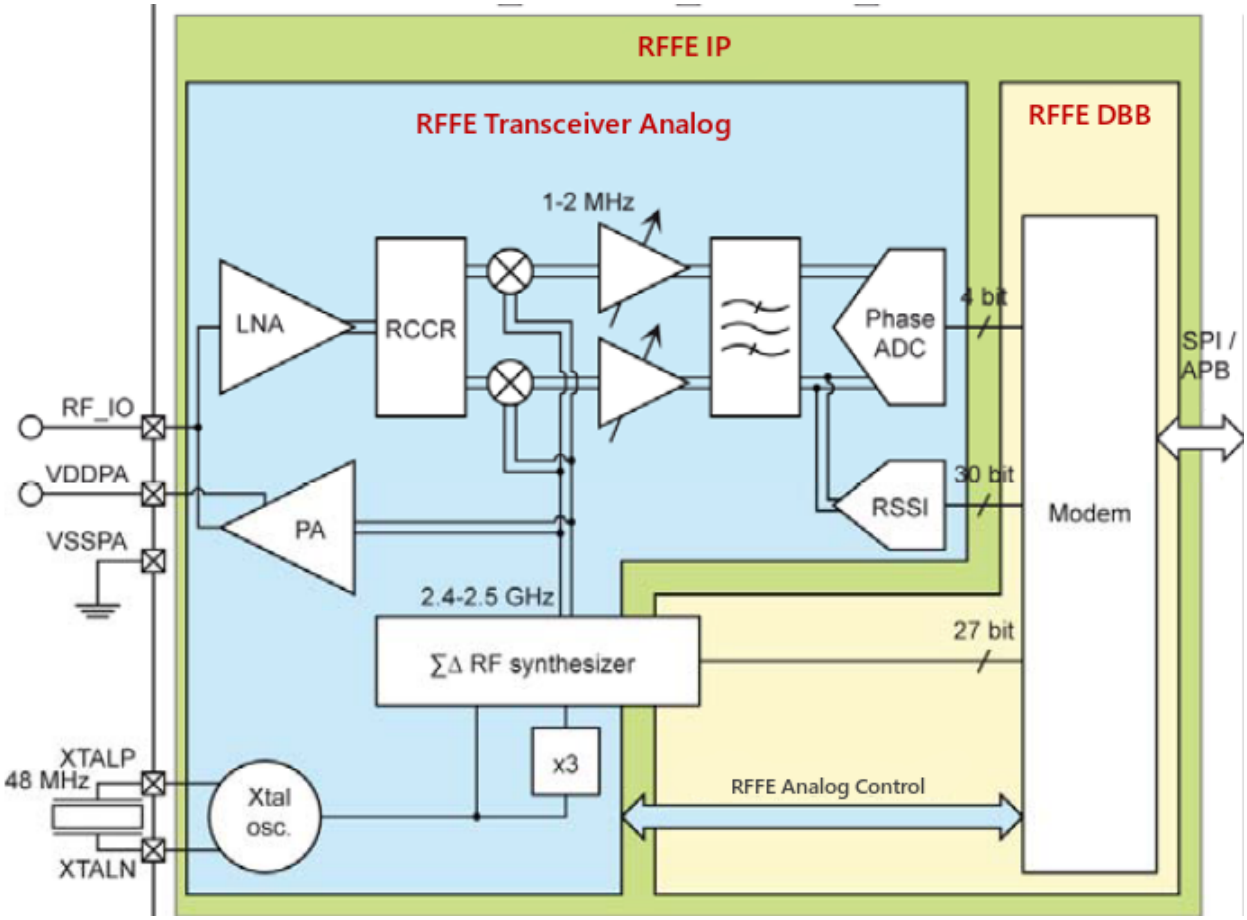


Figure 3. RFFE Block Diagram

The digital block of the RFFE contains a full transceiver with the following features:

- FSK modem with programmable pulse shape and modulation index
- Data-rate programmable from 3 Mbits/s to 62.5 kbits/s (4 Mbits/s with 4-FSK)
- IEEE 802.15.4 chip encoding & decoding
- Manchester encoding
- Data whitening
- Packet handling
- Automatic preamble and sync word insertion
- Automatic packet length handler
- Basic address check
- Automatic CRC calculation and verification with a programmable CRC polynomial
- Multi-frame support
- 2x128 bytes FIFO
- Bluetooth Low Energy Long Range
- SPI and AMBA APB interfaces

Combined with the 2 Mbits/s analog front-end, the RFFE is capable of addressing Bluetooth Low Energy and Zigbee modulations.

RFFE is clocked from the 48 MHz crystal oscillator and is powered and enabled through registers ACS_VDDRF_CTRL, SYSCTRL_RF_POWER_CFG, and SYSCTRL_RF_ACCESS_CFG.

The power amplifier (PA) of RFFE is powered through VDDRF or VDDPA, configured by the user application for desired output power. For power outputs higher than 2 dBm, VDDPA needs to be enabled and connected to the PA (power amplifier) supply; otherwise VDDPA can be disabled and VDDRF supplies PA.

5.2 RFFE Interface

5.2.1 RFFE PowerUp

The RF front-end interface is controlled by two blocks/register groups that respectively configure the components external to the RFFE, and configure and access the components that are internal to the RFFE. The first register group is part of the SYSCTRL registers that are used to power and access RFFE, and to control or configure the interface to RFFE. Without having RFFE enabled they are accessible, and are located outside of the RFFE block. The second group of registers are located inside the RFFE block, and are accessible only if the SYSCTRL_RF_POWER_CFG and SYSCTRL_RF_ACCESS_CFG registers are configured for RF power and access.

The following steps are required to power up the RFFE:

1. Enable power to the RFFE interface, by enabling the VDDRF supply or ensuring that it has been previously enabled.
2. Disable the VDDPA if it is not needed, connecting VDDPA to VDDRF.
3. Enable the RFFE itself.
4. Enable the 48 MHz crystal oscillator needed by the RFFE.
5. Route interrupts from the RFFE to the system.

To enable the VDDRF and disable VDDPA if it is not needed, a user application needs to perform the following steps:

1. Enable the VDDRF supply, by setting the ACS_VDDRF_CTRL_VDDRF_ENABLE bit in the ACS_VDDRF_CTRL register, making no changes to the trimming settings.
2. Wait until the ACS_VDDRF_CTRL_VDDRF_READY bit from the ACS_VDDRF_CTRL register is set to VDDRF_READY, indicating the VDDRF supply has powered up.
3. Connect VDDPA to VDDRF, by disabling the VDDPA regulator. This is accomplished by clearing the ACS_VDDPA_CTRL_VDDPA_ENABLE bit in the ACS_VDDPA_CTRL register to VDDPA_DISABLE.

To enable RFFE, the following sequence needs to be applied:

1. Set the SYSCTRL_RF_POWER_CFG_RF_STARTUP field of register SYSCTRL_RF_POWER_CFG.
2. Set the SYSCTRL_RF_POWER_CFG_RF_ENABLE bit of the same register (keeping the startup bit set), to enable the power switch.
3. Set the SYSCTRL_RF_ACCESS_CFG_RF_ACCESS bit in register SYSCTRL_RF_ACCESS_CFG, to grant access.

A power-up duration of 1.3 µs is automatically enforced using wait states on the internal bus when powering up the RF block. See Chapter 5.9 "RF Front-End Registers" on page 188 for more details.

NOTE: To disable the RFFE, follow the enabling steps in reverse order by disabling access first and then disabling the power switch.

To enable the 48 MHz XTAL oscillator to provide the clock to RFFE, perform the following steps:

1. Set the RF_XTAL_CTRL_DISABLE_OSCILLATOR and RF_XTAL_CTRL_XTAL_CTRL_REG_VALUE_SEL_INTERNAL bits from the RF_XTAL_CTRL register, to start the 48 MHz crystal oscillator.
2. Set the RF0_REG33_CK_DIV_1_6_CK_DIV_1_6 field in the RF0_REG33 register to the desired prescale value for enabling the 48 MHz crystal oscillator divider.
3. Pause the code until the 48 MHz crystal oscillator is started, using value of the RF0_ANALOG_INFO_ANALOG_INFO_CLK_DIG_READY bit in the RF0_ANALOG_INFO register a Ready signal for the clock.

To enable the routing of interrupts from the RFFE baseband to the system, the SYSCTRL_RF_ACCESS_CFG_RF_IRQ_ACCESS field of register SYSCTRL_RF_ACCESS_CFG needs to be enabled. This configuration allows an application to enable and use the internal configuration of the IRQs of the RFFE as well, as described later in this chapter.

5.2.2 VDDPA Dynamic Control

VDDPA is required to be enabled for output powers higher than 2 dBm. It can be enabled and used for lower output powers as well, but this causes more power consumption. To mitigate the affect of power amplifier activation during signal transmission and help to pass band edge regulatory tests, there is a hardware mechanism called VDDPA dynamic control, which can be enabled by setting the SYSCTRL_VDDPA_CFG0_DYNAMIC_CTRL bit of the SYSCTRL_VDDPA_CFG0 register. The purpose of the mechanism is twofold:

1. Dynamically enable/disable the VDDPA regulator during (respectively) TX and RX operations, to minimize the power consumption. The mechanism automatically controls the enabling/switching of the VDDPA regulator when the ACS_VDDPA_CTRL_VDDPA_ENABLE bit in the ACS_VDDPA_CTRL register is reset, and the ACS_VDDPA_CTRL_VDDPA_SW_CTRL bit in the same register is set.
2. Mitigate possible out-of-band issues by applying a progressive voltage ramp-up/-down on the VDDPA regulator supplying the RF front-end PA. The mechanism dynamically controls the VDDPA voltage in the range specified by the ACS_VDDPA_CTRL_VDDPA_INITIAL_VTRIM and ACS_VDDPA_CTRL_VDDPA VTRIM fields of the ACS_VDDPA_CTRL register.

The different voltage ramp-up/-down steps need to be properly synchronized with the RFFE FSM TX operations as shown in the "Timing Relationship Between VDDPA Dynamic Control Mechanism and RFFE Operations" figure (Figure 4). All the times and delays hereafter are specified in system clock cycles. Note that when VDDRF is used to power PA, there is no need to enable this mechanism and it has no effect.

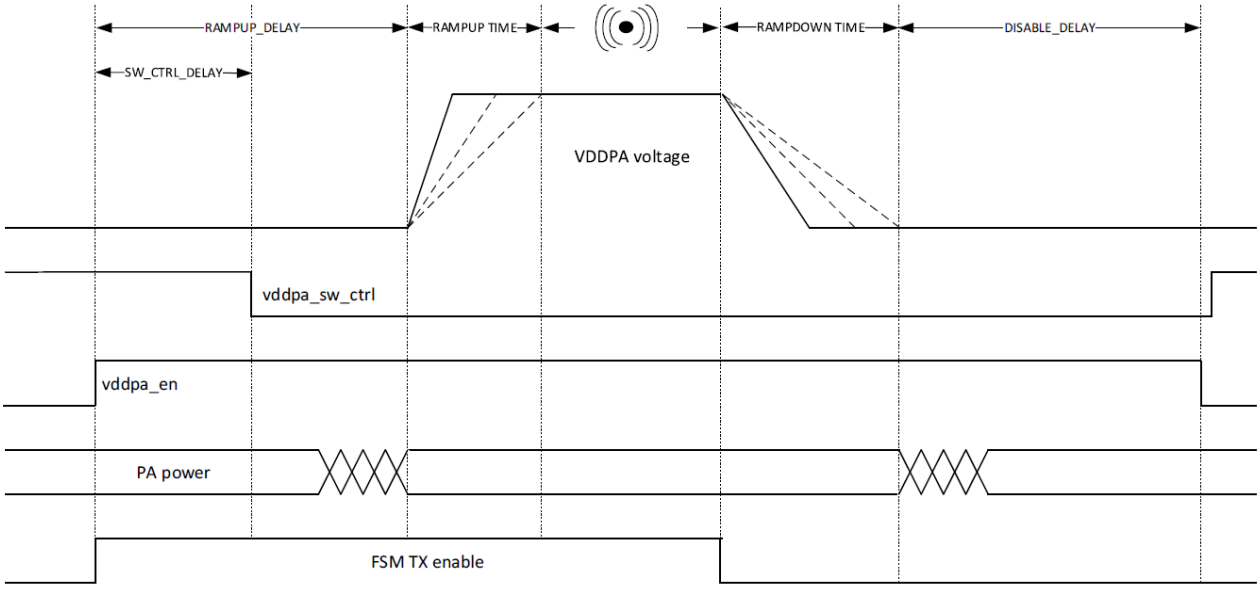


Figure 4. Timing Relationship Between VDDPA Dynamic Control Mechanism and RFFE Operations

1. Apply the voltage ramp-up in the continuity of the internal RF front-end PA power ramp-up, but before the transmission of the first bit over the air:
    a. The VDDPA regulator is automatically enabled when the FSM starts in TX mode.
    b. The SW_CTRL switches the PA supply source from VDDRF to VDDPA after the delay specified in the SYSCTRL_VDDPA_CFG0_SW_CTRL_DELAY field of the SYSCTRL_VDDPA_CFG0 register. The VDDPA regulator is trimmed at the value specified in the ACS_VDDPA_CTRL_INITIAL_VTRIM field of the ACS_VDDPA_CTRL register.
    c. The voltage ramp-up starts after the delay specified in the SYSCTRL_VDDPA_CFG0_RAMPUP_DELAY field of the SYSCTRL_VDDPA_CFG0 register. The voltage ramp-up stops when VDDPA reaches the VTRIM trim target specified in the ACS_VDDPA_CTRL_INITIAL_VTRIM field of the ACS_VDDPA_CTRL register.
    d. The number of ramp-up voltage steps is configured in the SYSCTRL_VDDPA_CFG1_RAMPUP_STEP field of the SYSCTRL_VDDPA_CFG1 register, while the duration of each step is defined in the SYSCTRL_VDDPA_CFG1_RAMPUP_STEP_TIME field of the same register. Consequently, the total ramp-up time is given by SYSCTRL_VDDPA_CFG1_RAMPUP_STEP times SYSCTRL_VDDPA_CFG1_RAMPUP_STEP_TIME.
2. Apply the voltage ramp-down when TX stops, but before the start of the internal RF front-end PA power ramp-down:
    a. The voltage ramp-down automatically starts when the FSM stops. The voltage ramp-down stops when VDDPA reaches the trim target value specified in the ACS_VDDPA_CTRL_INITIAL_VTRIM field of the ACS_VDDPA_CTRL register.
    b. The amount of ramp-down voltage steps is configured in the SYSCTRL_VDDPA_CFG1_RAMDOWN_STEP field of the SYSCTRL_VDDPA_CFG1 register, while the duration of each step is defined in the SYSCTRL_VDDPA_CFG1_RAMPDOWN_STEP_TIME field of the same register. Consequently, the total ramp-down time is given by SYSCTRL_VDDPA_CFG1_RAMPDOWN_STEP times SYSCTRL_VDDPA_CFG1_RAMPDOWN_STEP_TIME.

   c. The VDDPA regulator is disabled once the ramp-down is over and after the delay specified in the SYSCTRL_VDDPA_CFG0_DISABLE_DELAY field of the SYSCTRL_VDDPA_CFG0 register. The PA supply source is automatically switched from VDDPA to VDDRF one cycle later.
3. The initial VDDPA voltage (given by the ACS_VDDPA_CTRL_INITIAL_VTRIM field of the ACS_VDDPA_CTRL register) needs to be as close as possible to VDDRF. This ensures voltage continuity when switching the PA supply source from VDDPA to VDDRF and vice versa.
4. The dynamic regulator enabling/disabling can be bypassed by setting the ACS_VDDPA_CTRL_VDDPA_ENABLE bit in the ACS_VDDPA_CTRL register, and resetting the ACS_VDDPA_CTRL_VDDPA_SW_CTRL in the same register. In this case, the regulator is kept always on while the dynamic voltage control remains active.
5. We recommend that you enable/disable the VDDPA dynamic control only when the radio is idle (no TX or RX), to avoid unexpected behavior of the VDDPA voltage.
6. The actual status of the VDDPA regulator (trimming, enable and switch) can be read in the SYSCTRL_VDDPA_CFG0 register.

5.2.3 RFFE Data Steaming

The phase ADC has a resolution of 4 bits, and provides the Arctan(Q/I) and RSSI detectors. The RSSI detector outputs a 30-bit thermometric code that is coded as a 5-bit value output in the RFFE modem. The received radio signals after down-conversion to the IF band is sampled thorough the phase ADC and the RSSI detector. While this information is internally passed to the demodulator of the RFFE, it is possible that the information is instead routed to the RFFE interface for access by system. There are two types of data that can be routed and streamed to the Arm Cortex-M33 processor or the DMA:

- RAW data: the phase ADC has a resolution of 4 bits, and it provides the Arctan(Q/I) and RSSI detector output that is a 30-bit thermometric code, coded as a 5-bit value output in the RFFE modem. Since the AGC algorithm changes different attenuation stages during signal reception to keep the signal amplitude in the working dynamic range of the receiver, the final RSSI output is affected by the total attenuation level. To know the actual signal level of the input signal, the AGC attenuation level needs to be taken into consideration. For this purpose, the phase ADC data, RSSI data, and corresponding AGC attenuation level (4 bits) can be configured in such a way that these data samples that are available at RFFE interface level can be written to some registers to be accessed by the Arm Cortex-M33 processor or the DMA.

- I/Q data: It is possible that I and Q channel data can be sampled periodically after matched filtering, and placed into the RFFE interface.

5.2.3.1 7.2.6.3.1 Phase ADC and RSSI data streaming

Phase ADC and RSSI streaming, when required, needs to be enabled in SYSCTRL and RFFE. This is done both through setting the RF_DATA_STREAMING_DATA_STREAMING_DMA_EN_BUS field of the RF_DATA_STREAMING register in the RFFE for streaming data to the interface, and by setting the bit SYSCTRL_RFIF_CTRL_PHASE_ADC_STREAMING of the SYSCTRL_RFIF_CTRL register for sampling the data on the core side. To help data processing in SW, some flags can be added to the streamed data when a packet is received. When the access word of a packet is detected, a flag can be added to all data after that moment, and when the RX reception signal goes down in the RFFE, another flag can be added to all data after that time, indicating the end of the packet. In the middle of a Bluetooth Low energy packet reception, CTE can be included, and if it is configured to stream data for the CTE period, another flag can be added to packed data related to CTE duration. The RF front-end starts or stops streaming data with the RX activity. The different flags are reset when the RX activity starts, and are set as soon as the corresponding event is detected (sync word, CTE mode or packet end). Two consecutive 16-bit phase ADC data items are temporarily stored into a 32-bit data buffer. The latter is updated when a new phase ADC data pair is available, at the rate of 4 MHz and 8 MHz for 1 Mbps and 2

Mbps data rates over the air (respectively). The 32-bit data is available at the RFFE interface as shown in the "Packed Data Format of Phase ADC, RSSI, and Related Flags at the RFFE Interface" figure (Figure 5) where indexes 0 and 1 refer to the data at two consecutive time instants:

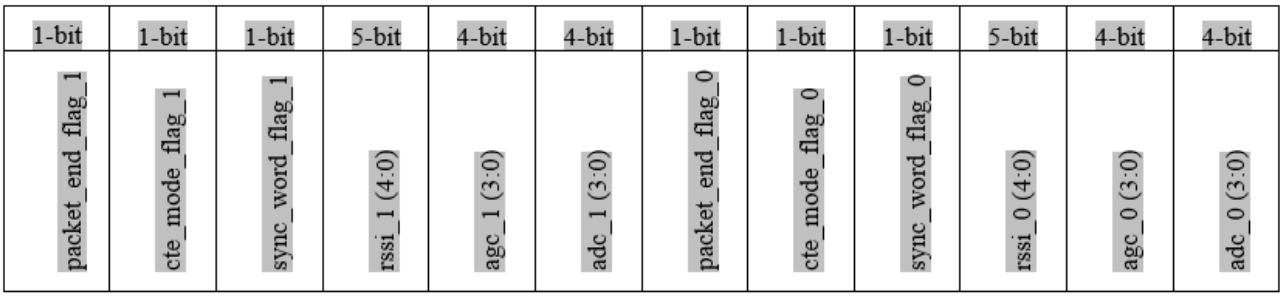


Figure 5. Packed Data Format of Phase ADC, RSSI, and Related Flags at the RFFE Interface

The phase ADC data-ready signal is set when new 32-bit data (i.e., two consecutive phase ADC data items) is available. Consequently, the last phase ADC data is lost if the amount of phase ADC data is an odd number. The system core clock needs to be at least four times faster than the 32-bit data rate (16 and 32 MHz for 1 and 2 Mbps Bluetooth Low Energy PHY rates). The sampled 32-bit data is mapped to several registers providing different data formatting (ADC here refers to phase ADC):

a. The `SYSCTRL_RFIF_PHASE_ADC_FLAGS` register provides the 32-bit data information of two consecutive phase ADC data with the flags.
b. The `SYSCTRL_RFIF_PHASE_ADC` register provides the 32-bit data information of two consecutive phase ADC data items without the flags.
c. The `SYSCTRL_RFIF_PHASE_ADC_0` register provides a 32-bit byte-aligned version of the phase ADC data 0 and flags (ADC, AGC, RSSI and flags are stored in separate bytes).
d. The `SYSCTRL_RFIF_PHASE_ADC_1` register provides a 32-bit byte-aligned version of the phase ADC data 1 and flags (ADC, AGC, RSSI and flags are stored in separate bytes).

The data-ready signal is used as a request source for the DMA to copy one item of data from the `SYSCTRL_RFIF_PHASE_ADC_FLAGS` or `SYSCTRL_RFIF_PHASE_ADC` registers, or two data items from the `SYSCTRL_RFIF_PHASE_ADC_0` and `SYSCTRL_RFIF_PHASE_ADC_1` registers.

A 16-bit counter is incremented by two when a data-ready rising edge is detected. The counter is reset to 0xFFFF by setting the `SYSCTRL_RFIF_CTRL_COUNTER_CLEAR` bit in the `SYSCTRL_RFIF_CTRL` register. The current counter state is mapped to the `SYSCTRL_RFIF_COUNTER` register. The counter state is stored in the following dedicated registers when the following events happens:

a. Stored in the `SYSCTRL_RFIF_SYNC_WORD` register when the sync word flag is set
b. Stored in the `SYSCTRL_RFIF_CTE_START` register when the CTE mode flag is set
c. Stored in the `SYSCTRL_RFIF_PACKET_END` register when the packet end flag is set

The LSB of the counter state stored in the above registers is reset if the event coincides with the phase ADC data 0. Each register is updated only once, when the first corresponding event is detected. Each register can be independently reset to 0xFFFF by setting the corresponding clear bit in the `SYSCTRL_RFIF_CTRL` register. It is the user application's responsibility to clear those registers when enabling the phase ADC data streaming and before every new RX event.

For testing purpose, dummy deterministic phase ADC data is generated by enabling the RF_AGC_ADVANCED_DEBUG_FAKE_IQ_SAMPLES bit of the RF_AGC_ADVANCED RFFE register.

5.2.3.2 I/Q data streaming

The I/Q data streaming is enabled by configuring the following two registers to enable this functionality in RFFE and RFIF:

- Setting the RF_DATA_STREAMING_DATA_STREAMING_EN_PERIODIC_SAMPLE_IQ bit of the RF_DATA_STREAMING register in the RFFE for streaming data to the interface
- Setting the SYSCTRL_RFIF_CTRL_IQ_STREAMING bit of the SYSCTRL_RFIF_CTRL register for sampling the data on the core side

The 16-bit I/Q data is the RF_IQFIFO_IQFIFO_IQ_DATA signal currently available at the RFFE interface, found in the RF_IQFIFO_IQFIFO_IQ_DATA field of the RF_IQFIFO register. The LSB byte is the I signal and the MSB byte is Q signal. I/Q data streaming is available in both CTE and non-CTE modes. The RF front-end starts streaming when the sync word is detected, and stops when the packet end is detected (in Bluetooth Low Energy mode) or when the RX activity stops (in packet handler mode). The behavior in CTE mode remains unchanged. The data-ready signal is set when a new 16-bit data item is available at the RFFE interface. For both 1 and 2 Mbps Bluetooth Low Energy PHY rates and in non-CTE mode, the IQ data rate is 4 MHz, and it is 250 Kbps, 500 KHz, or 1 MHz in CTE mode according to the CTE sampling rate (slot duration of CTE).

For testing purpose, dummy deterministic I/Q data is generated by enabling the RF_AGC_ADVANCED_DEBUG_FAKE_IQ_SAMPLES bit of the RF_AGC_ADVANCED RF front-end register.

5.3 RFFE System Resources

An overall and simplified digital diagram of the RFFE is shown in the "Simplified Digital Schematic Diagram" figure (Figure 6).

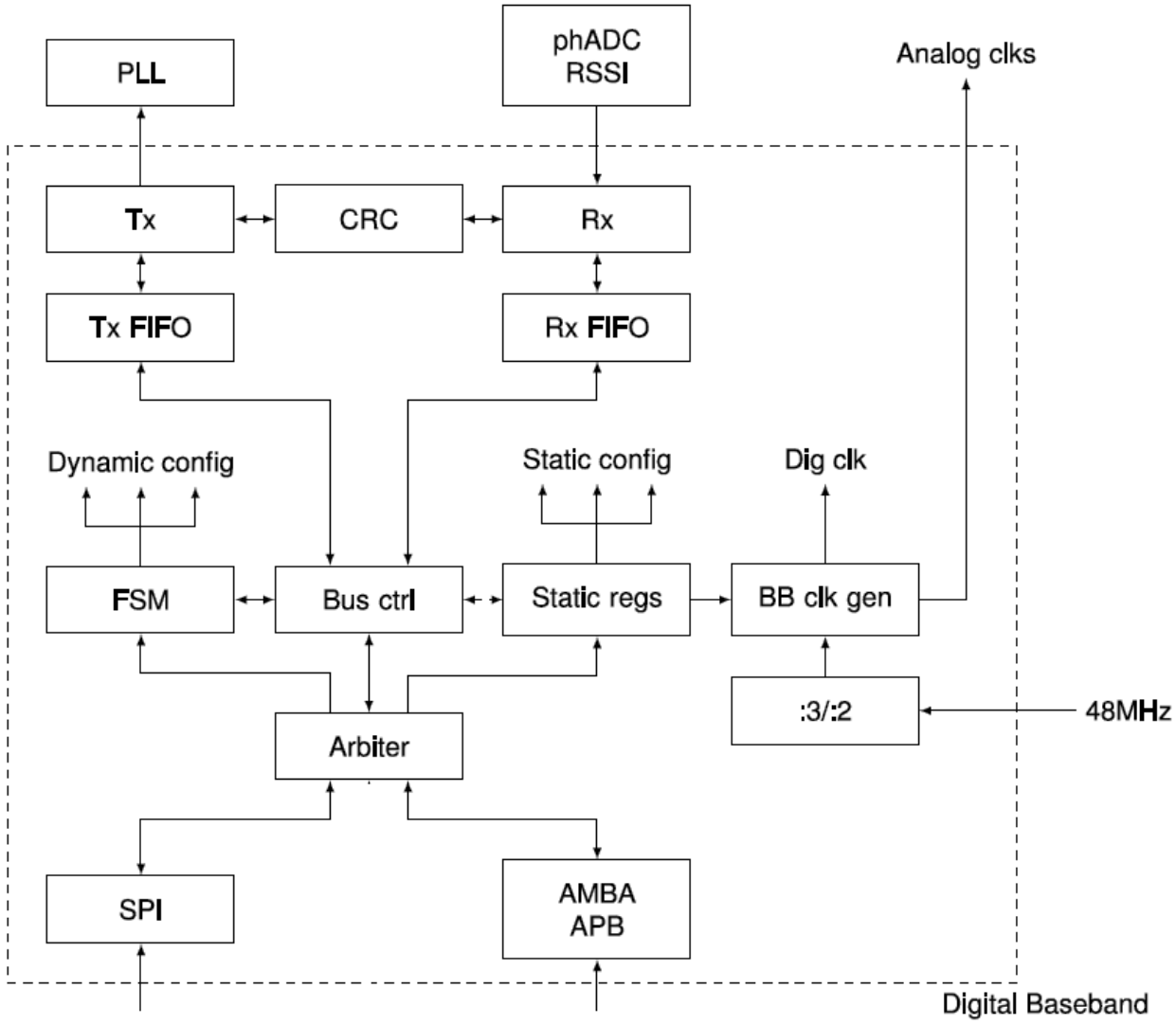


Figure 6. Simplified Digital Schematic Diagram

The required clocks are generated from the RFCLK.

Other key RFFE elements include:

FSM

This block contains the finite state machine (FSM) that controls the transceiver status. The FSM block provides the dynamic configuration to turn on the analog blocks, for sending or receiving data or packets.

BB clk gen block

This block generates the required clocks for the digital baseband (CK_BIT, CK_SYM, CK_OS, CK_FE, CK_TX). It also provides the signals for the clock generation of the analog blocks that require it (phADC and channel filter).

TX and RX FIFO

These blocks are a 128-byte FIFO. The TX FIFO receives data from the interface (APB or SPI) and sends it to the TX when requested. The RX FIFO receives data from the RX and sends it to the interface when requested.

TX

This block handles the TX process and it is composed of three sub-blocks: the serializer, the coder, and the modulator. The serializer takes the data from the TX FIFO and sends it to the coder in the order specified by the packet format. The coder takes these data and encodes them (Manchester, IEEE 802.15.4, etc.) Finally, the modulator processes the encoded data to be sent to the PLL for direct modulation.

RX

This block is the digital receiver of the system. It is composed of three sub-blocks: the demodulator, the decoder, and the deserializer. The demodulator takes the data from the phADC and the RSSI, and performs many operations: carrier recovery, matched filtering, clock recovery, RSSI averaging, ISI corrections, etc. The output is fed to the decoder, which is able to decode several types of encoding. Once decoded, the data is sent to the deserializer, which handles the packet format and fills the RX FIFO.

5.3.1 48 MHz Crystal Oscillator

As has been indicated, a 48 MHz clock is required for the RFFE to work as expected with clock accuracy conforming to the technology's standard specification. This clock is is generated using a 48 MHz crystal oscillator connected to the RSL15 XTAL pins, and is routed directly to the RFFE (independent of the RFCLK configuration). For more information, see "Clock Generation" on page 383.

For Bluetooth Low Energy, the clock accuracy must be better than 50 ppm during data transmission or reception.

The 48 MHz crystal oscillator can be trimmed through the `RF_REG2E_XTAL_TRIM_XTAL_TRIM` field of register `RF_REG2E`. Once it is written, it starts an algorithms that trims the 48 MHz frequency. This trim bit field is divided into 5 MSBs that provide coarse trimming, and 3 LSBs for fine trimming. The end of the trimming algorithm can be checked by reading the `RF_ANALOG_INFO_ANALOG_INFO_XTAL_FINISH` bit of the `RF_ANALOG_INFO` register.

This algorithm can be bypassed by setting the `RF_XTAL_CTRL_XTAL_CTRL_XTAL_CTRL_BYPASS` bit of the `RF_XTAL_CTRL` register to 1.

The `XTAL_TRIM` for a specific crystal can be calibrated by outputting a divided RF clock and trimming that clock to the expected division of 48 MHz. Trim settings on RSL15 are expected to be consistent for a population of crystals showing similar impedance characteristics.

NOTE: Custom calibration settings for `XTAL_TRIM` can be stored to NVR6, as described in the *RSL15 Firmware Reference Manual* Resource Usage table. If available, these values are loaded by the sample code provided in the SDK.

5.3.2 Radio Clocks

RFFE needs several clocks to work correctly:

- CK_OS: an oversampled clock compared to the symbol rate. Golden rule: the oversampling ratio is 8.
- CK_SYM: the symbol clock. This clock has the same frequency as the symbol rate.
- CK_BIT: the bit clock. This clock has the same frequency as the bit rate. If there is 1 bit/symbol, CK_BIT and CK_SYM have the same frequency; in case of 2 bits/symbol, CK_BIT has a frequency two times higher than the CK_SYM. This clock is generated by the coding blocks.
- CK_TX: this is the clock that is used in the TX interpolator. Basically the output to the PLL is clocked at this frequency.
- CK_FRONTEND: this is the clock that is used in the RX frontend processor. It is the same clock as the phADC. The RSSI has another clock but it can be programmed to use the same clock as the phADC.
- CK_FILTER: this clock is used by the channel filter in order to tune the center frequency and the bandwidth.

The frequencies of the clocks are specified by an integer value. The frequency is given by the system frequency divided by this value increased by 1. So a value of 3, when the system frequency is set to 16 MHz, gives 16 MHz/(3+1) = 4 MHz.

The CK_OS, CK_BIT, and CK_SYM clocks are controlled by the `RF_REG08_MOD_INFO_*_DR_M_*[4:0]` and `RF_REG0_MOD_INFO_*_SYMBOL_2BIT_*` fields of the `RF_REG08` register. The former field specifies the oversampling frequency CK_OS. The CK_TX is controlled by the `RF_REG01_MOD_TX_CK_TX_M[4:0]` field of the `RF_REG01` register. The CK_FRONTEND clock for the phADC is controlled by the `RF_FRONTEND_FRONTEND_DIV_PHADC[3:0]` field of the RF_FRONTEND register. The RSSI clock is controlled by `RF_REG01B_ANACLK_DIV_RSSI[3:0]` field of the `RF_REG1B` register. Alternatively, the RSSI can use the same clock as the phADC by setting the `RF_REG2D_RSSI_TUN_2_RSSI_ONE_CK_RSSI_PHADC` field of the `RF_REG2D` register. Finally, the CK_FILTER clock is controlled by the `RF_REG1B_ANACLK_DIV_FILT[3:0]` field of the `RF_REG1B` register. The CK_FILTER and CK_FRONTEND clocks are also fed to the analog blocks.

5.3.3 Registers

There are two kinds of registers used for the RF front-end: static registers, and dynamic registers. The static registers are used to configure the so-called static configuration of the transceiver. The dynamic registers are inside blocks that are working with the RFFE system clock. Accessing these two types of register is carried out in the same way for each type, and is transparent to the user except for the fact that the clock has to be enabled in the digital blocks. To activate this clock, the `RF_REG00_MODE_MODE[1:0]` field of the `RF_REG00` register has to be set to 10.

The RFFE has two banks of registers that contain the essential configuration needed to change modulation parameters on the fly. The registers that are banked are noted as "banked" in the register list. The register bank can be chosen with the `RF_REG08_BANK_BANK[0:1]` field of the `RF_REG08` register (using the `RF0_REG` notation). Bank 0 registers can be accessed directly using `RF1_REG` notation, while the bank 1 registers can be accessed directly using `RF2_REG` notation. To provide a more flexible configuration, the configuration bits that are common to TX and RX have been split into two registers. These configuration bits are `RF_REG08_MOD_INFO_*_DIV_CK_*`, `RF_REG08_MOD_INFO_*_SYMBOL_2BIT_*`, and `RF_REG08_MOD_INFO_*_DR_M_*[4:0]`, all from the `RF_REG08` register. The value to use is specified by the `RF_REG08_BANK_DATARATE_TX_NRX` field of the `RF_REG08` register.

The Bluetooth 5 standard specifies 4 data-rates: 1 Mbps, 2 Mbps, coded 125 kbps, and coded 500 kbps. The two coded data-rates used the 1Mbps modulation schemes and simply add a coding stage to the bitstream. RFFE is capable of supporting these 4 data-rates by configuring only the 1 Mbps and 2 Mbps rates. In this case, bank 0 needs to be loaded with the 1 Mbps configuration and all the parameters for the coded rates correctly set (except for the enable bits), and bank 1 with the 2 Mbps rate. The `RF_REG08_BANK_STD_BLE_RATES` bit of the `RF_REG08` register need to be

set to 1. At this point, selecting bank 2 corresponds to the coded 125 kbps data rate, and bank 3 corresponds to the 500 kbps data rate. Note that this is only valid for the TX; in RX mode, the packet rate indicator shows the data rate. So, in RX, the behaviors of bank 2 and 3 are the same.

5.3.4 FSM and Global Configuration

The behavior of the digital baseband radio is determined by the values in the following fields of the RF_REG00 register:

- RF_REG00_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
- RF_REG00_MODE_TX_NRX: if set to 1, it uses TX, otherwise RX.
- RF_REG00_MODE_EN_SERIALIZER: if set to 1, enables the serializer.
- RF_REG00_MODE_EN_DESERIALIZER: if set to 1, enables the deserializer.
- RF_REG00_MODE_EN_FSM: if set to 1, enables the radio FSM.
- RF_REG00_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.
- RF_REG00_MODE2_TESTMODE: set the output Test Mode.
- RF_REG00_MODE2_PSK_NFSK: if set to 1, the PSK Mode is selected; FSK is selected otherwise.
- RF_REG00_MODE2_DIFF_CODING: if set to 1, enables the differential coding/decoding.

RF_REG00_MODE_MODE[1:0] is the most important in the above list, as it determines the status of the digital baseband. RX or TX mode is determined by the RF_REG00_MODE_TX_NRX field; note that this is overridden when the system is in FSM Mode. The serializer and the deserializer can be activated with the RF_REG00_MODE_EN_DESERIALIZER and RF_REG00_MODE_EN_SERIALIZER fields. However, the most suitable working mode is the FSM Mode.

The modulation scheme can be chosen using the RF_REG00_MODE2_PSK_NFSK field of the RF_REG00 register. However, this is applied only on the RX side, with some limitations: for PSK modulation, only O-QPSK modulation is supported.

The radio is activated by setting bit 0, which is the RF_FSM_CTRL_FSM_MODE_MODE field of the RF_FSM_CTRL register, to 1. The FSM then activates the TX or the RX, depending on the value of bit 2, the RF_FSM_CTRL_FSM_MODE_TX_NRX bit of the same register. The sub-band algorithm can be activated at the same time, by setting bit 1, the RF_FSM_CTRL_FSM_MODE_TX_MODE bit of the same register, to 1. The calibration is performed before the packet's transmission/reception. Finally, the FSM can be reset at any time by setting the RF_FSM_CTRL_FSM_MODE_RESET bit in the same register to 1.

When FSM is enabled and set to start TX or RX, it uses some internal timers to enable different blocks required for TX or RX. For TX, sub-blocks PLL, DLL, DIG RF, and RF TX (analog) are turned on. For RX, sub-blocks PLL, RF RX, and DIG RX are on. The transitional states have to be maintained for the right amount of time, to avoid such problems as sending a message when the PLL is not yet stabilized. The time needed by each category of blocks can be specified in the RF_AGC_ATT2_TIMINGS_* fields of the RF_AGC_ATT2 register, and the RF_REG25_TIMINGS_* fields of the RF_REG25 register. The timings of the categories are specified by an integer value. The corresponding time is given by this value increased by 1 and multiplied by the time granularity. This is specified by the RF_AGC_ATT2_TIMINGS_1_T_GRANULARITY_TX[3:0] and RF_AGC_ATT2_TIMINGS_1_T_GRANULARITY_RX[3:0] fields of the RF_AGC_ATT2 registers. The time granularity is given by:

$$t_{\text{gran}_{\text{tx}}} = 2^{\text{t\_granularity\_tx}-2}\mu s,$$

$$t_{\text{gran}_{\text{rx}}} = 2^{\text{t\_granularity\_rx-2}}\mu s.$$

The timings can range from 0.25 µs to 262.1 ms, a range that is generally large enough. In the case of sub-band selection during startup, the sequence is slightly modified: since the PLL and/or the DLL need to run during sub-band selection, the activation time of these two blocks is modified in order for them to wake up before the beginning of the algorithm. In practice, the activation time of the sub-band selection algorithm is added to the activation time of the PLL and DLL.

5.3.4.1 TX and RX Start Sequence Example

To explain how long TX and RX are on from the time The `RF_FSM_CTRL_FSM_MODE_*` bits of the `RF_FSM_CTRL` register are written, two examples are provided.

- TX Example: A system is configured with the following bit field values of the `RF_AGC_ATT2` register:

`RF_AGC_ATT2_TIMINGS_1_T_GRANULARITY_TX` = 2

`RF_AGC_ATT2_TIMINGS_2_T_SUBBAND_TX` = 4

`RF_AGC_ATT2_TIMINGS_2_T_TX_RF` = 0

`RF_AGC_ATT2_TIMINGS_3_T_DLL` = 4

`RF_AGC_ATT2_TIMINGS_3_T_PLL_TX` = 2

If the FSM is activated with the `activate TX only` command—that is to say, without performing sub-band selection — during the startup sequence of the TX, the FSM timer is set to 4; this is the maximum value between *RF_AGC_ATT2_TIMINGS_2_t_tx_rf*, *RF_AGC_ATT2_TIMINGS_2_t_pll_tx*, and *RF_AGC_ATT2_TIMINGS_2_t_dll*. This timer decreases every 1 µs. If the sub-band selection is performed during the activation, the *RF_AGC_ATT2_TIMINGS_2_t_pll_tx* and *RF_AGC_ATT2_TIMINGS_2_t_dll* values are increased by the value of *RF_AGC_ATT2_TIMINGS_2_t_subband_tx*. As a result, the internal timer starts at the value of 8, which is the maximum value between *RF_AGC_ATT2_TIMINGS_2_t_pll* + *RF_AGC_ATT2_TIMINGS_2_t_subband_tx*, *RF_AGC_ATT2_TIMINGS_2_t_dll* + *RF_AGC_ATT2_TIMINGS_2_t_subband_tx*, and *RF_AGC_ATT2_TIMINGS_2_t_tx_rf*. All variables are fields in the `RF_AGC_ATT2_TIMINGS_2` register.

- RX Example: A system is configured with the following bit field values:

`RF_AGC_ATT2_TIMINGS_1_T_GRANULARITY_RX` in register `RF_AGC_ATT2` = 0x4

`RF_REG25_TIMINGS_4_T_PLL_RX` in register `RF_REG25` = 0x4

`RF_REG25_TIMINGS_5_T_RX_RF` = 0x1

`RF_REG25_TIMINGS_5_T_RX_BB` = 0x4

The granularity is 4 µs, meaning that the internal timer changes every 4 µs. In this case the maximum value is 4, so the internal timer is set to 4. Note that since the granularity of the counter is 4, the whole sequence takes 20 µs. Also

note that 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, that data is available only after the RX processing delay. Sub-band selection is activated, the behavior is similar to the TX case; the sub-band time is added to the PLL power-up time only.

5.3.4.2 RX Timeout

The RX timeout can be enabled by setting the RF_REG25_TIMEOUT_EN_RX_TIMEOUT bit of the RF_REG25 register to 1. The length of the timeout can be specified by the RF_REG25_TIMEOUT_T_RX_TIMEOUT[3:0] and RF_REG25_TIMEOUT_T_TIMEOUT_GR[2:0] fields of the same register. The definition of granularity is the same as for the timings, except that it is 16 times larger. So the timeout range is from 16 µs to 4.2 s. If the timeout expires, the FSM stays in the RX state as long as the pattern has been received; otherwise, it switches to the Idle or SuspendRX state, depending on the configuration of RF_REG00_MODE_NOT_TO_IDLE bit of the RF_REG00 register. The timeout can be calculated using the following equation:

$$t_{timeout} = (t\_rx\_timeout + 1) \times 2^{2 \times t\_timeout\_gr + 4} \ [\mu s].$$

5.3.4.3 MAC Timers

Modern protocols require precise and short timings between packets. In order to achieve these constraints, the RFFE provides an automatic timed re-activation of the TX or RX. For example, the RX can be activated after a packet's transmission in readiness to receive the acknowledgment. Another example is a broadcaster node that sends a packet every 150 µs.

This function can be activated by setting to 1 the RF_REG03_MAC_CONF_RX_MAC_ACT or the RF_REG03_MAC_CONF_TX_MAC_ACT fields of the RF_REG03 register. The first enables the re-activation of the radio after an RX mode, while the second is used in a TX mode. The mode to activate is specified by the RF_REG03_MAC_CONF_RX_MAC_TX_NRX and RF_REG03_MAC_CONF_RX_MAC_TX_NRX fields of the RF_REG03 register: set to 1, the next mode is the TX mode, and set to 0, the next mode is the RX mode. The time to wait after each mode is specified by the fields RF_PADS_89_RX_MAC_TIMER_RX_MAC_TIMER and RF_PADS_89_TX_MAC_TIMER_TX_MAC_TIMER of the RF_PADS_89 register. The time base is specified by the RF_REG03_MAC_CONF_MAC_TIMER_GR[1:0] field of the RF_REG03 register, and is equal to $2^{(2mac\_timer\_gr)}$ µs. Timer activation can be chosen using the RF_REG03_MAC_CONF_RX_MAC_START_NSTOP and RF_REG03_MAC_CONF_TX_MAC_START_NSTOP fields of the RF_REG03 register. If RF_REG03_MAC_CONF_RX_MAC_START_NSTOP is set to 0, the timer is activated at the packet end (when the RX is turned off), while if it is set to 1, the timer is activated at the sync word reception. If the RF_REG03_MAC_CONF_TX_MAC_START_NSTOP field is set to 0, the timer is activated at the end of packet transmission, while if it is set to 1, the timer is activated with the activation of packet transmission.

5.3.4.4 Protocol Timers

The protocol timer is more complex that the MAC timers, but it offers more possibilities and flexibility. In order to enable the protocol timer, the RF_PROT_TIMER_PROT_TIMER_CONF_END_PROT_TIMER bit of the RF_PROT_TIMER register needs to be set to 1. To understand how to configure the protocol timer, several aspects of it need to be understood and programmed correctly.

5.3.4.5 Time Stamps

The protocol timer allows your application to work with time stamp information. There are two time stamps. A time stamp is used to memorize the state of an internal timer when a particular event triggers it. The time stamps configurations are programmed through the RF_PROT_TIMER_PROT_TIMER_CONF_PT_T_STP_* fields of register RF_PROT_TIMER. The available configurations are:

- 000 No time stamp
- 001 Protocol timer trigger: a time stamp is created if a protocol timer command is sent.
- 01- Reserved for future use
- 100 TX Stop: a time stamp is created at the end of the TX phase.
- 101 RX Sync: a time stamp is created at the detection of the synchronization word.
- 110 RX Stop: a time stamp is created at the end of an RX phase (FSM radio transition from RX to Idle).
- 111 RX Received: a time stamp is created at the reception of a correct packet (no CRC errors).

5.3.4.6 Delta Time

It is possible to specify time intervals between a time stamp and the next event that needs to be generated; these intervals are called delta time. In the actual version there are 2 delta time intervals. A delta time is programmed through the `RF_PT_DELTA_0_PT_DELTA_TS_0_PT_DELTA_T0_MULT[13:0]` and `RF_PT_DELTA_0_PT_DELTA_TS_0_PT_DELTA_T0 [1:0]` fields of the `RF_PT_DELTA_0` register, and the `RF_PT_DELTA_1_PT_DELTA_TS_1_PT_DELTA_T1_MULT[13:0]` and `RF_PT_DELTA_1_PT_DELTA_TS_1_PT_DELTA_T1 [1:0]` fields of the `RF_PT_DELTA_1` register. The delta time value in µs is given by the formula:

$$\Delta t_n(\mu s) = 2^{2\times pt\_delta\_tn\_mult} pt\_delta\_tn$$

NOTE: The definition of time delta and time stamp are completely separated, i.e., there is no connection between time stamp 0 and time delta 0.

5.3.4.7 Commands

It is possible to send a command to the protocol timer in order to schedule an event. A protocol timer command requires specifying the following points:

- An operation
- A time stamp
- A delta time
- An action

The protocol timer command is specified by the `RF_REG50_PROT_TIMER_PT_CMD[7:0]` field of register `RF_REG50`. It is composed as shown in the "Protocol Timer Command" figure (Figure 7):

Bit	7	6	5	4	3	2	1	0
Description	Action		Delta time		Time stamp		Operation	

Figure 7. Protocol Timer Command

The protocol timer operations are:

- 00: Add an event to be scheduled from the actual time.
- 01: Add an event to be scheduled from the time stamp specified in the command.
- 10: Add a periodic event to be scheduled from the time stamp specified in the command.
- 11: Clear all schedules.

The periodic event schedule is like a normal event schedule, except that it is renewed every time that a new time stamp is generated. So it is possible to automatically retrigger an event and reproduce a periodic behavior.

The possible actions are essentially the activation of the TX or RX, with or without sub-band selection. The MSB of the command's Action field (as shown in Figure 7) specifies if the sub-band selection need to be performed, and the LSB specifies whether it is a TX or RX activation. In summary, the protocol timer actions are:

- 00: RX without sub-band selection
- 10: RX with sub-band selection
- 01: TX without sub-band selection
- 11: TX with sub-band selection

There are four possible parallel schedules. The internal timer has 24 bits, so its periodicity is 16.7 s. If an event scheduling is incorrectly programmed, it can happen 16 s later.

5.3.4.8 Protocol Timer, Delta Timer, and Command Example

As the first example, a packet needs to be sent 150 µs after the reception of a packet. Time stamp 0 and delta time 0 are used. Here are the steps in the example:

- Set the `RF_PROT_TIMER_PROT_TIMER_CONF_EN_PROT_TIMER` field of the `RF_PROT_TIMER` register to 1.
- First a time stamp is set on the RX Received signal: the `RF_PROT_TIMER_PROT_TIMER_CONF_PT_T_STP_0` field of the `RF_PROT_TIMER` register is set to "111".
- The `RF_PT_DELTA_0_PT_DELTA_TS_0_PT_DELTA_T0_MULT` field of the `RF_PT_DELTA_0` register = 00, since 150 µs is low enough
- For the `RF_PT_DELTA_0_PT_DELTA_TS_0_PT_DELTA_T0` field in the same register, we need to know the power-up time and the RX path delay. Here we suppose 25 µs for the TX power-up and 5 µs for the RX path delay, so `RF_PT_DELTA_0_PT_DELTA_TS_0_PT_DELTA_T0` = 120 = 0x78.
- Now the command can be sent with *pt_cmd* = 0xC1 :
    - Action = 11 : TX with subband selection
    - Delta time = 00
    - Time stamp = 00
    - Operation = 01: add to time stamp *n.*

In the second example, a packet needs to be sent after 1 ms and then another packet sent every 625 µs. Time stamp 0 is used for the periodic part, delta time 0 is used for the first delay of 1 ms, and delta time 1 is used for the periodic time.

- Set the `RF_PROT_TIMER_PROT_TIMER_CONF_EN_PROT_TIMER` field of the `RF_PROT_TIMER` register to 1.
- Set time-stamp 0 to be trigger by the protocol timer: the `RF_PROT_TIMER_PROT_TIMER_CONF_PT_T_STP_0` field of the `RF_PROT_TIMER` register = 001
- `RF_PT_DELTA_0_PT_DELTA_TS_0_PT_DELTA_T0_MULT` = `RF_PT_DELTA_1_PT_DELTA_TS_1_PT_DELTA_T1_MULT` = 0
- Set the first delta time to 1 ms: the `RF_PT_DELTA_0_PT_DELTA_TS_0_PT_DELTA_T0` field in the `RF_PT_DELTA_0` register = 0x3E8
- Set the second delta time to 625 µs: the `RF_PT_DELTA_1_PT_DELTA_TS_1_PT_DELTA_T1` in the `RF_PT_DELTA_1` register = 0x271
- Send a first schedule for the first packet: pt_cmd = 0xC0
    - Action = 11 : TX with subband selection
    - Delta time = 00
    - Time stamp = 00
    - Operation = 00: add to actual time.

- Then set the periodic timer. This operation is ideally performed after the first transmission, since a packet can be triggered in between due to the periodicity of the internal counter. For the periodic packet *pt_cmd* = 0xD2:
    - Action = 11 : TX with subband selection
    - Delta time = 01
    - Time stamp = 00
    - Operation = 10: add periodic event to time stamp *n.*

5.3.5 RFFE GPIO

The RFFE has 10 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 `RF_PADS_03`, `PAD_CONF_5` , and `RF_PADS_89` registers, through their `RF_PADS_*PAD_CONF*_PAD_*_CONF` fields. The values of these fields are associated with the following functions, as shown in the "RFFE GPIO Configuration" figure (Figure 8):

Value	Direction	Description
00000	In	Off. The GPIO is configured as an input, but the input value is not used.
00001	Out	IRQ_TX
00010	Out	IRQ_RXSTOP
00011	Out	IRQ_RECEIVED
00100	Out	IRQ_SYNC
00101	Out	IRQ_TXFIFO
00110	Out	IRQ_RXFIFO
00111	In	Tx start. The GPIO is configured as an input. A rising edge on this input will be notified to the FSM as a Tx start.
01000	Out	Rx data
01001	Out	Rx recovered clock
01010	Out	Tx data
01011	Out	Tx clock
01100	Out	Digital clock.
01101	In	Clock in (if implemented)
01110	Out	Drive a constant 0 to the GPIO
01111	N.A.	Test mode. Direction depends on the test mode.
10000	Out	Antenna_id (bit 0)
10001	Out	Antenna_id (bit 1)
10010	Out	Antenna_id (bit 2)
10011	Out	Antenna id (bit 3)
10100	Out	IQ sample (signal that indicates when to sample IQ)
10101	Out	Antenna switching (signal indicating the antenna switching)
10110	Out	Tx active
10111	Out	Rx active
11000	Out	Sync found
11001	Out	Clock phADC
11010	Out	Clock RSSI

Figure 8. RFFE GPIO Configuration

When Test Mode (0x0F) is selected, the meaning and purpose of signals associated with the configured GPIO and the direction of the GPIO depends on the Test Mode number written in the `RF_REG00_MODE2_TESTMODE` field of register `RF_REG00`. Test Mode options are shown in the "Testbus Modes" figure (Figure 9):

testmode	bit #9	bit #8	bit #7	bit #6	bit #5	bit #4	bit #3	bit #2	bit #1	bit #0
b00000	0	0	0	0	0	0	0	0	0	0
b00001	0	0	0	0	0	0	0	0	clock	data
b00010	0	rssi_bin(4)	rssi_bin(3)	rssi_bin(2)	rssi_bin(1)	rssi_bin(0)	phase(3)	phase(2)	phase(1)	phase(0)
b00011	0	phase(3)	phase(2)	phase(1)	phase(0)	rssi_bin(4)	rssi_bin(3)	rssi_bin(2)	rssi_bin(1)	rssi_bin(0)
b00100	0	0	0	0	sync	rssi_therm(4)	rssi_therm(3)	rssi_therm(2)	rssi_therm(1)	rssi_therm(0)
b00101	0	0	rx_data(7)	rx_data(6)	rx_data(5)	rx_data(4)	rx_data(3)	rx_data(2)	rx_data(1)	rx_data(0)
b00110	0	0	0	vreg_low (testchip only)	dll_locked	clk_dig_ready	clk_pll_ready	sb_ready	sb_hi	sb_lo
b00111	0	0	0	0	0	sb_ready.	sb_band(3)	sb_band(2)	sb_band(1)	sb_band(0)
b01000	0	0	0	0	0	tx_data_val (input)	tx_data (input)	rx_sync_found	rx_clock	rx_data
b01001	0	0	0	0	0	0	0	ck_dig	ck_pll_ref	ck_div_test
b01010	0	valid_data (input)	f_in(7) (input)	f_in(6) (input)	f_in(5) (input)	f_in(4) (input)	f_in(3) (input)	f_in(2) (input)	f_in(1) (input)	f_in(0) (input)
b01011	n.u.	n.u.	n.u.	n.u.	n.u.	n.u.	n.u.	n.u.	n.u.	n.u.
b01100	0	0	0	0	ck_div_test	xtal_finish	dll_locked	clk_pll_ready	xo_en_b_reg	ck_pll_ref
b01101	0	0	0	0	ck_pll_ref	xtal_finish	dll_locked	clk_pll_ready	xo_en_b_reg	ck_div_test
b01110	0	0	0	0	peak_det_filt(2)	peak_det_filt(1)	peak_det_filt(0)	peak_det(2)	peak_det(1)	peak_det(0)
b01111	0	0	0	0	0	0	agc_level(3)	agc_level(2)	agc_level(1)	agc_level(0)
b10000	0	0	0	0	0	tx_valid_data	tx_clock	tx_data	rx_clock (input)	rx_data (input)
other	n.u.	n.u.	n.u.	n.u.	n.u.	n.u.	n.u.	n.u.	n.u.	n.u.

Figure 9. Testbus Modes

The RSL15 Bluetooth stack communicates with the RFFE through dedicated hardware that sends and receives data and clocks in Test Mode equal to 0x08. To enable FSM and start TX or RX, it uses an internal SPI to communicate to the RFFE SPI for control purposes.

To route an RFFE GPIO, in addition configuring the desired GPIO in RFFE, one of the RSL15 GPIOs needs to be configured so that the system GPIO multiplexer can connect the RFFE GPIO to the SoC GPIO. For example, the following code can configue GPIO10 of RSL15 to route the RFFE TX active signal through connecting RFFE GPIO#5 to RSL15 GPIO10:

```
 RF0_PADS_47->PAD_CONF_2_PAD_5_CONF_BYTE = 0x16;
 SYS_GPIO_CONFIG(10, (GPIO_6X_DRIVE | GPIO_LPF_DISABLE | GPIO_WEAK_PULL_UP  | GPIO_MODE_
RF_GPIO5));
```

5.3.6 Interrupts

The RFFE 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 RF_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 RF_IRQ_STATUS or RF_DESER_STATUS registers.

RF_IRQ_RECEIVED_IRQ

Interrupt is raised when a packet is received and stored in the FIFO. The IRQ is cleared by reading the RF_IRQ_STATUS or the RF_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 RF_IRQ_STATUS or the RF_DESER_STATUS register.

RF_TXFIFO_IRQ

Interrupt is raised when the RF_FSM_CTRL_TXFIFO_STATUS_TXFIFO_NEAR_UNDERFLOW bit from the RF_FSM_CTRL register 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 RF_FSM_CTRL_RXFIFO_STATUS_RXFIFO_NEAR_OVERFLOW bit from the RF_FSM_CTRL register 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 RF_REG03_IRQ_CONF_IRQS_MASK field of the RF_REG03 register. For example, the IRQ RXSTOP status can be activated by setting bit 1 of the RF_REG03_IRQ_CONF_IRQS_MASK field to 1. By default, the IRQs are active high, but this behavior can be switched by writing 1 to the RF_REG03_IRQ_CONF_IRQ_ACTIVE_LOW field of the RF_REG03_ register.

In addition, the pad can also be configured to be in High-Z state when the IRQ is not active, by setting to 1 the RF_REG03_IRQ_CONF_IRQ_HIGH_Z field of the same register.

For example, to enable the IRQ RXSTOP status, the following code can be used:

```
RF->REG03 |= IRQ_CONF_IRQS_MASK_RX_STOP;
NVIC_EnableIRQ(RF_RXSTOP_IRQn);
```

5.3.7 FIFOs

There are two 128-byte FIFOs: one for the RX and one for the TX.

5.3.7.1 Accessing the FIFOs and FIFO Status

The two FIFOs' data are 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 to the RF_RXFIFO register corresponds to a pop. Reading the RF_TXFIFO register is possible, but there is no action on the FIFO (no pop implied). Writing to the RF_RXFIFO register is not possible and would have no impact, as it is read-only.

The status of each FIFO can be read in the RF_FSM_CTRL_RXFIFO_STATUS_* and RF_FSM_CTRL_TXFIFO_STATUS_* fields of the RF_FSM_CTRL register. There are indications regarding overflows, underflows, and whether the FIFO is empty or full. The RF_FSM_CTRL_*XFIFO_STATUS_*X_NEAR_OVERFLOW and RF_FSM_CTRL_*XFIFO_STATUS_X*_NEAR_UNDERFLOW fields of the RF_FSM_CTRL register are controlled by the value of the RF_REG03_FIFO_FIFO_THR[2:0] field in the RF_REG03 register for the RX and the value of the RF_REG03_FIFO_2_FIFO_THR_TX[2:0] in the same register for the TX. In the "Available FIFO Thresholds" figure (Figure 10), the thresholds for the value of RF_REG03_FIFO_FIFO_THR are given. The FIFOs can be flushed at any time by setting bit 0 of the 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).

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

Figure 10. Available FIFO Thresholds

If the bit RF_PACKET_EXTRA_PACKET_EXTRA_FIFO_REWIND of register RF_PACKET_EXTRA is set to 1, then after a TX transmission, the TX FIFO is reset to its previous stage. This function allows the sending of the same packet several times without the necessity of filling the FIFO every time.

During the reception of a message, many events can occur: a CRC error, a packet length error, etc. In all cases, the data has to be stored in the FIFO, at least temporarily. If a given 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 application starts to look at the FIFO's content before the end of the packet, it is important that the FIFO status is updated only at the end of the packet. The RX FIFO supports these two features. The automatic flush is controlled by the RF_REG03_FIFO_FIFO_FLUSH_ON_ADDR_ERR, RF_REG03_FIFO_FIFO_FLUSH_ON_PL_ERR, RF_REG03_FIFO_FIFO_FLUSH_ON_CRC_ERR, and RF_REG03_FIFO_FIFO_FLUSH_ON_OVFLW fields of the RF_REG03 register. The RF_REG03_FIFO_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.

If software writes to the RF_FSM_CTRL_TX_FIFO_STATUS_TX_FLUSH or RF_FSM_CTRL_RX_FIFO_STATUS_RX_FLUSH fields of the RF_FSM_CTRL register, the FIFO is flushed.

Additionally, if the RF_REG03_FIFO_2_RXFF_FLUSH_ON_START or RF_REG03_FIFO_2_TXFF_FLUSH_ON_START field in the RF_REG03 register is set, the RX FIFO is flushed when the RX is enabled (so a packet with an empty FIFO can be received), or the TX FIFO is flushed after the end of a packet transmission (to provide an empty FIFO).

5.4 PACKET HANDLING

The RFFE can be used as standalone radio where the Arm Cortex-M33 processor can interface and communicate with the RFFE through IRQs and TX/RX FIFOs. Test Mode in this case needs to be disabled (value 0x0) and the Bluetooth Low Energy stack of RSL15 and its related Bluetooth Low Energy hardware cannot be used. This standalone mode is called Packet Handling Mode.

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)

5.4.1 Packet Format

The packet handler supports several packet formats. Some of the possible packet formats are shown in the "Serializer-Supported Packet Formats (Partial List)" figure (Figure 11). The bytes shown in blue can be inserted automatically.

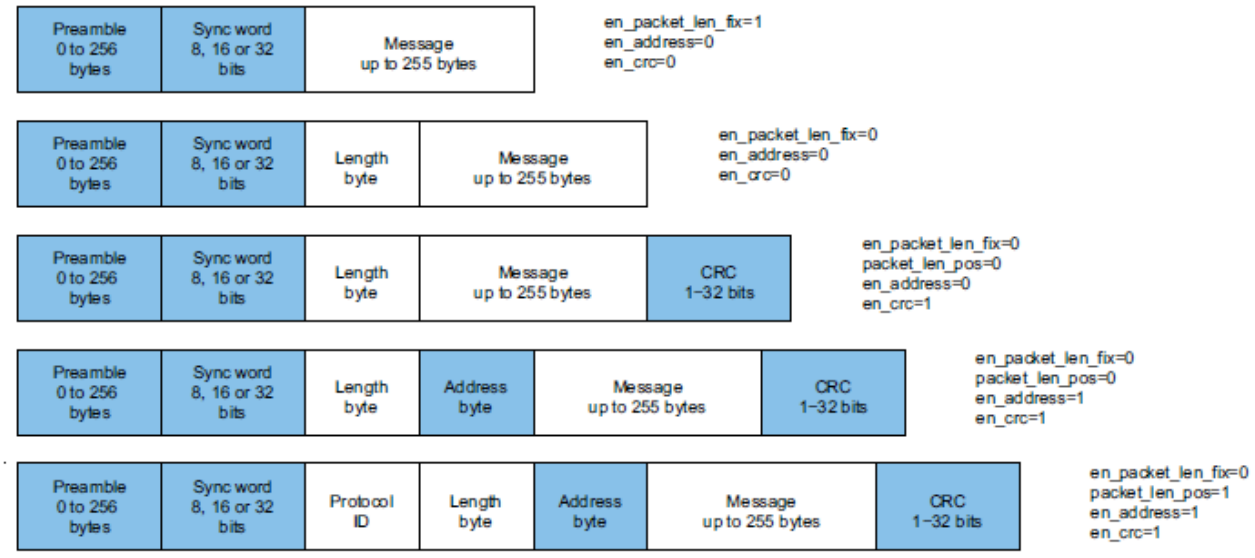


Figure 11. Serializer-Supported Packet Formats (Partial List)

5.4.1.1 Preamble

The preamble can be added automatically to the data, even if the packet structure is handled completely by the Arm Cortex-M33 processor or the firmware.. This feature can be turned on by setting the RF_PACKET_HANDLING_PACKET_HANDLING_EN_PREAMBLE field of the RF_PACKET_HANDLING register to 1. The length of the preamble in bytes is found in the RF_REG0C_PREAMBLE_LENGTH_PREAMBLE_LEN field of the RF_REG0C register increased by 1, and the preamble itself is located in the RF_PACKET_HANDLING register.

5.4.1.2 Pattern

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

5.4.1.3 Packet Length

If the RF_PACKET_HANDLING_PACKET_HANDLING_EN_PATTERN bit of the RF_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 RF_PACKET_HANDLING_PACKET_LENGTH_OPTS_EN_PACKET_LEN_FIX field of the RF_PACKET_HANDLING register must be set to 1. In such a case, the length of the packet is found in the RF_PACKET_HANDLING_PACKET_LENGTH_PACKET_LEN bit of the RF_PACKET_HANDLING register. In the case of a variable packet length, this is normally specified as one of the first bytes of the packet. The RF_PACKET_HANDLING_PACKET_LENGTH_OPTS_PACKET_LEN_POS field of the RF_PACKET_HANDLING 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 byte 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 RF_PACKET_HANDLING_PACKET_LENGTH_OPTS_PACKET_LEN_CORR field of the RF_PACKET_HANDLING 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 RF_PACKET_HANDLING_PACKET_LENGTH_OPTS_PACKET_LEN_CORR field has to be set to -2. the "Packet Length Definition and Associated Correction Examples" figure (Figure 12) shows the length definition of the packet.

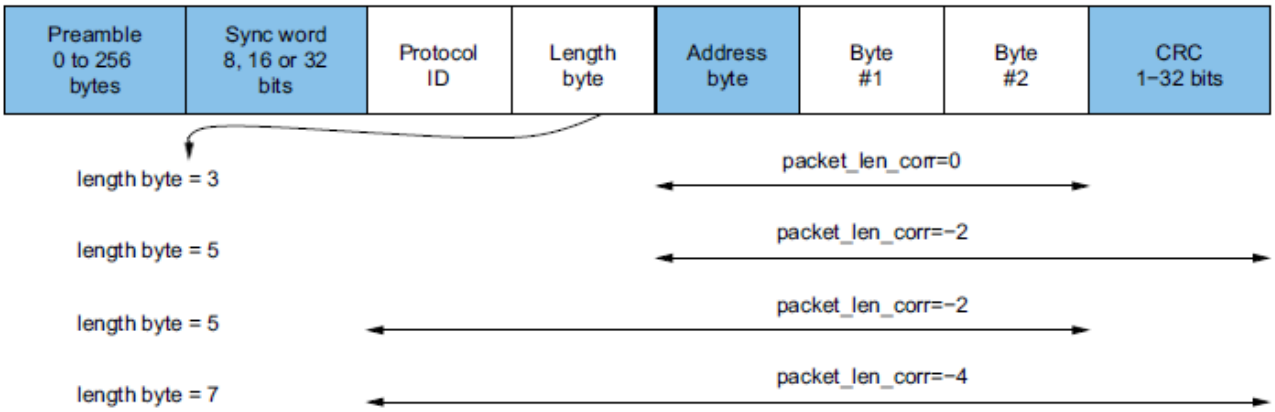


Figure 12. Packet Length Definition and Associated Correction Examples

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

5.4.1.4 Address

An address can be inserted automatically after the packet length, if the RF_REG0C_ADDRESS_CONF_EN_ADDRESS_TX field of the RF_REG0C register is set to 1. The address is given by the RF_REG0C_ADDRESS_ADDRESS bit of the RF_REG0C register. On the RX side, if the RF_REG0C_ADDRESS_CONF_EN_ADDRESS_RX field of the same register is set to 1, an address comparison is made. If the addresses do not match, an address error is generated by the deserializer.

A broadcast address can be specified in the RF_PACKET_EXTRA_ADDRESS_BROADCAST_ADDRESS_BR bit of the RF_PACKET_EXTRA register. The RX broadcast address reception can be enabled by setting the RF_REG0C_ADDRESS_CONF_EN_ADDRESS_RX_BR of the RF_REG0C register to 1; in such a case, the RX accepts the normal address and the broadcast address during reception. Confirmation of broadcast address reception is found in the RF_DESER_STATUS_DESER_STATUS_IS_ADDRESS_BR_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 RF_REG0C_ADDRESS_CONF_ADDRESS_LEN field of the RF_REG0C register.

5.4.1.5 Multi-Frame

If the RF_PACKET_HANDLING_PACKET_HANDLING_EN_MULTI_FRAME bit of the RF_PACKET_HANDLING 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 RF_PACKET_HANDLING_PACKET_HANDLING_EN_MULTI_FRAME bit is set to 1, Multi-Frame Mode is enabled.

5.4.1.6 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:

- The CCITT CRC16 polynomial x16+x12+x5+1 is given by 0x8810.
- The IBM CRC16 polynomial x16+x15+x2+1 is given by 0xC002.
- The Bluetooth LE CRC24 polynomial x24+x10+x9+x6+x4+x3+x+1 is given by 0x80032D.

This hardware CRC implementation works on the serialized stream, so the bit order depends on the value of the RF_PACKET_HANDLING_PACKET_HANDLING_LSB_FIRST bit in the RF_PACKET_HANDLING register. 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 RF_PACKET_HANDLING_PACKET_HANDLING_EN_CRC field of the RF_PACKET_HANDLING 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.

5.4.2 Bluetooth Low Energy Direct Test Mode

The Direct Test Mode (DTM) of the Bluetooth Low Energy standard specifies the packet format that needs to be used to test the PHY layer performance. For the TX mode, these packets can be provided by activating the Bluetooth Low Energy DTM mode; do this by setting the RF_PROT_TIMER_BLE_DTM_EN_BLE_DTM bit of register RF_PROT_TIMER to 1. Then the packet type and the packet length are defined by the the RF_PROT_TIMER_BLE_DTM_BLE_DTM_PKT_TYPE[3:0] and RF_PROT_TIMER_BLE_DTM_LEN[7:0] fields of the same register.

NOTE: When the BLE_DTM_EN_BLE_DTM Mode is enabled, the TX FIFO is completely bypassed.

5.4.3 Bluetooth Low-Energy Direction Finding

The RFFE supports the direction finding feature introduced in Bluetooth version 5.1. This feature allows calculation of the Angle of Departure (AoD) or the Angle of Arrival (AoA), based on the IQ samples taken during the constant tone extension (CTE) of the packet. The presence of the CTE is determined by the CTEinfo Present bit (CP bit, only present in the data connection packets). When this bit is present, the packet header is 1 byte longer, since it contains the CTEinfo byte that provides information on the CTE. The description here is used when the RFFE is used as standalone (Packet Handling Mode); otherwise, by default for the RSL15 Bluetooth Low Energy stack, the RFFE communicates to Bluetooth Low Energy hardware internally through dedicated signals.

5.4.3.1 TX Mode

In TX there are 3 possibilities to handle the CTEinfo.

- Put the CTEinfo byte in the FIFO and let the packet handler process the packet. This can be done by setting the RF_CTE_OPTS_CTE_OPTS_EN_READ_CP bit of register RF_ CTE_OPTS. In this case, the information stored in the FIFO at the CTEinfo position is saved and used for the CTE.
- Force the CTEinfo insertion in the packet. This functionality is activated by setting the RF_CTE_OPTS_CTE_OPTS_CP_INSERT bit of register RF_CTE_OPTS to 1. In this case, the packet in the FIFO needs to be a CTEinfo clear packet. The CP bit is automatically set to 1 by the packet handler, and the value in the RF_CTE_OPTS_CTE_OPTS_CTE_INFO[7:0] field of the RF_CTE_OPTS register is inserted at the correct spot.

- Force the CTE without modifiying the packet header (this mode is used in cases of CTE for the advertisement packets). This mode is activated with the RF_CTE_OPTS_CTE_OPTS_USE_CTE_WO_CP field of the RF_CTE_OPTS register. In this case, at the end of the packet, the CTE is inserted based on the information in the RF_CTE_OPTS_CTE_OPTS_CTE_INFO[7:0] field of the same register.

5.4.3.2 RX Mode

On the RX side, there are only two possibilities:

- Read the CP bit to parse the packet correctly for CTE information. This is done with the RF_CTE_OPTS_CTE_OPTS_EN_READ_CP bit of register RF_CTE_OPTS.
- Force the CTE analysis through the RF_CTE_OPTS_CTE_OPTS_USE_CTE_WO_CP bit from the RF_CTE_OPTS register. In this case the packet handler goes to the CTE phase after the CRC. The sampling information is provided by the RF_CTE_OPTS_CTE_OPTS_CTE_INFO[7:0] field of register RF_CTE_OPTS.

5.4.3.3 CTE Sampling

The CTE sampling is done automatically at the end of the packet, during the CTE phase. For AoD, the sampling period is determined by the CTEinfo. In case of AoA, it is possible to choose between 1 µs and 2 µs with the RF_CTE_OPTS_CTE_OPTS_DF_AOA_SLOT_TIME bit of register RF_CTE_OPTS. The IQ samples are available in the IQ_FIFO and can be read in register RF_IQFIFO. The samples are 8 bits wide and are stored I first. The number of available samples is stored in the RF_IQFIFO_STATUS_IQFIFO_COUNTIQFIFO_COUNT field of the RF_IQFIFO_STATUS register.

The sampling time and the antenna switching time have to be tuned to the RX and TX data path delays. The antenna switching delay in case of AoD is tuned through the RF_CTE_IF_CTE_CTRL_DELAY_TX_DF_DELAY_TX[9:0] field of the RF_CTE_IF register. The granularity of this delay is 62.5 ns. When this value is increased, the antenna switching happens at a later point. In the case of AoA, the antenna switching happens in RX and the delay is specified by the RF_CTE_CTRL_CTE_CTRL_DELAY_RX_DF_DELAY_SWITCH_RX[9:0] field of register RF_CTE_CTRL. This value represents the delay from the antenna to the deserializer. By increasing the value, the user anticipates the antenna switching. The sampling delay is controlled by the RF_CTE_CTRL_CTE_CTRL_DELAY_RX_DF_DELAY_SAMPLE_RX[9:0] field of the RF_CTE_CTRL register. This value represents the delay from the matched filter to the deserializer. By increasing this value, the user anticipates IQ samplings.

By its receiver architecture, the IQ signals doesn't have an amplitude information, since the phase is obtained through the phADC block. In order to provide an amplitude information, the RSSI instant value has to be combined with the phase to generate two real IQ values. This feature can be activated by setting the RF_CTE_OPTS_CTE_OPTS_CTE_AMPLl bit of the RF_CTE_OPTS register. Note that the quality of the I & Q samples can be slightly degraded if this feature is activated (noisy).

5.4.3.4 Antenna Switching Pattern

The antenna switching pattern is specified through a lookup table (LUT): the fields of this LUT specify the state of the antenna controls, and the LUT is parsed linearly. The RF_CTE_IF_ANTENNA_CONF_* fields in the RF_CTE_IF register and the RF_LUT_ANTENNA_ARRAY_*_LUT_ANTENNA_ARRAY_*_ANTENNA_* fields in the RF_LUT_ANTENNA_ARRAY_* registers are used to specify the LUT and the number of fields that are parsed inside it. So if the RF_CTE_IF_ANTENNA_CONF_ANT_LUT_M[3.0] field in the RF_CTE_IF register is set to 3 (meaning that a 4-antenna configuration pattern is being used), then the antenna is at the value of the RF_LUT_ANTENNA_ARRAY_1_LUT_ANTENNA_ARRAY_1_ANTENNA_0[3:0] field in the RF_LUT_ANTENNA_ARRAY_1 register during the packet and the reference period, after which it follows this pattern:

a1, a2, a3, a0, a1,...

Where a0 is RF_LUT_ANTENNA_ARRAY_1_LUT_ANTENNA_ARRAY_1_ANTENNA_1[3:0], a1 is RF_LUT_ANTENNA_ARRAY_1_LUT_ANTENNA_ARRAY_1_ANTENNA_2[3:0], a2 is RF_LUT_ANTENNA_ARRAY_1_LUT_ANTENNA_ARRAY_1_ANTENNA_1[3:0], and so on. The values inside the RF_LUT_ANTENNA_ARRAY_*_LUT_ANTENNA_ARRAY_*_ANTENNA_*[3:0] fields are completely arbitrary, so it is possible to specify one-shot coding as well as a binary code for the antenna switches. This is the basic antenna switching scheme. It is possible to code a separated antenna switching pattern from the rest of the packet. In this case, the RF_CTE_IF_ANTENNA_CONF_DF_IND_ANTENNA bit of register RF_CTE_IF needs to be set to 1. Then the packet is sent on RF_LUT_ANTENNA_ARRAY_1_LUT_ANTENNA_ARRAY_1_ANTENNA_0[3:0], the reference period of the CTE is on RF_LUT_ANTENNA_ARRAY_1_LUT_ANTENNA_ARRAY_1_ANTENNA_1[3:0], and the antennal switching pattern is:

a2, a3, a1, a2, a3,...

As an example, suppose that the RF_CTE_IF_ANTENNA_CONF_ANT_LUT_M field in the RF_CTE_IF register is set to 3. The switching pattern can also be separated from the reference by setting the RF_CTE_IF_ANTENNA_CONF_DF_INT_PATTERN field in the RF_CTE_IF register to 1. In this case, the switching pattern is as follows:

a2, a3, a2, a3, ...

If RF_CTE_IF_ANTENNA_CONF_DF_IND_PATTERN is set to 1 and RF_CTE_IF_ANTENNA_CONF_DF_IND_PATTERN_ANTENNA is set to 0, then the CTE reference period is on RF_LUT_ANTENNA_ARRAY_1_LUT_ANTENNA_ARRAY_1_ANTENNA_0[3:0] and the switching pattern is as follows:

a1, a2, a3, a1, a2,...

All possible switching schemes are represented in the "Possible Antenna Switching Schemes" figure (Figure 13).

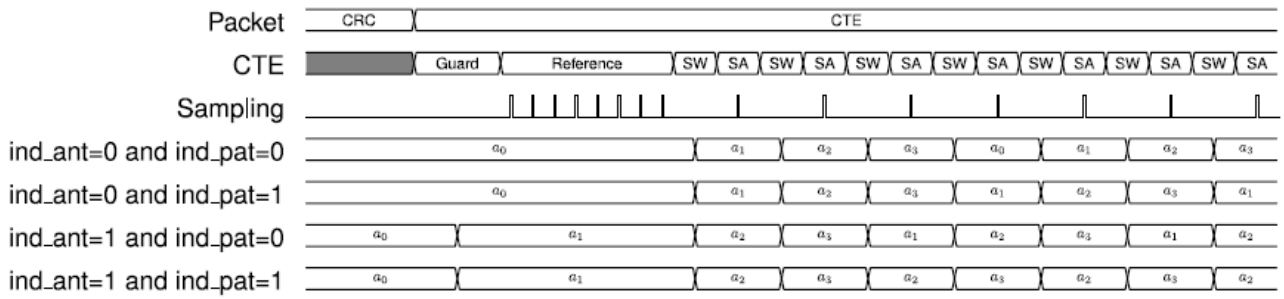


Figure 13. Possible Antenna Switching Schemes

5.4.4 Coding

The outgoing and incoming bit-streams often need to be encoded/decoded. There are several options available for both TX and RX.

5.4.4.1 Manchester Coding

Manchester encoding is available through the RF_CODING_CODING_EN_MANCHESTER bit of the RF_CODING_CODING register. This code is 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). This can be inverted by using the bit_invert configuration.

5.4.4.2 Data-Whitening

The data can be whitened by setting the RF_CODING_CODING_EN_DATAWHITE bit of the RF_CODING register. The whitening sequence is a PN9. Whitening is used to avoid long sequences of 0 or 1.

For the Bluetooth Low Energy standard, the LFSR used is a Galois LFSR7 with a specific reset status. This particular LFSR can be activated by setting the RF_REG00_DATAWHITE_BTLE_DW_BTLE bit of the RF_REG00 register; the RF_CODING_CODING_EN_DATAWHITE bit from the RF_CODING register also has to be set. In the RF_REG00 register, the RF_REG00_DATAWHITE_BTLE_DW_BTLE_RST[6:0] field specifies the reset status of the LFSR. If the Bluetooth Low Energy channels are used (bit RF_CODING_CHANNELS_2_EN_CHN_BLE of register RF_CODING), the reset value of the Galois LFSR7 does not need to be programmed.

5.4.4.3 IEEE 802.15.4 Bit to Chip Conversion

The IEEE 802.15.4 standard specifies conversion from a sequence of four bits to a transmitted sequence of 32 chips. This conversion can be activated by setting the RF_CODING_CODING_EN_802154_B2C bit of the RF_CODING register to 1. On the RX side when this bit is set, the chip sequence synchronization is based on sequences of 0000: this is mainly because the IEEE 802.15.4 standard preamble is composed of this sequence. However, pattern recognition works transparently.

5.4.4.4 New IEEE 802.15.4 Chip to Bit Decoder

A new version of the IEEE 802.15.4 chip decoder that works on the oversamples side has been implemented. It can be activated with the RF_AGC_LUT5_IEEE802154_OPTS_USE_OS_802154 bit of the RF_AGC_LUT5 register. This algorithm uses a correlator with an adaptive threshold to detect the correct sequences, and synchronizes the clock recovery in consequence. As for the old algorithm, the “0000” sequences are tracked to synchronize the symbol reception. Three parameters are used to track this sequence; they affect how the correlation peaks are detected. The parameters are the values of the RF_AGC_LUT5_IEEE802154_OPTS_CNT_LIM_802154[1:0], RF_AGC_LUT5_IEEE802154_OPTS_CNT_OK_INC_802154[2:0], and RF_AGC_LUT5_IEEE802154_OPTS_C2B_THR[2:0] fields of the RF_AGC_LUT5 register.

The correlation peaks of these sequences are set to 0 when there is a perfect match; so ideally, the threshold is set also to 0. However, due to the low SNR that can be achieved in the IEEE 802.15.4 mode, this might not be a good solution. The adaptive threshold follows the correlation peaks, i.e., when a correlation peak is lower than the threshold, the threshold assumes the correlation peak value. After $2^{(RF\_AGC\_LUT5\_IEEE802154\_OPTS\_CNT\_LIM\_802154+4)}$ samples, the threshold is increased by 1. A correlation peak is considered correct if its value is lower than the actual threshold added to the RF_AGC_LUT5_IEEE802154_OPTS_C2B_THR[2:0] value. If a peak is considered valid, then an internal 4-bit counter is increased by the value of the RF_AGC_LUT5_IEEE802154_OPTS_CNT_OK_INC_802154[2:0] field in the RF_AGC_LUT5 register. If the counter is larger or equal to eight, then the block is determined to be synchronized and starts to provide samples to the rest of the radio.

5.4.4.5 Linear to Frequency

The IEEE 802.15.4 standard specifies an O-QPSK modulation. This can be considered as linear, since the In phase and Quadrature phases are encoded directly. However, on the RSL15 RFFE there is a direct modulation, meaning that the frequency is encoded directly. To maintain the linear code and be able to modulate in the frequency domain, a linear to frequency coding is available. It can be activated by setting the RF_CODING_CODING_EN_802154_L2F bit of the RF_CODING register to 1.

NOTE: On the RX side, a phase ambiguity can arise. To prevent it, the RX correlators work on the frequency spreading sequences, and not on the phase sequences.

5.4.4.6 Convolutional Codes

In the RFFE, a forward error correction (FEC) is available that is realized with programmable convolutional codes. The decoding is performed with the classic Viterbi algorithm. These convolutional codes are well known and documented, so their principles are not detailed here.

It is possible to specify up to three codes, which gives a minimum rate of 1/3. The convolutional codes are specified in the fields RF_REG10_CONV_CODES_POLY_CC_POLY_0 and RF_REG10_CONV_CODES_POLY_CC_POLY_1 in the RF_REG10 register, by inserting the bits of the generator polynomials in reverse order. For example, if the code (x^3+x^2+x+1, x^3+x^2+1) (written as (17,15) since convolutional codes are generally written in octal) has to be coded, the code in binary is (0b1111,0b1101). So the RF_REG10_CONV_CODES_POLY_CC_POLY_0[4:0] field of the RF_REG10 register is 0xF (0b1111), and the RF_REG10_CONV_CODES_POLY_CC_POLY_1[4:1] field of the same register is 0xB (0b1011).

The convolutional codes can be activated by setting the RF_PACKET_EXTRA_CONV_CODES_CONF_EN_CONV_CODE bit of the RF_PACKET_EXTRA register to 1.

It is good practice to add some encoded bits at the end of the packet, to supply more information to the Viterbi decoder. The additional bits can be activated by the RF_PACKET_EXTRA_CONV_CODES_CONF_CC_EN_TX_STOP bit of the RF_PACKET_EXTRA register. The number of stop words is specified by the RF_PACKET_EXTRA_CONV_CODES_CONF_STOP_WORD_LEN[1:0] field of the RF_PACKET_EXTRA register.

On the decoder side, the memory length of the Viterbi algorithm is given by the RF_PACKET_EXTRA_CONV_CODES_CONF_CC_VITERBI_LEN[1:0] of the RF_PACKET_EXTRA register.

5.4.4.7 Puncturing Codes

Puncturing codes can be added to the convolutional codes: this increases the coding ratio at the cost of degraded correction capability. Puncturing codes are specified in reverse order, using the fields RF_REG10_CONV_CODES_PUNCT_CC_PUNCT_0[4:0] and RF_REG10_CONV_CODES_PUNCT_CC_PUNCT_1[4:0] of the RF_REG10 register. For example, if the code specified above has a puncturing code (1 1 1;1 0 0), then RF_REG10_CONV_CODES_PUNCT_CC_PUNCT_0 = 0x7, and RF_REG10_CONV_CODES_PUNCT_CC_PUNCT_1 = 0x1.

5.4.4.8 4FSK coding

The Bluetooth 5 specification introduces three additional data-rates: 2 Mbps, and coded 125 kbps and 500 kbps. While the former is a straightforward implementation in the RFFE, the coded data-rates (also called Bluetooth Low Energy Long Range (BLR)) require some additional coding/decoding features. These data-rates are implemented through a pattern mapping and a coded stream. For the coded stream, the convolutional codes (see Section 5.4.4.6 “Convolutional Codes” on page 168) need to be set correctly:

- RF_REG10_CONV_CODES_POLY_CC_POLY_0 field in register RF_REG10 = 0xF
- RF_REG10_CONV_CODES_POLY_CC_POLY_1 field in register RF_REG10 = 0xD
- RF_REG10_CONV_CODES_PUNCT_CC_PUNCT_0 field in register RF_REG10 = 0x1
- RF_REG10_CONV_CODES_PUNCT_CC_PUNCT_1 field in register RF_REG10 = 0x1

The BLR feature can be enabled by setting the RF_BLE_LR_BLE_LONG_RANGE_EN_BLR field of the RF_BLE_LR register/ On the TX side, it is required to specify whether a 500 kbps or a 125 kbps data rate needs to be sent, by setting the bit RF_BLE_LR_BLE_LONG_RANGE_BLR_500_N125 in the RF_BLE_LR register to 1 for a 500 kbps packet, or to 0 for a 125 kbps packet. This is enough for a TX packet to be sent correctly. The preamble is set correctly without any need to change the RF_PACKET_HANDLING_PREAMBLE_PREAMBLE field of the RF_PACKET_HANDLING register. The pattern mapping and convolutional codes are activated automatically with the RF_BLE_LR_BLE_LONG_RANGE_EN_BLR bit, so there is no need to change any other registers.

On the RX side, some additional parameters need to be set. The packet detection is performed with the following steps:

- preamble synchronization
- synchronization word detection
- decoding

Preamble synchronization is used to align the sampling time to the pattern mapping of the preamble , to demap it correctly. This is done through a correlation filter; its threshold is set with the RF_BLE_LR_BLR_PREAMBLE_BLE_PRE_THR[3:0] field of the RF_BLE_LR register. The correlation peak increases towards 0; if this field is set to 0, the maximum constraint is given to this filter. In the case of bad SNR, the preamble synchronization can be missed.

The synchronization word detection is made in two sets: first, a demapped but still coded signal is processed with a correlator filter to match the coded sync word. If there is a match, the coded sync word is fed to the Viterbi algorithm to be decoded, and then another correlator filter matches it against the uncoded syncword. For the first correlator, the maximum value for the correlation peak is 64 (since a coded 32-bit sync word is 64 bits long). Matching is best not performed on the value of 64, to take into account the possibility of low SNR; but the packet can still be decoded because of the Viterbi algorithm. So the peak detection can be selected by changing the threshold field RF_BLE_LR_BLR_SYNC_THRESHOLD_BLE_SYNC_THR[6:0] in the RF_BLE_LR register. If the value 64 is chosen, the maximum requirement is set to the correlator.

The packet is then decoded through the Viterbi algorithm and fed to the deserializer; after this, the packet is simply received. However, the Viterbi algorithm introduces a long latency that can be greater than the inter frame space (IFS) time specified by the Bluetooth Low Energy standard. To prevent this latency, the bit en_blr_flush in the register needs to be set to 1. This sends a signal to the Viterbi algorithm, when the last bit is received, to flush out its queue, since the path in the trellis is now known (termination bits of the BLR packet).

5.4.4.9 Bluetooth Low Energy Long Range

Since there is no universal 4FSK coding in RFFE, it is possible to change the mapping of the 4FSK encoding in order to meet almost any possible coding.

The four levels of the 4FSK modulation are generally represented as {-3;-1;+1;+3}. The two bits per symbol have to be coded into these four possible symbols. To reduce the number of errors, a Grey coding is generally used. It is possible to prove that one bit is coding the sign, and the other one the absolute amplitude of the symbol; this means that one bit codes if it is 1 or 3, and the other if it is positive or negative.

In the coder block the bits are fed in two paths named I and Q (this is a legacy name that is no longer related to the I and Q paths of the RF). 4FSK coding allows you to choose which signal is coding the sign (I or Q), and if the value is inverted or not. Separate codings for the TX and the RX are provided. 4FSK coding is activated by setting the RF_REG00_FOURFSK_CODING_EN_FOURFSK_CODING bit of the RF_REG00 register, and the different types of coding are chosen with the RF_REG00_FOURFSK_CODING_TX_FOURFSK_CODING[2:0] and the RF_REG00_FOURFSK_CODING_RX_FOURFSK_CODING[2:0] fields of the same register.

In the normal stream, the I and Q order is chosen with the the RF_CODING_CODING_EVEN_BEFORE_ODD bit of the RF_CODING register. If this bit is set to 1, the first bit to enter the data path is fed to the I path; otherwise the Q path is chosen. Then the RF_REG00_FOURFSK_CODING_*X_FOURFSK_CODING[0] bit from the RF_REG00 register specifies if the bit that determines the sign of the symbol is I or Q (0 => Q, 1 => I). If RF_REG00_FOURFSK_CODING_*X_FOURFSK_CODING[2] is 0, the sign is positive if the sign bit is 1, and negative if is 0. If RF_REG00_FOURFSK_CODING_*X_FOURFSK_CODING[2] is 1, the configuration is inverted. If RF_REG00_FOURFSK_CODING_*X_FOURFSK_

CODING[1] is 0 and the amplitude bit is 0, the absolute value of the symbol is 1; otherwise it is 3. If the RF_REG00_FOURFSK_CODING_*X_FOURFSK_CODING[1] is 1 and the amplitude bit is 0, the absolute value of the symbol is 3; otherwise, it is 1.

Examples: RF_CODING_CODING_EVEN_BEFORE_ODD = 1, RF_REG00_FOURFSK_CODING_*X_FOURFSK_CODING[2:0] = 0b011, and the message is 0b00111001 (LSB first). The first bit is 1 and, since RF_CODING_CODING_EVEN_BEFORE_ODD is set to 1, it is fed to the I path. So for the first symbol, the I path is 1 and the Q path is 0. RF_REG00_FOURFSK_CODING_*X_FOURFSK_CODING[0] is set to 1, so the sign of the symbol is given by the I bit and so the amplitude is controlled by the Q bit. The RF_REG00_FOURFSK_CODING_*X_FOURFSK_CODING[1] bit is set to 1, so the amplitude is inverted; since the amplitude bit (Q) is set to 0, the absolute amplitude is 3. Finally the RF_REG00_FOURFSK_CODING_*X_FOURFSK_CODING[2] bit is set to 0, so the sign is not inverted; since the sign bit (I) is set to 1, the sign is positive. In conclusion, the first two bits code the symbol +3. The others symbols are -1; -3, and 1.

The "4FSK Coding" figure (Figure 14) contains a summary of all available modes. Note that it is always a Grey coding.

IQ bits

*x_fourfsk_coding(2:0)	00	01	10	11
000	-1	1	-3	3
001	-1	-3	1	3
010	-3	3	-1	1
011	-3	-1	3	1
100	1	-1	3	-3
101	1	3	-1	-3
110	3	-3	1	-1
111	3	1	-3	-1

Figure 14. 4FSK Coding

5.4.4.10 Miscellaneous and Encoding Order

Several minor encoding options are available, including:

- Differential encoding: this can be activated by setting the RF_REG00_MODE2_DIFF_CODDING bit of the of the RF_REG00 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.

- Bit inversion: when the RF_CODING_CODING_BIT_INVERT bit of the RF_CODING register is set to 1, the encoding stream and the decoding stream are inverted.
- Bit order in quadrature modulations: this bit order (especially in O-QPSK) can be determined by the RF_CODING_CODING_EVEN_BEFORE_ODD, RF_CODING_CODING_OFFSET, and RF_CODING_CODING_I_NQ_DELAYED bits of the RF_CODING register.

The encoding order on the TX side is the following (on the RX side it is the opposite):

```
 Data whitening => Manchester => Bit2Chip => Lin2Freq => Bit invert => QuadMod => Diff
coding
```

5.5 Radio Configuration

5.5.1 Data Rate

The symbol rate is specified indirectly through the RF_REG08_MOD_INFO_*_DR_M_* field of the RF_REG08 register. The RF_REG08_MOD_INFO_*_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:

$$MOD\_INFO\_*\_DR\_M\_* = \frac{f_{sys}}{8DR} - 1$$

5.5.1.1 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 RF_REG11_TX_FRAC_CONF_TX_FRAC_NUM and RF_REG11_RX_FRAC_CONF_TX_FRAC_NUM fields in the RF_REG11 register:

$$dr\_eff = \frac{N}{DR} dr\_m$$

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 RF_REG10_FRAC_CONF_TX_FRAC_GAIN bit is specified in the register RF_REG10:

$$G = \frac{D}{2^{3+FRAC\_CONF\_TX\_FRAC\_GAIN}}$$

So if D = 10 and RF_REG10_FRAC_CONF_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 5.5.10 “Matched Filtering” on page 181).

5.5.2 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:

$$CENTER\_FREQ\_CENTER\_FREQUENCY = \frac{f_{RF} \times 2^{19(21)}}{f_{refTX}}$$

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. 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 RF_CENTER_FREQ_CENTER_FREQ_ADAPT_CFREQ bit of the RF_CENTER_FREQ register must be set to 1.

NOTE: The meaning of the RF_CENTER_FREQ_CENTER_FREQ_CENTER_FREQUENCY field (in the same register) 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:

$$CENTER\_FREQ\_CENTER\_FREQUENCY = \frac{f_{RF} \times 2^{21}}{f_{refTX}}$$

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.

5.5.3 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 RF_CODING_CHANNELS_2_EN_CHANNEL_SEL bit of the RF_CODING register to 1. The channel spacing is specified in the RF_CODING_CHANNELS_1_CHANNEL_SPACING_LO and RF_CODING_CHANNELS_2_CHANNEL_SPACING_HI fields of the RF_CODING register. The value in this field is given by the following formula, where f_{sp} is the channel spacing:

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

If channel 4 is wanted (that means 4 × f_{sp} from the central frequency), the value 0x4 must be written to the RF_REG08_CHANNEL_CHANNEL field of the RF_REG08 register.

5.5.4 Pulse Shape

The pulse shape is specified through the RF_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 RF_REG01_MOD_TX_PULSE_NSYM bit of the RF_REG01 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.

5.5.5 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 `RF_REG11` register (`RF_REG11_TX_MULT_TX_MULT_EXP` and `RF_REG11_TX_MULT_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 4FSK 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.

5.5.6 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 RF_REG01_MOD_TX_CK_TX_M(4:0) field of the RF_REG01 register. Its definition is:

$$F_{out} = \frac{f_{in}}{MOD\_TX\_CK\_TX\_M+1}$$

The interpolator is enabled using the RF_REG01_MOD_TX_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).

5.5.7 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 RFFE system frequency, and so it is either 16 MHz or 24 MHz, and K_f is the value of the field RF_REG_1B_ANACLK_DIV_FILT of the RF_REG_1B register:

$$F_{sc} = \frac{f_{sys}}{1+K_f}$$

The bias of the filter can be tuned via the RF_RSSI_CTRL_FILTER_BIAS_IQ_FI_BW and RF_RSSI_CTRL_FILTER_BIAS_IQ_FI_FC fields of the RF_RSSI_CTRL register.

An approximated configuration of the filter can be made by using the following equations:

$$f_c = \frac{f_{sc}}{2MHz}(238 \div 93.6 \times iq\_fi\_fc)[kHz]$$

$$bw = \frac{f_{sc}}{2MHz}(178 + 63.5 \times iq\_fi\_bw)[kHz]$$

For example, in Bluetooth Low Energy technology, RF_REG_1B_ANACLK_DIV_FILTDIV_FILT = 7, RF_RSSI_CTRL_FILTER_BIAS_IQ_FI_FC = 8, and RF_RSSI_CTRL_FILTER_BIAS_IQ_FI_BW = 14, so:

$$f_{sc} = \frac{\frac{16MHz}{8}}{2MHz}(238 + 93.6 \times 8) = 986.8kHz$$

$$bw = \frac{\frac{16MHz}{8}}{2MHz}(178 + 63.5 \times 14) = 1067kHz$$

5.5.8 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. See the "Simplified Block Diagram of the Resampler Block for the Phase" figure (Figure 15).

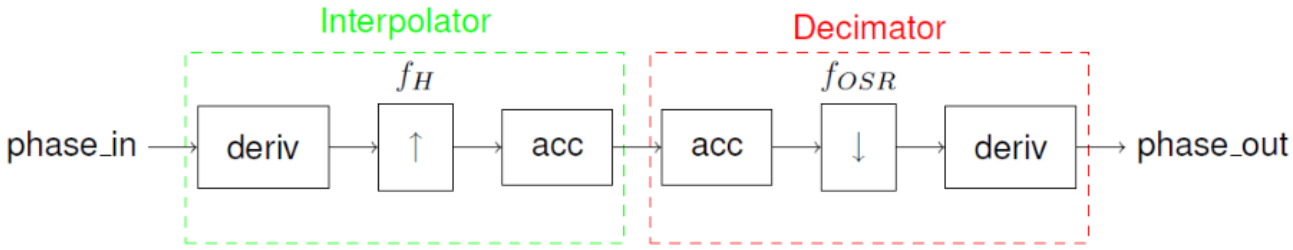


Figure 15. Simplified Block Diagram of the Resampler Block for the Phase

Note that in Figure 15, 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: `RF_FRONTEND_FRONTEND_EN_RESAMPLE_PHADC`, `RF_FRONTEND_FRONTEND_EN_RESAMPLE_RSSI`, `RF_FRONTEND_FRONTEND_DIV_PHADC`, `RF_FRONTEND_FRONTEND_RESAMPLE_RSSI_G1`, `RF_FRONTEND_FRONTEND_RESAMPLE_RSSI_G2`, `RF_FRONTEND_FRONTEND_RESAMPLE_PH_GAIN`, and `RF_FRONTEND_RX_IF_DIG_IF_DIG` field from the `RF_FRONTEND` register.

The incoming signals are clocked at the frequency given by the `RF_FRONTEND_FRONTEND_DIV_PHADC` field of the `RF_FRONTEND_FRONTEND` register. At the first stage they are upsampled at the f_{sys} frequency, giving a gain of `RF_FRONTEND_FRONTEND_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 `RF_REG08_MOD_INFO_*_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 RF_FRONTEND_RX_IF_DIG_IF_DIG field from the RF_FRONTEND register. 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:

$$RX\_IF\_DIG\_IF\_DIG = \frac{16 f_{IF}}{f_R}$$

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 RFFE baseband – that is to say, 16 MHz or 24 MHz – 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 RFFE 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 RF_FRONTEND_FRONTEND_RESAMPLE_PH_GAIN: this variable is an unsigned word. The gain is given by:

$$G_c = 2^{FRONTEND\_RESAMPLE\_PH\_GAIN - 7}$$

On the RSSI side, an additional gain is added after the interpolator. This gain is given by $2^{-RF\_FRONTEND\_FRONTEND\_RESAMPLE\_RSSI\_G1}$. A second gain is placed after the decimator, and its value is given by $2^{-RF\_FRONTEND\_FRONTEND\_RESAMPLE\_RSSI\_G2}$.

5.5.9 Carrier Recovery

5.5.9.1 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 RF_REG01_TAU_ROUGH_RECOV_TAU_ROUGH_RECOV field of the RF_REG01 register. Rough carrier recovery is enabled by setting the RF_REG01_CARRIER_RECOVERY_EN_ROUGH_RECOV bit of the RF_REG01 register to 1.

In this block, the IF is also canceled. The IF is specified in the RX_IF_DIG_IF_DIG field of the RX_IF_DIG register. This field is given by the following equation, where f_{IF} is the intermediate frequency and f_{sym} is the symbol rate:

$$RX\_IF\_DIG = \frac{128 f_{IF}}{f_{sym}}$$

5.5.9.2 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 RF_REG18_FSK_FCR_AMP1_FSK_FCR_AMP1 bit from the RF_REG18 register, and the RF_REG19_FSK_FCR_AMP2_FSK_FCR_AMP2 and RF_REG19_FSK_FCR_AMP3_FSK_FCR_AMP3 bits from the RF_REG19 register.

In the case of an FSK modulation with 1 bit per symbol, the three values are used to recreate the intersymbol interference (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 RF_REG19_FSK_FCR_AMP3_FSK_FCR_AMP3 in the case of a sequence [1;1;1], RF_REG19_FSK_FCR_AMP_FSK_FCR_AMP2 for [1;1;0] or [0;1;1], and RF_REG19_FSK_FCR_AMP1_FSK_FCR_AMP1 for [0;1;0]. Respectively, it is -RF_REG19_FSK_FCR_AMP3_FSK_FCR_AMP3 for [0;0;0], -RF_REG19_FSK_FCR_AMP2_FSK_FCR_AMP2 for [0;0;1] or [1;0;0], and -RF_REG19_FSK_FCR_AMP1_FSK_FCR_AMP1 for [1;0;1].

In the case of a 4FSK, 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 -RF_REG18_FSK_FCR_AMP_1_FSK_FCR_AMP1. It is RF_REG18_FSK_FCR_AMP_1_FSK_FCR_AMP1 when I and Q are [0;1], - RF_REG19_FSK_FCR_AMP3_FSK_FCR_AMP3 if I and Q are [1;0], and finally RF_REG19_FSK_FCR_AMP3_FSK_FCR_AMP3 with [1;1]. This configuration can be changed by changing the RF_REG00_FOURFSK_CODING_EN_FOURFSK_CODING field in the RF_REG00 register.

If rough carrier recovery is far from being completed, the $+\Delta f$ and the $-\Delta f$ 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 $-\Delta f$ to match the $+\Delta f$, 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 RF_REG01_TAU_PHASE_RECOV_TAU_PHASE_RECOV field of the RF_REG01 register. The block is enabled by setting the RF_REG01_CARRIER_RECOVERY_EN_FINE_RECOV bit of the RF_REG01 register.

5.5.9.3 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 RF_REG18_CARRIER_RECOVERY_EXTRA_FREQ_LIMIT_MAN and exponent RF_REG18_CARRIER_RECOVERY_EXTRA_FREQ_LIMIT_EXP fields of the RF_REG18_ 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:

$$f_1 = \frac{3\left(1+\frac{m}{8}\right)2^e f_{sym}}{2^4}$$

The mantissa needs to be specified as an unsigned value, while the exponent is signed. The carrier is searched for in the range:

$$[f_{IF} - f_1 : f_{IF} + f_1]$$

In practice, to calculate these values, this set of equations is used:

$$f_1 = \frac{3Kf_{sym}}{2^4}$$

$$K = \left(1 + \frac{m}{8}\right)2^e$$

Example

In the Bluetooth low energy technology standard, the carrier precision is given by ±150 kHz. This means that fl = 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).

5.5.9.4 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 RF_REG31_RSSI_DETECT_EN_DIFF_RSSI_DETECT bit of the RSSI_DETECT register, while absolute detection is activated using the RF_REG31_RSSI_DETECT_EN_ABS_RSSI_DETECT bit of the same register. The thresholds are specified in the RF_DEMOD_CTRL_RSSI_DETECT_DIFF_THR_RSSI_DETECT_DIFF_THR and RF_DEMOD_CTRL_RSSI_DETECT_ABS_THR_RSSI_DETECT_ABS_THR fields, respectively, of the RF_DEMOD_CTRL register. 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 value of the RF_REG19_RSSI_BANK_TAU_RSSI_FILTERING field of the RF_REG19 register. 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 FR_DEMOD_CTRL_DEMOD_CTRL_RSSI_DET_FILT bit of the register RF_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 t is equal to 1, 2, 4, or 6, depending on the value of RF_REG31_RSSI_DETECT_RSSI_DET_DIFF_LL of the RF_REG31 register:

$$Hz(z) = 1 - z^{-t}$$

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 RF_REG31_RSSI_DETECT_RSSI_DET_WAIT field of the RF_REG31 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 `RF_REG31_RSSI_DETECT_RSSI_DET_CR_LEN` field of the `RF_REG31` 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 average on the sync word too; in order to get rid of a biased sync word, sync word bias compensation needs to be switched on.

Sync Word Bias Compensation

After the RSSI ramp is detected, the state machine considers 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 might be a bias on the average. In order to compensate for this bias, the state machine corrects 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 `RF_DEMOD_CTRL_SYNC_WORD_CORR_EN_SYNC_WORD_CORR` bit of the `RF_DEMOD_CTRL` register to 1. The amplitude of the compensation is controlled by the `RF_DEMOD_CTRL_SYNC_WORD_CORR_SYNC_WORD_BIAS` field of the same register, and essentially depends on the modulation index. For a modulation index of 0.5, the value 0x4 needs to 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 `RF_DEMOD_CTRL_DEMOD_CTRL_EARLY_FINE_RECOV` bit of the `RF_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 might 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 arrives with a deterministic delay, so the state machine can tell precisely how long it has to slow down the system. This functionality is activated by setting the `RF_DEMOD_CTRL_DEMOD_CTRL_EN_PRE_SYNC` bit of the `RF_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 might 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 must not be activated if the carrier estimation is longer than eight symbols; otherwise it gives a false value, since the early estimation is made on data not yet corrected. The functionality is activated by the `RF_DEMOD_CTRL_DEMOD_CTRL_EN_MIN_MAX_MF` bit of the `RF_DEMOD_CTRL` register being set to 1.

Fast Clock Recovery

On the 4FSK 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 `RF_DEMOD_CTRL_DEMOD_CTRL_EN_FAST_CLK_RECOV` bit of the register `RF_DEMOD_CTRL` to 1.

5.5.9.5 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 `RF_DEMOD_CTRL_DEMOD_CTRL_EN_DELLINE_SYNC_DET` bit of the register `RF_DEMOD_CTRL` to 1.

When this mode is activated, it is recommended that the sync word correction bias be activated by setting the `RF_DEMOD_CTRL_SYNC_WORD_CORR_EN_SYNC_WORD_CORR` bit in the `RF_DEMOD_CTRL` register to 1. The `RF_DEMOD_CTRL_SYNC_WORD_CORR_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 looks for a peak. The precision of this peak search can be controlled by `RF_REG18_DELAY_LINE_CONF_MAX_ERR_IN_DL_SYNC` field in the `RF_REG18` register. In practice, it defines the maximum number of errors in the sync word, from 0 to 3.

The correction for carrier recovery is 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 `RF_REG18_DELAY_LINE_CONF_EN_SYNC_OK_DELAY_LINE` bit of the `RF_REG18` register 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.

5.5.10 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 RF_RX_PULSE_SHAPE_RX_PULSE_SHAPE_RX_COEF* fields of the RF_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 RF_REG11_FILTER_GAIN_GAIN_M and RF_REG11_FILTER_GAIN_GAIN_E fields of the RF_REG11 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:

$$G = \left(1 + \frac{m}{8}\right) 2^e$$

5.5.11 Clock and Data-Rate Recovery

This block recovers the clock of the signal, and its data rate (inside a specific range): in practice it generates 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 is 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 constant fields RF_REF_02_TAU_CLK_RECOV_TAU_CLK_RECOV and RF_REF_02_TAU_DATARATE_RECOV_TAU_DATARATE_RECOV, the data-rate recover limit field RF_REG02_DATARATE_OFFSET_DR_LIMIT, and the data-rate offset field RF_REG02_DATARATE_OFFSET_DATARATE_OFFSET, all from the RF_REG02 register. The time constants determine the time that the block needs in order to achieve clock or data-rate recovery, respectively.

RF_REG02_DATARATE_OFFSET_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.

RF_REG02_DATARATE_OFFSET_DR_LIMIT specifies the range of data-rate recovery. The values are given in the "Data-Rate Recovery Search Range" table (Table 4).

Table 4. 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 the "Data-Rate Recovery Issue" figure (Figure 16).

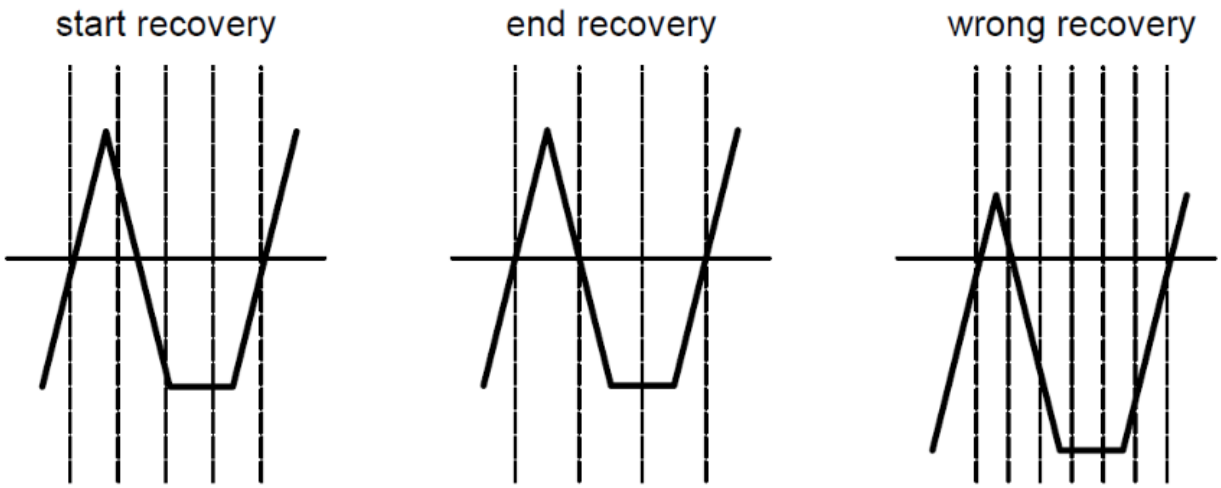


Figure 16. Data-Rate Recovery Issue

As the "Data-Rate Recovery Issue" figure (Figure 16) 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 the "Data-Rate Recovery Issue" figure (Figure 16), 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 4FSK modulation: it is always based on the zero crossing detection. However, because of the 4FSK modulation, the eye diagram horizontal opening is narrower than that of a 2-FSK modulation, as can be seen in "Eye Diagram for 2FSK Modulation and 4FSK Modulation" figure (Figure 17).

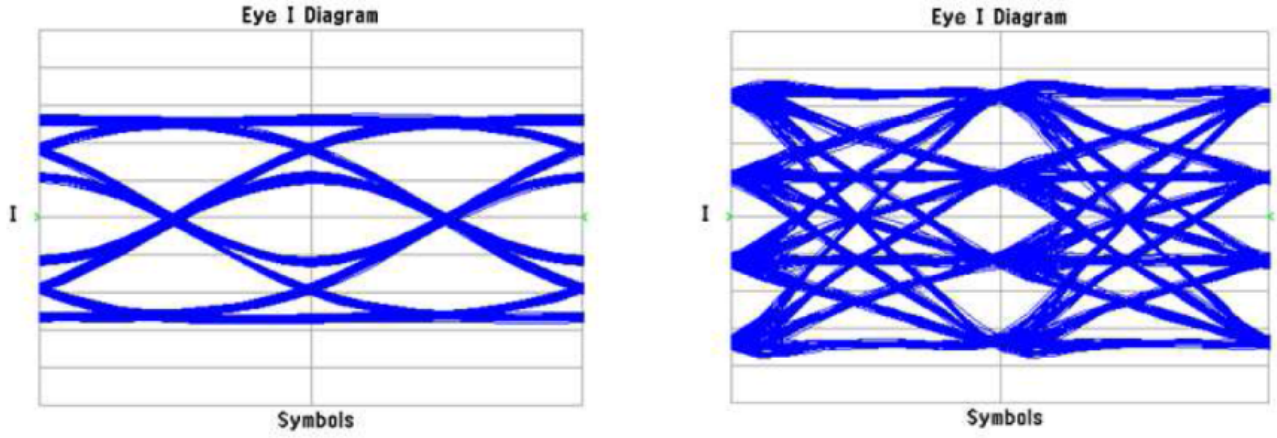


Figure 17. Eye Diagram for 2FSK Modulation and 4FSK Modulation

This means that the time constant for the 4FSK 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 might 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.

5.5.12 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.

5.5.12.1 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 `RF_REG19_DECISION_EN_VITERBI_GFSK` bit of the `RF_REG19` register. The amplitudes of the expected signal are specified in the `RF_REG18_FSK_FCR_AMP_1_FSK_FCR_AMP1` field in the `RF_REG18` register, and the `RF_REG19_FSK_FCR_AMP_2_FSK_FCR_AMP2` and `RF_REG19_FSK_FCR_AMP_3_FSK_FCR_AMP3` fields in the `RF_REG19` register. The path length of the estimator is specified by the `RF_REG19_DECISION_VITERBI_LEN` field of the `RF_REG19` register.

5.5.13 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 `RF_REG19_RSSI_BANK_TAU_RSSI_FILTERING` field in the `RF_REG19` register. A fast mode can also be made available, by setting the `RF_REG19_RSSI_BANK_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 also evaluate a minimum and a maximum value of the RSSI. These RSSI values are available in the `RF_REG45_RSSI_AVG_RSSI_AVG` bit from the `RF_REG45` register and the `RF_RSSI_MIN_MAX_RSSI_MAX_RSSI_MAX` and `RF_RSSI_MIN_MAX_RSSI_MIN_RSSI_MIN` bits from the `RF_REG45` register. 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 RF_REG1D_AGC_THR_HIGH_AGC_THR_HIGH field from the RF_REG1D register, the counter increases; if it is lower than the value specified in the RF_REG1D_AGC_THR_LOW_AGC_THR_LOW bit from the same register, the counter decreases. The counter has three bits and starts at 0. Its maximum value is fixed by the value of the RF_REG1D_ATT_CTRL_ATT_CTRL_MAX field of the RF_REG1D register. The value of the counter is then used as the input of the AGC look-up table specified in the RF_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 RF_AGC_LUT_* registers 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, 6 dB of attenuation
  - 10: 4 kΩ, 12 dB of attenuation
  - 11: 2 kΩ, 18 dB 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 the AGC step. To do so, it needs to know the attenuation between every AGC level. These attenuations can be specified by the RF_AGC_ATT*_RF_AGC_ATT_*_RF_AGC_ATT* fields of the RF_AGC_ATT* registers. The field RF_AGC_ATT1_AGC_ATT_1_AGC_ATT_01 from the RF_AGC_ATT1 register, 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. To activate the mode of RSSI correction, the RF_RSSI_CTRL_RSSI_CTRL_AGC_MODE bit of the register RF_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 RF_RSSI_CTRL_RSSI_CTRL_AGC_WAIT field of the RF_RSSI_CTRL register.

The AGC algorithm can be switched off by setting the RF_RSSI_CTRL_RSSI_CTRL_BYPASS_AGC bit of the RF_RSSI_CTRL register to 1. The RX chain attenuation is then determined by the RF_REG1D_ATT_CTRL_SET_RX_ATT_CTRL field of the RF_REG1D register.

5.5.13.1 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 RF_REG2D_CTRL_RX_USE_PEAK_DETECTOR of the RF_REG2D register needs to be set to 1. The AGC algorithm uses the peak detector information if the RF_REG1D_AGC_PEAK_DET_EN_AGC_PEAK bit of the RF_REG1D 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 RF_REG1D_AGC_PEAK_DET_PEAK_DET_THR_LOW field and the RF_REG1D_AGC_PEAK_DET_PEAK_DET_THR_HIGH bit of the RF_REG1D register. In the same register there is also the RF_REG1D_AGC_PEAK_DET_PEAK_DET_TAU field, which defines a time constant for the filtering of the peak detector signals.

- RF_REG1D_AGC_PEAK_DET_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
- RF_REG1D_AGC_PEAK_DET_PEAK_DET_THR_HIGH — peak detector high threshold (AGC increment indicator)
  - 0: above level 2
  - 1: above level 3

5.5.13.2 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 RF_REG1D_AGC_THR_HIGH_AGC_THR_HIGH.

rssi_under
: Set to 1 if the RSSI value is smaller than RF_REG1D_AGC_THR_LOW_AGC_THR_LOW.

peak_over
: Set to 1 if the peak detector output is larger than RF_REG1D_AGC_PEAK_DET_PEAK_DET_THR_HIGH.

peak_under
: Set to 1 if the peak detector output is smaller than RF_REG1D_AGC_PEAK_DET_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 RF_REG1D_AGC_THR_HIGH_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.

5.6 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 RSL15 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 2 Mbps:

1. Load the *hci* hex file into flash memory, and then reset the RSL15 Evaluation and Development Board. The RSL15 RF Tools offer two menus where RSL15 can be put in CW mode, RX or TX, at the desired frequency. For TX, output power can be also set from RF Tools. If CW Mode needs to be handled in FW, the following settings need to be applied:
    a. Set the `RF_REG00_MODE2_TESTMODE` bit from the `RF_REG00` register to 0:
    b. Select register bank 0 :
    c. `RF0_REG08->BANK_BYTE = 0x4;`
2. Set the `RF_CENTER_FREQ` register to the frequency you desire, and disable Bluetooth Low Energy channel selection. It is possible specify the desired Bluetooth Low Energy channel number whiteout setting the center frequency and disabling channel selection. Find the required frequency using the equation in the following code:

```
center_freq = (((float)freq* (0x1 << 21)) / 144000000.0); /* frequency is in Hz */
RF->CENTER_FREQ = (CENTER_FREQ_ADAPT_CFREQ_ENABLE | center_freq);
RF->CODING &= (~CHANNELS_2_EN_CHANNEL_SEL_ENABLE);
RF->CODING &= (~CHANNELS_2_EN_CHN_BLE_ENABLE);
```

3. `RF0_REG08->BANK_BYTE` is set to 0, for 1 Mbps, by default. For 2 Mbps, set `RF0_REG08->BANK_BYTE` to 1.6.
4. To configure for RX mode, use `RF0_FSM_CTRL->FSM_MODE_BYTE = (FSM_MODE_MODE_CAL_PLL_TXRX | FSM_MODE_RX_NTX);`.
5. To configure for TX mode, use `RF0_FSM_CTRL->FSM_MODE_BYTE = (FSM_MODE_MODE_CAL_PLL_TXRX | FSM_MODE_TX_NRX);`.
6. To configure for idle mode (disable RF), set `RF_REG30->FSM_MODE_BYTE` to 8.

When working in CW configuration, use your own preferred settings for VDDRF, VDDPA enabling, VCC, VDDPA, DCCLK, the charge pump clock, and buck enabling.

5.7 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 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* sample application for details on how to use DTM.

Another option for configuring RSL15 in DTM mode is to develop the `GAP API DTM` command in embedded firmware, and then the application can send `DTM GAP` commands to the Bluetooth Low Energy stack whenever DTM is required. For example, DTM can be requested by an external device through any interface such as UART, SPI, or I^2C with a custom protocol, or it can be commanded by a Bluetooth Low Energy device over a Bluetooth Low Energy link. DTM is useful for final product testing in manufacturing or the production line.

IMPORTANT: All unrelated DTM activities must first be deleted for direct test mode to work properly.

5.8 Power Amplifier and Ramp Up/Down Sequence

The power amplifier (PA) bias supply is provided by VDDPA or VDDRF. The output power provided by the PA is determined by the voltage value of the selected regulator and the RF_REG1A_PA_PWR_PA_PWR field of the RF_REG1A register. The maxim output power is achieved when the RF_REG1A_PA_PWR_PA_PWR field is set to 0x0C. The effect of RF_REG1A_PA_PWR_PA_PWR on output power is shown in the "Relative Output Power" table (Table 5) relative to the maximum output power corresponding to the value 0x0C.

Table 5. Relative Output Power

RF_REG1A_PA_PWR_PA_PWR value	Relative output power (dB)
0x1D	-41
0x1E	-27 to -41
0x1F	-31
0x00	-18.5
0x01	-17
0x02	-15.2
0x03	-13.7
0x04	-12.2
0x05	-10.7
0x06	-9.2
0x07	-7.8
0x08	-6.3
0x09	-4.9
0x0A	-3.3
0x0B	-2
0x0C	Maximum

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 and additional ultra-low power output mode. The PA ramp-up is activated by setting the RF_REG31_PA_RAMPUP_EN_PA_RAMPUP bit of the RF_REG31 register to 1. The ramp-down is activated by setting the RF_REG31_PA_RAMPUP_EN_PA_RAMPDOWN bit of the same register to 0. 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 the RF_REG31_PA_RAMPUP_DEL_PA_RAMPUP[2:0] field of the RF_REG31 register waits for the PA to be on. The values of the delay are (in µs units):

- 0b000: 0.25
- 0b001: 0.31
- 0b010: 0.5
- 0b011: 0.81
- 0b100: 1.5
- 0b101: 2.1

- 0b110: 2.8
- 0b111: 4.1

The steepness of the ramp-up is controlled by the parameter in the RF_REG31_PA_RAMPUP_TAU_PA_RAMPUP[1:0] field of the RF_REG31 register. The ramp-up duration depends on the final value of the PA back-off field RF_REG1A_PA_PWR_PA_PWR from the RF_REG1A register. In the case of the maximum final value of the PA back-off and starting from step 0 , the conversion listfollows (in µs units):

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

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

The general behavior of the PA ramp-up is shown in the "Ramp-up sequence" figure (Figure 18).

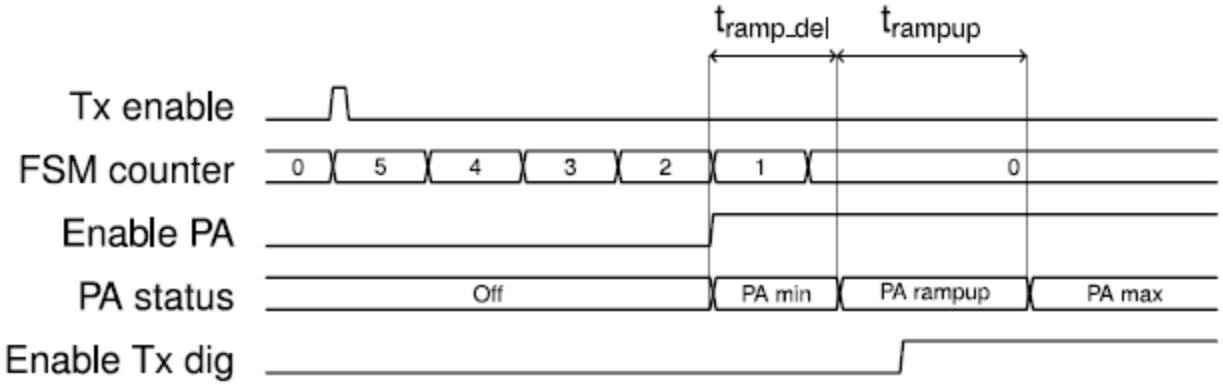


Figure 18. Ramp-up sequence

5.9 RF Front-End Registers

NOTE: RF0_REG* registers access the banked registers according to the RF_REG08_BANK_BANK register configuration. RF1_REG* registers access the banked registers 0, and RF2_REG* registers access the banked registers 1, independently of the RF_REG08_BANK_BANK register configuration.

Register Name	Register Description	Address
RF0_REG00	REG00	0x40040800
RF0_REG01	REG01	0x40040804
RF0_REG02	REG02	0x40040808
RF0_REG03	REG03	0x4004080C
RF0_PADS_03	PADS_03	0x40040810
RF0_PADS_47	PADS_47	0x40040814
RF0_CENTER_FREQ	CENTER_FREQ	0x40040818

Register Name	Register Description	Address
RF0_PADS_89	PADS_89	0x4004081C
RF0_REG08	REG08	0x40040820
RF0_CODING	CODING	0x40040824
RF0_PACKET_HANDLING	PACKET_HANDLING	0x40040828
RF0_SYNC_PATTERN	SYNC_PATTERN	0x4004082C
RF0_REG0C	REG0C	0x40040830
RF0_PACKET_EXTRA	PACKET_EXTRA	0x40040834
RF0_CRC_POLYNOMIAL	CRC_POLYNOMIAL	0x40040838
RF0_CRC_RST	CRC_RST	0x4004083C
RF0_REG10	REG10	0x40040840
RF0_REG11	REG11	0x40040844
RF0_TX_PULSE_SHAPE_1	TX_PULSE_SHAPE_1	0x40040848
RF0_TX_PULSE_SHAPE_2	TX_PULSE_SHAPE_2	0x4004084C
RF0_TX_PULSE_SHAPE_3	TX_PULSE_SHAPE_3	0x40040850
RF0_TX_PULSE_SHAPE_4	TX_PULSE_SHAPE_4	0x40040854
RF0_FRONTEND	FRONTEND	0x40040858
RF0_RX_PULSE_SHAPE	RX_PULSE_SHAPE	0x4004085C
RF0_REG18	REG18	0x40040860
RF0_REG19	REG19	0x40040864
RF0_REG1A	REG1A	0x40040868
RF0_REG1B	REG1B	0x4004086C
RF0_RSSI_CTRL	RSSI_CTRL	0x40040870
RF0_REG1D	REG1D	0x40040874
RF0_AGC_LUT1	AGC_LUT1	0x40040878
RF0_AGC_LUT2	AGC_LUT2	0x4004087C
RF0_AGC_LUT3	AGC_LUT3	0x40040880
RF0_AGC_LUT4	AGC_LUT4	0x40040884
RF0_AGC_LUT5	AGC_LUT5	0x40040888
RF0_AGC_ATT1	AGC_ATT1	0x4004088C
RF0_AGC_ATT2	AGC_ATT2	0x40040890
RF0_REG25	REG25	0x40040894

Register Name	Register Description	Address
RF0_BIAS_0_2	BIAS_0_2	0x40040898
RF0_BIAS_3_6	BIAS_3_6	0x4004089C
RF0_BIAS_7_9	BIAS_7_9	0x400408A0
RF0_BIAS_10_12	BIAS_10_12	0x400408A4
RF0_REG2A	REG2A	0x400408A8
RF0_PLL_CTRL	PLL_CTRL	0x400408AC
RF0_DLL_CTRL	DLL_CTRL	0x400408B0
RF0_REG2D	REG2D	0x400408B4
RF0_REG2E	REG2E	0x400408B8
RF0_XTAL_CTRL	XTAL_CTRL	0x400408BC
RF0_SUBBAND	SUBBAND	0x400408C0
RF0_REG31	REG31	0x400408C4
RF0_DEMOD_CTRL	DEMOD_CTRL	0x400408C8
RF0_REG33	REG33	0x400408CC
RF0_REG34	REG34	0x400408D0
RF0_BLE_LR	BLE_LR	0x400408D4
RF0_REG36	REG36	0x400408D8
RF0_PROT_TIMER	PROT_TIMER	0x400408DC
RF0_CTE_OPTS	CTE_OPTS	0x400408E0
RF0_PT_DELTA_0	PT_DELTA_0	0x400408E4
RF0_PT_DELTA_1	PT_DELTA_1	0x400408E8
RF0_CTE_IF	CTE_IF	0x400408EC
RF0_CTE_CTRL	CTE_CTRL	0x400408F0
RF0_AGC_ADVANCED	AGC_ADVANCED	0x400408F4
RF0_DATA_STREAMING	DATA_STREAMING	0x400408F8
RF0_REVISION	REVISION	0x400408FC
RF0_FSM_CTRL	FSM_CTRL	0x40040900
RF0_IQFIFO_STATUS	IQFIFO_STATUS	0x40040904
RF0_TXFIFO	TXFIFO	0x40040908
RF0_RXFIFO	RXFIFO	0x4004090C
RF0_IQFIFO	IQFIFO	0x40040910

Register Name	Register Description	Address
RF0_REG45	REG45	0x40040914
RF0_DESER_STATUS	DESER_STATUS	0x40040918
RF0_BLE_AEC_CCM	BLE_AEC_CCM	0x4004091C
RF0_IRQ_STATUS	IRQ_STATUS	0x40040920
RF0_RSSI_MIN_MAX	RSSI_MIN_MAX	0x40040924
RF0_REG4A	REG4A	0x40040928
RF0_FEI	FEI	0x4004092C
RF0_REG4C	REG4C	0x40040930
RF0_ANALOG_INFO	ANALOG_INFO	0x40040934
RF0_SAMPLE_RSSI	SAMPLE_RSSI	0x40040938
RF0_RSSI_THERM	RSSI_THERM	0x4004093C
RF0_LUT_ANTENNA_ARRAY_1	LUT_ANTENNA_ARRAY_1	0x40040980
RF0_LUT_ANTENNA_ARRAY_2	LUT_ANTENNA_ARRAY_2	0x40040984
RF0_LUT_ANTENNA_ARRAY_3	LUT_ANTENNA_ARRAY_3	0x40040988
RF0_LUT_ANTENNA_ARRAY_4	LUT_ANTENNA_ARRAY_4	0x4004098C
RF0_REG50	REG50	0x400409C0
RF0_REG51	REG51	0x400409E0
RF0_REG52	REG52	0x400409E8
RF0_REG53	REG53	0x400409F0
RF0_REG54	REG54	0x400409F4
RF0_REG55	REG55	0x400409F8
RF0_REG56	REG56	0x400409FC
RF1_REG00	REG00	0x40040A00
RF1_REG01	REG01	0x40040A04
RF1_REG02	REG02	0x40040A08
RF1_REG03	REG03	0x40040A0C
RF1_PADS_03	PADS_03	0x40040A10
RF1_PADS_47	PADS_47	0x40040A14
RF1_CENTER_FREQ	CENTER_FREQ	0x40040A18
RF1_PADS_89	PADS_89	0x40040A1C
RF1_REG08	REG08	0x40040A20

Register Name	Register Description	Address
RF1_CODING	CODING	0x40040A24
RF1_PACKET_HANDLING	PACKET_HANDLING	0x40040A28
RF1_SYNC_PATTERN	SYNC_PATTERN	0x40040A2C
RF1_REG0C	REG0C	0x40040A30
RF1_PACKET_EXTRA	PACKET_EXTRA	0x40040A34
RF1_CRC_POLYNOMIAL	CRC_POLYNOMIAL	0x40040A38
RF1_CRC_RST	CRC_RST	0x40040A3C
RF1_REG10	REG10	0x40040A40
RF1_REG11	REG11	0x40040A44
RF1_TX_PULSE_SHAPE_1	TX_PULSE_SHAPE_1	0x40040A48
RF1_TX_PULSE_SHAPE_2	TX_PULSE_SHAPE_2	0x40040A4C
RF1_TX_PULSE_SHAPE_3	TX_PULSE_SHAPE_3	0x40040A50
RF1_TX_PULSE_SHAPE_4	TX_PULSE_SHAPE_4	0x40040A54
RF1_FRONTEND	FRONTEND	0x40040A58
RF1_RX_PULSE_SHAPE	RX_PULSE_SHAPE	0x40040A5C
RF1_REG18	REG18	0x40040A60
RF1_REG19	REG19	0x40040A64
RF1_REG1A	REG1A	0x40040A68
RF1_REG1B	REG1B	0x40040A6C
RF1_RSSI_CTRL	RSSI_CTRL	0x40040A70
RF1_REG1D	REG1D	0x40040A74
RF1_AGC_LUT1	AGC_LUT1	0x40040A78
RF1_AGC_LUT2	AGC_LUT2	0x40040A7C
RF1_AGC_LUT3	AGC_LUT3	0x40040A80
RF1_AGC_LUT4	AGC_LUT4	0x40040A84
RF1_AGC_LUT5	AGC_LUT5	0x40040A88
RF1_AGC_ATT1	AGC_ATT1	0x40040A8C
RF1_AGC_ATT2	AGC_ATT2	0x40040A90
RF1_REG25	REG25	0x40040A94
RF1_BIAS_0_2	BIAS_0_2	0x40040A98
RF1_BIAS_3_6	BIAS_3_6	0x40040A9C

Register Name	Register Description	Address
RF1_BIAS_7_9	BIAS_7_9	0x40040AA0
RF1_BIAS_10_12	BIAS_10_12	0x40040AA4
RF1_REG2A	REG2A	0x40040AA8
RF1_PLL_CTRL	PLL_CTRL	0x40040AAC
RF1_DLL_CTRL	DLL_CTRL	0x40040AB0
RF1_REG2D	REG2D	0x40040AB4
RF1_REG2E	REG2E	0x40040AB8
RF1_XTAL_CTRL	XTAL_CTRL	0x40040ABC
RF1_SUBBAND	SUBBAND	0x40040AC0
RF1_REG31	REG31	0x40040AC4
RF1_DEMOD_CTRL	DEMOD_CTRL	0x40040AC8
RF1_REG33	REG33	0x40040ACC
RF1_REG34	REG34	0x40040AD0
RF1_BLE_LR	BLE_LR	0x40040AD4
RF1_REG36	REG36	0x40040AD8
RF1_PROT_TIMER	PROT_TIMER	0x40040ADC
RF1_CTE_OPTS	CTE_OPTS	0x40040AE0
RF1_PT_DELTA_0	PT_DELTA_0	0x40040AE4
RF1_PT_DELTA_1	PT_DELTA_1	0x40040AE8
RF1_CTE_IF	CTE_IF	0x40040AEC
RF1_CTE_CTRL	CTE_CTRL	0x40040AF0
RF1_AGC_ADVANCED	AGC_ADVANCED	0x40040AF4
RF1_DATA_STREAMING	DATA_STREAMING	0x40040AF8
RF1_REVISION	REVISION	0x40040AFC
RF1_FSM_CTRL	FSM_CTRL	0x40040B00
RF1_IQFIFO_STATUS	IQFIFO_STATUS	0x40040B04
RF1_TXFIFO	TXFIFO	0x40040B08
RF1_RXFIFO	RXFIFO	0x40040B0C
RF1_IQFIFO	IQFIFO	0x40040B10
RF1_REG45	REG45	0x40040B14
RF1_DESER_STATUS	DESER_STATUS	0x40040B18

Register Name	Register Description	Address
RF1_BLE_AEC_CCM	BLE_AEC_CCM	0x40040B1C
RF1_IRQ_STATUS	IRQ_STATUS	0x40040B20
RF1_RSSI_MIN_MAX	RSSI_MIN_MAX	0x40040B24
RF1_REG4A	REG4A	0x40040B28
RF1_FEI	FEI	0x40040B2C
RF1_REG4C	REG4C	0x40040B30
RF1_ANALOG_INFO	ANALOG_INFO	0x40040B34
RF1_SAMPLE_RSSI	SAMPLE_RSSI	0x40040B38
RF1_RSSI_THERM	RSSI_THERM	0x40040B3C
RF1_LUT_ANTENNA_ARRAY_1	LUT_ANTENNA_ARRAY_1	0x40040B80
RF1_LUT_ANTENNA_ARRAY_2	LUT_ANTENNA_ARRAY_2	0x40040B84
RF1_LUT_ANTENNA_ARRAY_3	LUT_ANTENNA_ARRAY_3	0x40040B88
RF1_LUT_ANTENNA_ARRAY_4	LUT_ANTENNA_ARRAY_4	0x40040B8C
RF1_REG50	REG50	0x40040BC0
RF1_REG51	REG51	0x40040BE0
RF1_REG52	REG52	0x40040BE8
RF1_REG53	REG53	0x40040BF0
RF1_REG54	REG54	0x40040BF4
RF1_REG55	REG55	0x40040BF8
RF1_REG56	REG56	0x40040BFC
RF2_REG00	REG00	0x40040C00
RF2_REG01	REG01	0x40040C04
RF2_REG02	REG02	0x40040C08
RF2_REG03	REG03	0x40040C0C
RF2_PADS_03	PADS_03	0x40040C10
RF2_PADS_47	PADS_47	0x40040C14
RF2_CENTER_FREQ	CENTER_FREQ	0x40040C18
RF2_PADS_89	PADS_89	0x40040C1C
RF2_REG08	REG08	0x40040C20
RF2_CODING	CODING	0x40040C24
RF2_PACKET_HANDLING	PACKET_HANDLING	0x40040C28

Register Name	Register Description	Address
RF2_SYNC_PATTERN	SYNC_PATTERN	0x40040C2C
RF2_REG0C	REG0C	0x40040C30
RF2_PACKET_EXTRA	PACKET_EXTRA	0x40040C34
RF2_CRC_POLYNOMIAL	CRC_POLYNOMIAL	0x40040C38
RF2_CRC_RST	CRC_RST	0x40040C3C
RF2_REG10	REG10	0x40040C40
RF2_REG11	REG11	0x40040C44
RF2_TX_PULSE_SHAPE_1	TX_PULSE_SHAPE_1	0x40040C48
RF2_TX_PULSE_SHAPE_2	TX_PULSE_SHAPE_2	0x40040C4C
RF2_TX_PULSE_SHAPE_3	TX_PULSE_SHAPE_3	0x40040C50
RF2_TX_PULSE_SHAPE_4	TX_PULSE_SHAPE_4	0x40040C54
RF2_FRONTEND	FRONTEND	0x40040C58
RF2_RX_PULSE_SHAPE	RX_PULSE_SHAPE	0x40040C5C
RF2_REG18	REG18	0x40040C60
RF2_REG19	REG19	0x40040C64
RF2_REG1A	REG1A	0x40040C68
RF2_REG1B	REG1B	0x40040C6C
RF2_RSSI_CTRL	RSSI_CTRL	0x40040C70
RF2_REG1D	REG1D	0x40040C74
RF2_AGC_LUT1	AGC_LUT1	0x40040C78
RF2_AGC_LUT2	AGC_LUT2	0x40040C7C
RF2_AGC_LUT3	AGC_LUT3	0x40040C80
RF2_AGC_LUT4	AGC_LUT4	0x40040C84
RF2_AGC_LUT5	AGC_LUT5	0x40040C88
RF2_AGC_ATT1	AGC_ATT1	0x40040C8C
RF2_AGC_ATT2	AGC_ATT2	0x40040C90
RF2_REG25	REG25	0x40040C94
RF2_BIAS_0_2	BIAS_0_2	0x40040C98
RF2_BIAS_3_6	BIAS_3_6	0x40040C9C
RF2_BIAS_7_9	BIAS_7_9	0x40040CA0
RF2_BIAS_10_12	BIAS_10_12	0x40040CA4

Register Name	Register Description	Address
RF2_REG2A	REG2A	0x40040CA8
RF2_PLL_CTRL	PLL_CTRL	0x40040CAC
RF2_DLL_CTRL	DLL_CTRL	0x40040CB0
RF2_REG2D	REG2D	0x40040CB4
RF2_REG2E	REG2E	0x40040CB8
RF2_XTAL_CTRL	XTAL_CTRL	0x40040CBC
RF2_SUBBAND	SUBBAND	0x40040CC0
RF2_REG31	REG31	0x40040CC4
RF2_DEMOD_CTRL	DEMOD_CTRL	0x40040CC8
RF2_REG33	REG33	0x40040CCC
RF2_REG34	REG34	0x40040CD0
RF2_BLE_LR	BLE_LR	0x40040CD4
RF2_REG36	REG36	0x40040CD8
RF2_PROT_TIMER	PROT_TIMER	0x40040CDC
RF2_CTE_OPTS	CTE_OPTS	0x40040CE0
RF2_PT_DELTA_0	PT_DELTA_0	0x40040CE4
RF2_PT_DELTA_1	PT_DELTA_1	0x40040CE8
RF2_CTE_IF	CTE_IF	0x40040CEC
RF2_CTE_CTRL	CTE_CTRL	0x40040CF0
RF2_AGC_ADVANCED	AGC_ADVANCED	0x40040CF4
RF2_DATA_STREAMING	DATA_STREAMING	0x40040CF8
RF2_REVISION	REVISION	0x40040CFC
RF2_FSM_CTRL	FSM_CTRL	0x40040D00
RF2_IQFIFO_STATUS	IQFIFO_STATUS	0x40040D04
RF2_TXFIFO	TXFIFO	0x40040D08
RF2_RXFIFO	RXFIFO	0x40040D0C
RF2_IQFIFO	IQFIFO	0x40040D10
RF2_REG45	REG45	0x40040D14
RF2_DESER_STATUS	DESER_STATUS	0x40040D18
RF2_BLE_AEC_CCM	BLE_AEC_CCM	0x40040D1C
RF2_IRQ_STATUS	IRQ_STATUS	0x40040D20

Register Name	Register Description	Address
RF2_RSSI_MIN_MAX	RSSI_MIN_MAX	0x40040D24
RF2_REG4A	REG4A	0x40040D28
RF2_FEI	FEI	0x40040D2C
RF2_REG4C	REG4C	0x40040D30
RF2_ANALOG_INFO	ANALOG_INFO	0x40040D34
RF2_SAMPLE_RSSI	SAMPLE_RSSI	0x40040D38
RF2_RSSI_THERM	RSSI_THERM	0x40040D3C
RF2_LUT_ANTENNA_ARRAY_1	LUT_ANTENNA_ARRAY_1	0x40040D80
RF2_LUT_ANTENNA_ARRAY_2	LUT_ANTENNA_ARRAY_2	0x40040D84
RF2_LUT_ANTENNA_ARRAY_3	LUT_ANTENNA_ARRAY_3	0x40040D88
RF2_LUT_ANTENNA_ARRAY_4	LUT_ANTENNA_ARRAY_4	0x40040D8C
RF2_REG50	REG50	0x40040DC0
RF2_REG51	REG51	0x40040DE0
RF2_REG52	REG52	0x40040DE8
RF2_REG53	REG53	0x40040DF0
RF2_REG54	REG54	0x40040DF4
RF2_REG55	REG55	0x40040DF8
RF2_REG56	REG56	0x40040DFC

5.9.0.1 RF_REG00

Bit Field	Read/Write	Field Name	Description
31	RW	DATAWHITE_BTLE_DW_BTLE	Data whitening control
30:24	RW	DATAWHITE_BTLE_DW_BTLE_RST	Reset value to put on the Bluetooth LE data whitening shift register
23	RW	FOURFSK_CODING_EN_FOURFSK_CODING	Enable 4FSK coding
22:20	RW	FOURFSK_CODING_TX_FOURFSK_CODING	Set the 4FSK coding (Tx mode)
18:16	RW	FOURFSK_CODING_RX_FOURFSK_CODING	Set the 4FSK decoding (Rx mode)
14	RW	MODE2_DIFF_CODING	Differential coding/decoding
13	RW	MODE2_PSK_NFSK	FSK/PSK mode selection

Bit Field	Read/Write	Field Name	Description
12:8	RW	MODE2_TESTMODE	Output test mode
7	RW	MODE_NOT_TO_IDLE	FSM goes in suspend mode after a Tx or Rx packet
5	RW	MODE_EN_FSM	Radio FSM control
4	RW	MODE_EN_DESERIALIZER	Deserializer control
3	RW	MODE_EN_SERIALIZER	Serializer control
2	RW	MODE_TX_NRX	Select Tx or Rx mode
1:0	RW	MODE_MODE	Select the working mode of the digital baseband

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31	DATAWHITE_BTLE_DW_BTLE	DATAWHITE_BTLE_DW_BTLE_DISABLE	Disable data whitening	0x0
		DATAWHITE_BTLE_DW_BTLE_ENABLE	Enable data whitening	0x1*
30:24	DATAWHITE_BTLE_DW_BTLE_RST	DATAWHITE_BTLE_DW_BTLE_RST_DEFAULT		0x0*
23	FOURFSK_CODING_EN_FOURFSK_CODING	FOURFSK_CODING_EN_FOURFSK_CODING_DISABLE	Disable 4FSK coding	0x0*
		FOURFSK_CODING_EN_FOURFSK_CODING_ENABLE	Enable 4FSK coding	0x1
22:20	FOURFSK_CODING_TX_FOURFSK_CODING	FOURFSK_CODING_TX_FOURFSK_CODING_DEFAULT	Bit 0 determines 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, bit 2 selects if the signal is inverted for the abs amplitude.	0x0*
18:16	FOURFSK_CODING_RX_FOURFSK_CODING	FOURFSK_CODING_RX_FOURFSK_CODING_DEFAULT	Bit 0 determines if the sign is given by the Q signal (0) or I signal (1). Bit 1 selects if the signal is inverted for the sign, bit 2 selects if the signal is inverted for the abs amplitude.	0x0*
14	MODE2_DIFF_CODING	MODE2_DIFF_CODING_DISABLE	Disable the differential coding/decoding	0x0*
		MODE2_DIFF_CODING_ENABLE	Enable the differential coding/decoding	0x1
13	MODE2_PSK_NFSK	MODE2_PSK_NFSK_FSK	FSK mode is selected	0x0*
		MODE2_PSK_NFSK_PSK	PSK mode is selected	0x1
12:8	MODE2_TESTMODE	MODE2_TESTMODE_OFF		0x0*
		MODE2_TESTMODE_CEVA		0x8

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
7	MODE_NOT_TO_IDLE	MODE_NOT_TO_IDLE_DISABLE	FSM not to go in suspend mode after a Tx or Rx packet	0x0*
		MODE_NOT_TO_IDLE_ENABLE	FSM to go in suspend mode after a Tx or Rx packet	0x1
5	MODE_EN_FSM	MODE_EN_FSM_DISABLE	Disable the radio FSM	0x0
		MODE_EN_FSM_ENABLE	Enable the radio FSM	0x1*
4	MODE_EN_ DESERIALIZER	MODE_EN_DESERIALIZER_ DISABLE	Disable the deserializer	0x0*
		MODE_EN_DESERIALIZER_ ENABLE	Enable the deserializer	0x1
3	MODE_EN_SERIALIZER	MODE_EN_SERIALIZER_DISABLE	Disable the serializer	0x0*
		MODE_EN_SERIALIZER_ENABLE	Enable the serializer	0x1
2	MODE_TX_NRX	MODE_TX_NRX_RX	Select Rx mode	0x0*
		MODE_TX_NRX_TX	Select Tx mode	0x1
1:0	MODE_MODE	MODE_MODE_0	The digital baseband is off	0x0
		MODE_MODE_1	The clock is generated but the blocks are reset (Tx, Rx, FIFOs and FSM)	0x1
		MODE_MODE_2	The digital baseband is frozen	0x2*
		MODE_MODE_3	Working	0x3

5.9.0.2 RF_REG01

Bit Field	Read/Write	Field Name	Description
31:24	RW	TAU_PHASE_RECOV_TAU_ PHASE_RECOV	Time constant of the fine carrier recovery block (banked)
23:16	RW	TAU_ROUGH_RECOV_TAU_ ROUGH_RECOV	Time constant of the rough carrier recovery block (banked)
15	RW	CARRIER_RECOVERY_EN_ CORRECT_CFREQ_AFC	Automatic AFC correction (banked)
14	RW	CARRIER_RECOVERY_ CORRECT_CFREQ_IF_NEG	IF correction (banked)
13	RW	CARRIER_RECOVERY_EN_ CORRECT_CFREQ_IF	Automatic IF correction (banked)
12	RW	CARRIER_RECOVERY_AFC_NEG	AFC correction (banked)
11	RW	CARRIER_RECOVERY_ STARTER_MODE	Starter mode (banked)

Bit Field	Read/Write	Field Name	Description
10	RW	CARRIER_RECOVERY_EN_AFC	Automatic frequency control (banked)
9	RW	CARRIER_RECOVERY_EN_ FINE_RECOV	Fine carrier recovery (banked)
8	RW	CARRIER_RECOVERY_EN_ ROUGH_RECOV	Rough carrier recovery (banked)
6	RW	MOD_TX_PULSE_NSYM	Tx pulse shape function
5	RW	MOD_TX_EN_INTERP	Tx CIC interpolator
4:0	RW	MOD_TX_CK_TX_M	Unsigned value determining the Tx CIC interpolator frequency

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:24	TAU_PHASE_RECOV_ TAU_PHASE_RECOV	TAU_PHASE_RECOV_TAU_PHASE_ RECOV_DEFAULT		0x14*
23:16	TAU_ROUGH_RECOV_ TAU_ROUGH_RECOV	TAU_ROUGH_RECOV_TAU_ROUGH_ RECOV_DEFAULT		0xB*
15	CARRIER_RECOVERY_ EN_CORRECT_CFREQ_ AFC	CARRIER_RECOVERY_EN_ CORRECT_CFREQ_AFC_DISABLE	Disable the automatic AFC correction	0x0*
		CARRIER_RECOVERY_EN_ CORRECT_CFREQ_AFC_ENABLE	Enable the automatic AFC correction	0x1
14	CARRIER_RECOVERY_ CORRECT_CFREQ_IF_ NEG	CARRIER_RECOVERY_CORRECT_ CFREQ_IF_NEG_POS	Positive IF correction	0x0*
		CARRIER_RECOVERY_CORRECT_ CFREQ_IF_NEG_NEG	Negative IF correction	0x1
13	CARRIER_RECOVERY_ EN_CORRECT_CFREQ_ IF	CARRIER_RECOVERY_EN_ CORRECT_CFREQ_IF_DISABLE	Disable the automatic IF correction	0x0
		CARRIER_RECOVERY_EN_ CORRECT_CFREQ_IF_ENABLE	Enable the automatic IF correction	0x1*
12	CARRIER_RECOVERY_ AFC_NEG	CARRIER_RECOVERY_AFC_NEG_ POS	Positive AFC correction	0x0*
		CARRIER_RECOVERY_AFC_NEG_ NEG	Negative AFC correction	0x1
11	CARRIER_RECOVERY_ STARTER_MODE	CARRIER_RECOVERY_STARTER_ MODE_DISABLE	Disable the starter mode	0x0*
		CARRIER_RECOVERY_STARTER_	Enable the starter mode (32x faster	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		MODE_ENABLE	carrier recovery)	
10	CARRIER_RECOVERY_EN_AFC	CARRIER_RECOVERY_EN_AFC_DISABLE	Disable the Automatic Frequency Control	0x0*
		CARRIER_RECOVERY_EN_AFC_ENABLE	Enable the Automatic Frequency Control	0x1
9	CARRIER_RECOVERY_EN_FINE_RECOV	CARRIER_RECOVERY_EN_FINE_RECOV_DISABLE	Disable the fine carrier recovery	0x0
		CARRIER_RECOVERY_EN_FINE_RECOV_ENABLE	Enable the fine carrier recovery	0x1*
8	CARRIER_RECOVERY_EN_ROUGH_RECOV	CARRIER_RECOVERY_EN_ROUGH_RECOV_DISABLE	Disable the rough carrier recovery	0x0*
		CARRIER_RECOVERY_EN_ROUGH_RECOV_ENABLE	Enable the rough carrier recovery	0x1
6	MOD_TX_PULSE_NSYM	MOD_TX_PULSE_NSYM_EVEN	Tx pulse shape is an even function	0x0*
		MOD_TX_PULSE_NSYM_ODD	Tx pulse shape is an odd function	0x1
5	MOD_TX_EN_INTERP	MOD_TX_EN_INTERP_DISABLE	Disable the Tx CIC interpolator	0x0*
		MOD_TX_EN_INTERP_ENABLE	Enable the Tx CIC interpolator	0x1
4:0	MOD_TX_CK_TX_M	MOD_TX_CK_TX_M_DEFAULT	The formula is similar to the evaluation of the oversampling frequency	0x0*

5.9.0.3 RF_REG02

Bit Field	Read/Write	Field Name	Description
25:24	RW	DATARATE_OFFSET_DR_LIMIT	Set the data-rate recovery limits
23:16	RW	DATARATE_OFFSET_DATARATE_OFFSET	Data-rate offset
15:8	RW	TAU_DATARATE_RECOV_TAU_DATARATE_RECOV	Time constant of the data-rate recovery
7:0	RW	TAU_CLK_RECOV_TAU_CLK_RECOV	Time constant of the clock recovery (banked)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
25:24	DATARATE_OFFSET_DR_LIMIT	DATARATE_OFFSET_DR_LIMIT_DEFAULT		0x0*
23:16	DATARATE_OFFSET_DATARATE_OFFSET	DATARATE_OFFSET_DATARATE_OFFSET_DEFAULT	Signed value and the full scale (0x7f) corresponds to a data-rate offset of 12.5%	0x0*
15:8	TAU_DATARATE_RECOV_TAU_DATARATE_RECOV	TAU_DATARATE_RECOV_TAU_DATARATE_RECOV_DEFAULT		0x20*
7:0	TAU_CLK_RECOV_TAU_CLK_RECOV	TAU_CLK_RECOV_TAU_CLK_RECOV_DEFAULT		0x9*

5.9.0.4 RF_REG03

Bit Field	Read/Write	Field Name	Description
31:30	RW	MAC_CONF_MAC_TIMER_GR	MAC timer granularity
29	RW	MAC_CONF_RX_MAC_ACT	Switch FSM to Rx or Tx mode after an Rx mode
28	RW	MAC_CONF_RX_MAC_TX_NRX	Switch FSM to Tx mode after an Rx mode (Rx otherwise)
27	RW	MAC_CONF_RX_MAC_START_NSTOP	MAC timer activation after sync word detection
26	RW	MAC_CONF_TX_MAC_ACT	Switch FSM to Rx or Tx mode after a Tx mode
25	RW	MAC_CONF_TX_MAC_TX_NRX	Switch FSM to Tx mode after a Tx mode (Rx otherwise)
24	RW	MAC_CONF_TX_MAC_START_NSTOP	MAC timer activation after packet transmission
23	RW	IRQ_CONF_IRQ_HIGH_Z	Pads are set to high-Z when the IRQ is not active
22	RW	IRQ_CONF_IRQ_ACTIVE_LOW	IRQ are active low
21:16	RW	IRQ_CONF_IRQS_MASK	Mask to determine which IRQs are enabled (active high)
15:13	RW	FIFO_2_FIFO_THR_TX	Threshold indicating the "almost empty" Tx FIFO state
12	RW	FIFO_2_WAIT_TXFIFO_WR	FSM will wait a Tx FIFO write before starting the Tx mode in case of an empty Tx FIFO
11	RW	FIFO_2_STOP_ON_RXFF_OVFLW	Stop the reception in case of a FIFO overflow
10	RW	FIFO_2_STOP_ON_TXFF_UNFLW	Stop the transmission in case of a FIFO underflow
9	RW	FIFO_2_RXFF_FLUSH_ON_START	Flush the Rx FIFO when the Rx mode is enabled in order to receive a packet with an empty FIFO
8	RW	FIFO_2_TXFF_FLUSH_ON_	Flush the Tx FIFO after the end of a packet transmission in order to

Bit Field	Read/Write	Field Name	Description
		STOP	have an empty FIFO
7	RW	FIFO_FIFO_FLUSH_ON_OVFLW	Overflow FIFO flush control
6	RW	FIFO_FIFO_FLUSH_ON_ADDR_ERR	Address error FIFO flush control
5	RW	FIFO_FIFO_FLUSH_ON_PL_ERR	Packet length error FIFO flush control
4	RW	FIFO_FIFO_FLUSH_ON_CRC_ERR	CRC error FIFO flush control
3	RW	FIFO_RX_FIFO_ACK	Rx FIFO acknowledgement
2:0	RW	FIFO_FIFO_THR	Threshold indicating the "almost full" Rx FIFO state

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:30	MAC_CONF_MAC_TIMER_GR	MAC_CONF_MAC_TIMER_GR_DEFAULT	The granularity is given by (2^(2mac_timer_gr)) x 1us	0x2*
29	MAC_CONF_RX_MAC_ACT	MAC_CONF_RX_MAC_ACT_DISABLE	FSM will not switch to Rx or Tx after an Rx mode	0x0*
		MAC_CONF_RX_MAC_ACT_ENABLE	FSM will switch to Rx or Tx after an Rx mode	0x1
28	MAC_CONF_RX_MAC_TX_NRX	MAC_CONF_RX_MAC_TX_NRX_DISABLE	FSM will not switch to Tx after an Rx mode, Rx otherwise.	0x0*
		MAC_CONF_RX_MAC_TX_NRX_ENABLE	FSM will switch to Tx after an Rx mode, Rx otherwise.	0x1
27	MAC_CONF_RX_MAC_START_NSTOP	MAC_CONF_RX_MAC_START_NSTOP_DISABLE	MAC timer is not activated at the reception of the sync word, at the end of the packet otherwise	0x0*
		MAC_CONF_RX_MAC_START_NSTOP_ENABLE	MAC timer is activated at the reception of the sync word, at the end of the packet otherwise	0x1
26	MAC_CONF_TX_MAC_ACT	MAC_CONF_TX_MAC_ACT_DISABLE	FSM will not switch to Rx or Tx after a Tx mode.	0x0*
		MAC_CONF_TX_MAC_ACT_ENABLE	FSM will switch to Rx or Tx after a Tx mode.	0x1
25	MAC_CONF_TX_MAC_TX_NRX	MAC_CONF_TX_MAC_TX_NRX_DISABLE	FSM will not switch to Tx after an Tx mode, Rx otherwise	0x0*
		MAC_CONF_TX_MAC_TX_NRX_ENABLE	FSM will switch to Tx after an Tx mode, Rx otherwise	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
24	MAC_CONF_TX_MAC_START_NSTOP	MAC_CONF_TX_MAC_START_NSTOP_DISABLE	MAC timer is not activated at beginning of the packet, otherwise at the end of the packet transmission	0x0*
		MAC_CONF_TX_MAC_START_NSTOP_ENABLE	MAC timer is activated at beginning of the packet, otherwise at the end of the packet transmission	0x1
23	IRQ_CONF_IRQ_HIGH_Z	IRQ_CONF_IRQ_HIGH_Z_DISABLE	The pads are not set to High-Z when the IRQ is not active	0x0*
		IRQ_CONF_IRQ_HIGH_Z_ENABLE	The pads are set to High-Z when the IRQ is not active	0x1
22	IRQ_CONF_IRQ_ACTIVE_LOW	IRQ_CONF_IRQ_ACTIVE_LOW_DISABLE	IRQ are active high	0x0
		IRQ_CONF_IRQ_ACTIVE_LOW_ENABLE	IRQ are active low	0x1*
21:16	IRQ_CONF_IRQS_MASK	IRQ_CONF_IRQS_MASK_DEFAULT	No IRQ enable	0x0*
		IRQ_CONF_IRQS_MASK_TX	Tx IRQ enable	0x1
		IRQ_CONF_IRQS_MASK_RX_STOP	Rx stop IRQ enable	0x2
		IRQ_CONF_IRQS_MASK_RECEIVED	Rx received IRQ enable	0x4
		IRQ_CONF_IRQS_MASK_SYNC	Sync IRQ enable	0x8
		IRQ_CONF_IRQS_MASK_TX_FIFO	Tx FIFO IRQ enable	0x10
		IRQ_CONF_IRQS_MASK_RX_FIFO	Rx FIFO IRQ enable	0x20
15:13	FIFO_2_FIFO_THR_TX	FIFO_2_FIFO_THR_TX_16	Tx FIFO threshold is 16 samples	0x0*
		FIFO_2_FIFO_THR_TX_48	Tx FIFO threshold is 48 samples	0x1
		FIFO_2_FIFO_THR_TX_80	Tx FIFO threshold is 80 samples	0x2
		FIFO_2_FIFO_THR_TX_112	Tx FIFO threshold is 112 samples	0x3
		FIFO_2_FIFO_THR_TX_144	Tx FIFO threshold is 144 samples	0x4
		FIFO_2_FIFO_THR_TX_176	Tx FIFO threshold is 176 samples	0x5
		FIFO_2_FIFO_THR_TX_208	Tx FIFO threshold is 208 samples	0x6
		FIFO_2_FIFO_THR_TX_240	Tx FIFO threshold is 240 samples	0x7
12	FIFO_2_WAIT_TXFIFO_WR	FIFO_2_WAIT_TXFIFO_WR_DISABLE	FSM will not wait a Tx FIFO write before starting the Tx in case of an empty Tx FIFO	0x0*
		FIFO_2_WAIT_TXFIFO_WR_ENABLE	FSM will wait a Tx FIFO write before starting the Tx in case of an empty Tx	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
			FIFO	
11	FIFO_2_STOP_ON_ RXFF_OVFLW	FIFO_2_STOP_ON_RXFF_OVFLW_ DISABLE	Keep the reception in case of a FIFO overflow	0x0*
		FIFO_2_STOP_ON_RXFF_OVFLW_ ENABLE	Stop the reception in case of a FIFO overflow	0x1
10	FIFO_2_STOP_ON_ TXFF_UNFLW	FIFO_2_STOP_ON_TXFF_UNFLW_ DISABLE	Keep the transmission in case of a FIFO underflow	0x0*
		FIFO_2_STOP_ON_TXFF_UNFLW_ ENABLE	Stop the transmission in case of a FIFO underflow	0x1
9	FIFO_2_RXFF_FLUSH_ ON_START	FIFO_2_RXFF_FLUSH_ON_ START_DISABLE	Keep the Rx FIFO when the Rx is enabled	0x0
		FIFO_2_RXFF_FLUSH_ON_ START_ENABLE	Flush the Rx FIFO when the Rx is enabled, in order to receive a packet with an empty FIFO	0x1*
8	FIFO_2_TXFF_FLUSH_ ON_STOP	FIFO_2_TXFF_FLUSH_ON_STOP_ DISABLE	Keep the Tx FIFO after the end of a packet transmission in order to have an empty FIFO	0x0
		FIFO_2_TXFF_FLUSH_ON_STOP_ ENABLE	Flush the Tx FIFO after the end of a packet transmission in order to have an empty FIFO	0x1*
7	FIFO_FIFO_FLUSH_ON_ OVFLW	FIFO_FIFO_FLUSH_ON_OVFLW_ DISABLE	Keep the Rx and the FIFO in case of overflow	0x0*
		FIFO_FIFO_FLUSH_ON_OVFLW_ ENABLE	Stop the Rx and flush the FIFO in case of overflow	0x1
6	FIFO_FIFO_FLUSH_ON_ ADDR_ERR	FIFO_FIFO_FLUSH_ON_ADDR_ ERR_DISABLE	Keep the Rx and the FIFO in case of address error	0x0*
		FIFO_FIFO_FLUSH_ON_ADDR_ ERR_ENABLE	Stop the Rx and flush the FIFO in case of address error	0x1
5	FIFO_FIFO_FLUSH_ON_ PL_ERR	FIFO_FIFO_FLUSH_ON_PL_ERR_ DISABLE	Keep the Rx and the FIFO in case of packet length error	0x0*
		FIFO_FIFO_FLUSH_ON_PL_ERR_ ENABLE	Stop the Rx and flush the FIFO in case of packet length error	0x1
4	FIFO_FIFO_FLUSH_ON_ CRC_ERR	FIFO_FIFO_FLUSH_ON_CRC_ ERR_DISABLE	Keep the Rx and the FIFO in case of CRC error	0x0
		FIFO_FIFO_FLUSH_ON_CRC_ ERR_ENABLE	Stop the Rx and flush the FIFO in case of CRC error	0x1*
3	FIFO_RX_FIFO_ACK	FIFO_RX_FIFO_ACK_DISABLE	Rx FIFO doesn't need an	0x0*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
			acknowledgement (packet received correctly) to change its state	
		FIFO_RX_FIFO_ACK_ENABLE	Rx FIFO needs an acknowledgement (packet received correctly) to change its state	0x1
2:0	FIFO_FIFO_THR	FIFO_FIFO_THR_240	Rx FIFO threshold is 240 samples	0x0*
		FIFO_FIFO_THR_208	Rx FIFO threshold is 208 samples	0x1
		FIFO_FIFO_THR_176	Rx FIFO threshold is 176 samples	0x2
		FIFO_FIFO_THR_144	Rx FIFO threshold is 144 samples	0x3
		FIFO_FIFO_THR_112	Rx FIFO threshold is 112 samples	0x4
		FIFO_FIFO_THR_80	Rx FIFO threshold is 80 samples	0x5
		FIFO_FIFO_THR_48	Rx FIFO threshold is 48 samples	0x6
		FIFO_FIFO_THR_16	Rx FIFO threshold is 16 samples	0x7

5.9.0.5 RF_PADS_03

Bit Field	Read/Write	Field Name	Description
28:24	RW	PAD_CONF_1_PAD_3_CONF	Configuration of GPIO pad 3
20:16	RW	PAD_CONF_1_PAD_2_CONF	Configuration of GPIO pad 2
12:8	RW	PAD_CONF_1_PAD_1_CONF	Configuration of GPIO pad 1
4:0	RW	PAD_CONF_1_PAD_0_CONF	Configuration of GPIO pad 0

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
28:24	PAD_CONF_1_PAD_3_ CONF	PAD_CONF_1_PAD_3_OFF		0x0*
		PAD_CONF_1_PAD_3_TEST_ MODE		0xF
20:16	PAD_CONF_1_PAD_2_ CONF	PAD_CONF_1_PAD_2_OFF		0x0*
		PAD_CONF_1_PAD_2_TEST_ MODE		0xF
12:8	PAD_CONF_1_PAD_1_ CONF	PAD_CONF_1_PAD_1_OFF		0x0*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		PAD_CONF_1_PAD_1_TEST_MODE		0xF
4:0	PAD_CONF_1_PAD_0_CONF	PAD_CONF_1_PAD_0_OFF		0x0*
		PAD_CONF_1_PAD_0_TEST_MODE		0xF

5.9.0.6 RF_PADS_47

Bit Field	Read/Write	Field Name	Description
28:24	RW	PAD_CONF_2_PAD_7_CONF	Configuration of GPIO pad 7
20:16	RW	PAD_CONF_2_PAD_6_CONF	Configuration of GPIO pad 6
12:8	RW	PAD_CONF_2_PAD_5_CONF	Configuration of GPIO pad 5
4:0	RW	PAD_CONF_2_PAD_4_CONF	Configuration of GPIO pad 4

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
28:24	PAD_CONF_2_PAD_7_CONF	PAD_CONF_2_PAD_7_OFF		0x0*
		PAD_CONF_2_PAD_7_TEST_MODE		0xF
20:16	PAD_CONF_2_PAD_6_CONF	PAD_CONF_2_PAD_6_OFF		0x0*
		PAD_CONF_2_PAD_6_TEST_MODE		0xF
12:8	PAD_CONF_2_PAD_5_CONF	PAD_CONF_2_PAD_5_OFF		0x0*
		PAD_CONF_2_PAD_5_TEST_MODE		0xF
4:0	PAD_CONF_2_PAD_4_CONF	PAD_CONF_2_PAD_4_OFF		0x0*
		PAD_CONF_2_PAD_4_TEST_MODE		0xF

5.9.0.7 RF_CENTER_FREQ

Bit Field	Read/Write	Field Name	Description
31	RW	CENTER_FREQ_ADAPT_CFREQ	Frequency adaptation between Tx and Rx modes
30	RW	CENTER_FREQ_RX_DIV_5_N6	Ratio of the PLL reference between Tx and Rx modes
29:0	RW	CENTER_FREQ_CENTER_FREQUENCY	Set the center frequency

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31	CENTER_FREQ_ADAPT_CFREQ	CENTER_FREQ_ADAPT_CFREQ_DISABLE	Do not adapt frequency between Tx and Rx	0x0
		CENTER_FREQ_ADAPT_CFREQ_ENABLE	Automatically adapt frequency between Tx and Rx	0x1*
30	CENTER_FREQ_RX_DIV_5_N6	CENTER_FREQ_RX_DIV_5_N6_DISABLE	The ratio of the PLL reference between Tx and Rx is 6 instead of 5	0x0*
		CENTER_FREQ_RX_DIV_5_N6_ENABLE	The ratio of the PLL reference between Tx and Rx is 5 instead of 6	0x1
29:0	CENTER_FREQ_CENTER_FREQUENCY	CENTER_FREQ_CENTER_FREQUENCY_DEFAULT		0x215C71B*

5.9.0.8 RF_PADS_89

Bit Field	Read/Write	Field Name	Description
31:24	RW	TX_MAC_TIMER_TX_MAC_TIMER	Time to wait after the Tx mode
23:16	RW	RX_MAC_TIMER_RX_MAC_TIMER	Time to wait after the Rx mode
12:8	RW	PAD_CONF_3_PAD_9_CONF	Configuration of GPIO pad 9
4:0	RW	PAD_CONF_3_PAD_8_CONF	Configuration of GPIO pad 8

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:24	TX_MAC_TIMER_TX_MAC_TIMER	TX_MAC_TIMER_TX_MAC_TIMER_DEFAULT		0x82*
23:16	RX_MAC_TIMER_RX_MAC_TIMER	RX_MAC_TIMER_RX_MAC_TIMER_DEFAULT		0x23*
12:8	PAD_CONF_3_PAD_9_CONF	PAD_CONF_3_PAD_9_OFF		0x0*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		PAD_CONF_3_PAD_9_TEST_MODE		0xF
4:0	PAD_CONF_3_PAD_8_CONF	PAD_CONF_3_PAD_8_OFF		0x0*
		PAD_CONF_3_PAD_8_TEST_MODE		0xF

5.9.0.9 RF_REG08

Bit Field	Read/Write	Field Name	Description
31:30	RW	MOD_INFO_RX_DIV_CK_RX	Set the clock divider for the Rx mode (banked)
29	RW	MOD_INFO_RX_SYMBOL_2BIT_RX	Rx symbol bits composition (banked)
28:24	RW	MOD_INFO_RX_DR_M_RX	Unsigned value determining the oversampling frequency and consequently the data-rate (banked)
23:22	RW	MOD_INFO_TX_DIV_CK_TX	Set the clock divider for the Tx mode (banked)
21	RW	MOD_INFO_TX_SYMBOL_2BIT_TX	Tx symbol bits composition (banked)
20:16	RW	MOD_INFO_TX_DR_M_TX	Unsigned value determining the oversampling frequency and consequently the data-rate (banked)
14	RW	CHANNEL_SWITCH_IQ	Switch I and Q channels
13:8	RW	CHANNEL_CHANNEL	Channel number
3	RW	BANK_DATARATE_TX_NRX	Select the data-rate register
2	RW	BANK_STD_BLE_RATES	Select the actual bank behavior
1:0	RW	BANK_BANK	Select the used bank

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:30	MOD_INFO_RX_DIV_CK_RX	MOD_INFO_RX_DIV_CK_RX_DEFAULT	00 => /1, 01 => /2, 1x => /3	0x0*
29	MOD_INFO_RX_SYMBOL_2BIT_RX	MOD_INFO_RX_SYMBOL_2BIT_RX_DISABLE	Each symbol is not composed by 2 bits (OQPSK or 4FSK)	0x0*
		MOD_INFO_RX_SYMBOL_2BIT_RX_ENABLE	Each symbol is composed by 2 bits (OQPSK or 4FSK)	0x1
28:24	MOD_INFO_RX_DR_M_RX	MOD_INFO_RX_DR_M_RX_DEFAULT	This frequency is the system frequency (16 or 24 MHz) divided by this value+1	0x0*
23:22	MOD_INFO_TX_DIV_	MOD_INFO_TX_DIV_CK_TX_	00 => /1, 01 => /2, 1x => /3	0x0*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
	CK_TX	DEFAULT		
21	MOD_INFO_TX_ SYMBOL_2BIT_TX	MOD_INFO_TX_SYMBOL_2BIT_ TX_DISABLE	Each symbol is not composed by 2 bits (OQPSK or 4FSK)	0x0*
		MOD_INFO_TX_SYMBOL_2BIT_ TX_ENABLE	Each symbol is composed by 2 bits (OQPSK or 4FSK)	0x1
20:16	MOD_INFO_TX_DR_M_ TX	MOD_INFO_TX_DR_M_TX_ DEFAULT	This frequency is the system frequency (16 or 24 MHz) divided by this value+1	0x0*
14	CHANNEL_SWITCH_IQ	CHANNEL_SWITCH_IQ_DISABLE	Don't switch I and Q channels	0x0*
		CHANNEL_SWITCH_IQ_ENABLE	Switch I and Q channels	0x1
13:8	CHANNEL_CHANNEL	CHANNEL_CHANNEL_DEFAULT		0x0*
3	BANK_DATARATE_TX_ NRX	BANK_DATARATE_TX_NRX_RX	Rx data-rate	0x0*
		BANK_DATARATE_TX_NRX_TX	Tx data-rate	0x1
2	BANK_STD_BLE_RATES	BANK_STD_BLE_RATES_CUSTOM	Custom rates	0x0*
		BANK_STD_BLE_RATES_ STANDARD	Standard BLE rates	0x1
1:0	BANK_BANK	BANK_BANK_DEFAULT		0x0*

5.9.0.10 RF_CODING

Bit Field	Read/Write	Field Name	Description
31	RW	CODING_EN_DATAWHITE	Data-whitening enabling (banked)
30	RW	CODING_I_NQ_DELAYED	Channel I delay (banked)
29	RW	CODING_OFFSET	Offset (delay) introduction (banked)
28	RW	CODING_BIT_INVERT	Bit value inversion in Tx and Rx modes (banked)
27	RW	CODING_EVEN_BEFORE_ODD	Determine the bit order in case of a 2 bits per symbol modulation (banked)
26	RW	CODING_EN_802154_L2F	Linear to frequency encoding needed in order to modulate an OQPSK as an MSK (banked)
25	RW	CODING_EN_802154_B2C	Bit to chips encoding used in the IEEE 802.15.4 standard (banked)
24	RW	CODING_EN_MANCHESTER	Manchester encoding (banked)
23	RW	CHANNELS_2_EN_CHANNEL_ SEL	Definition of channels (banked)

Bit Field	Read/Write	Field Name	Description
22	RW	CHANNELS_2_EN_CHN_BLE	BLE channels index LUT (banked)
19:16	RW	CHANNELS_2_CHANNEL_SPACING_HI	Channel spacing MSB (banked)
15:0	RW	CHANNELS_1_CHANNEL_SPACING_LO	Channel spacing LSB (banked)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31	CODING_EN_DATAWHITE	CODING_EN_DATAWHITE_DISABLE	Disable the data-whitening	0x0
		CODING_EN_DATAWHITE_ENABLE	Enable the data-whitening	0x1*
30	CODING_I_NQ_DELAYED	CODING_I_NQ_DELAYED_DISABLE	Channel I is not considered as delayed in case of a 2bit per symbol modulation	0x0*
		CODING_I_NQ_DELAYED_ENABLE	Channel I is considered as delayed in case of a 2bit per symbol modulation	0x1
29	CODING_OFFSET	CODING_OFFSET_DISABLE	No offset (delay) is introduced in one of the two channels (2 bits per symbol modulation)	0x0*
		CODING_OFFSET_ENABLE	An offset (delay) is introduced in one of the two channels (2 bits per symbol modulation)	0x1
28	CODING_BIT_INVERT	CODING_BIT_INVERT_DISABLE	Original bit value (Tx and Rx)	0x0*
		CODING_BIT_INVERT_ENABLE	Inversed bit value (Tx and Rx)	0x1
27	CODING_EVEN_BEFORE_ODD	CODING_EVEN_BEFORE_ODD_DISABLE	Second bit (bit 1, odd) goes to the I path	0x0*
		CODING_EVEN_BEFORE_ODD_ENABLE	First bit (bit 0, even) goes to the I path	0x1
26	CODING_EN_802154_L2F	CODING_EN_802154_L2F_DISABLE	Disable the linear to frequency encoding	0x0*
		CODING_EN_802154_L2F_ENABLE	Enable the linear to frequency encoding	0x1
25	CODING_EN_802154_B2C	CODING_EN_802154_B2C_DISABLE	Disable the bit to chips encoding used in the IEEE 802.15.4 standard	0x0*
		CODING_EN_802154_B2C_ENABLE	Enable the bit to chips encoding used in the IEEE 802.15.4 standard	0x1
24	CODING_EN_	CODING_EN_MANCHESTER_	Disable the Manchester encoding	0x0*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
	MANCHESTER	DISABLE		
		CODING_EN_MANCHESTER_ENABLE	Enable the Manchester encoding	0x1
23	CHANNELS_2_EN_CHANNEL_SEL	CHANNELS_2_EN_CHANNEL_SEL_DISABLE	Disable the definition of channels	0x0
		CHANNELS_2_EN_CHANNEL_SEL_ENABLE	Enable the definition of channels	0x1*
22	CHANNELS_2_EN_CHN_BLE	CHANNELS_2_EN_CHN_BLE_DISABLE	Disable the BLE channels index LUT	0x0
		CHANNELS_2_EN_CHN_BLE_ENABLE	Enable the BLE channels index LUT	0x1*
19:16	CHANNELS_2_CHANNEL_SPACING_HI	CHANNELS_2_CHANNEL_SPACING_HI_DEFAULT	The formula that determines this value is the same as for the central frequency	0x7*
15:0	CHANNELS_1_CHANNEL_SPACING_LO	CHANNELS_1_CHANNEL_SPACING_LO_DEFAULT	The formula that determines this value is the same as for the central frequency	0x1C72*

5.9.0.11 RF_PACKET_HANDLING

Bit Field	Read/Write	Field Name	Description
31:24	RW	PREAMBLE_PREAMBLE	Preamble to be inserted (banked)
22	RW	PACKET_LENGTH_OPTS_EN_PACKET_LEN_FIX	Packet length configuration (banked)
21:18	RW	PACKET_LENGTH_OPTS_PACKET_LEN_CORR	Signed value specifying the correction to apply to the specified packet length (banked)
17:16	RW	PACKET_LENGTH_OPTS_PACKET_LEN_POS	Unsigned value that specifies the position of the packet length after the pattern (banked)
15:8	RW	PACKET_LENGTH_PACKET_LEN	The packet length in the fixed packet length mode (banked)
7	RW	PACKET_HANDLING_LSB_FIRST	Select LSB or MSB to send first (banked)
6	RW	PACKET_HANDLING_EN_CRC	Automatic CRC evaluation and insertion (banked)
5	RW	PACKET_HANDLING_EN_CRC_ON_PKTLEN	CRC calculation on the packet length part of the packet (banked)
4	RW	PACKET_HANDLING_EN_PREAMBLE	Automatic preamble insertion (banked)
3	RW	PACKET_HANDLING_EN_MULTI_FRAME	Multi-frame packet (banked)

Bit Field	Read/Write	Field Name	Description
2	RW	PACKET_HANDLING_ENB_DW_ON_CRC	Data-whitening on the CRC disabling (banked)
1	RW	PACKET_HANDLING_EN_PATTERN	Automatic pattern insertion and recognition (banked)
0	RW	PACKET_HANDLING_EN_PACKET	Packet handler enabling (banked)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:24	PREAMBLE_PREAMBLE	PREAMBLE_PREAMBLE_DEFAULT		0x55*
22	PACKET_LENGTH_OPTS_EN_PACKET_LEN_FIX	PACKET_LENGTH_OPTS_EN_PACKET_LEN_FIX_DISABLE	Packet length is not fixed	0x0*
		PACKET_LENGTH_OPTS_EN_PACKET_LEN_FIX_ENABLE	Packet length is fixed and specified in the PACKET_LEN register	0x1
21:18	PACKET_LENGTH_OPTS_PACKET_LEN_CORR	PACKET_LENGTH_OPTS_PACKET_LEN_CORR_DEFAULT	The packet length here is specified by the byte number after the packet length byte, with the exclusion of the CRC	0x0*
17:16	PACKET_LENGTH_OPTS_PACKET_LEN_POS	PACKET_LENGTH_OPTS_PACKET_LEN_POS_DEFAULT		0x1*
15:8	PACKET_LENGTH_PACKET_LEN	PACKET_LENGTH_PACKET_LEN_DEFAULT	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.	0xFF*
7	PACKET_HANDLING_LSB_FIRST	PACKET_HANDLING_LSB_FIRST_MSB	MSB is the first bit to be sent	0x0
		PACKET_HANDLING_LSB_FIRST_LSB	LSB is the first bit to be sent	0x1*
6	PACKET_HANDLING_EN_CRC	PACKET_HANDLING_EN_CRC_DISABLE	Disable the automatic CRC evaluation and insertion	0x0
		PACKET_HANDLING_EN_CRC_ENABLE	Enable the automatic CRC evaluation and insertion	0x1*
5	PACKET_HANDLING_EN_CRC_ON_PKTLEN	PACKET_HANDLING_EN_CRC_ON_PKTLEN_DISABLE	Disable the CRC calculation on the packet length part of the packet.	0x0
		PACKET_HANDLING_EN_CRC_ON_PKTLEN_ENABLE	Enable the CRC calculation on the packet length part of the packet.	0x1*
4	PACKET_HANDLING_	PACKET_HANDLING_EN_	Disable the automatic preamble insertion	0x0

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
	EN_PREAMBLE	PREAMBLE_DISABLE		
		PACKET_HANDLING_EN_PREAMBLE_ENABLE	Enable the automatic preamble insertion	0x1*
3	PACKET_HANDLING_EN_MULTI_FRAME	PACKET_HANDLING_EN_MULTI_FRAME_DISABLE	Disable the multi-frame packet (preamble-pattern-data-CRC-data-CRC-...)	0x0*
		PACKET_HANDLING_EN_MULTI_FRAME_ENABLE	Enable the multi-frame packet (preamble-pattern-data-CRC-data-CRC-...)	0x1
2	PACKET_HANDLING_ENB_DW_ON_CRC	PACKET_HANDLING_ENB_DW_ON_CRC_ENABLE	Enable the data-whitening on the CRC	0x0*
		PACKET_HANDLING_ENB_DW_ON_CRC_DISABLE	Disable the data-whitening on the CRC	0x1
1	PACKET_HANDLING_EN_PATTERN	PACKET_HANDLING_EN_PATTERN_DISABLE	Disable the automatic pattern insertion and recognition	0x0
		PACKET_HANDLING_EN_PATTERN_ENABLE	Enable the automatic pattern insertion and recognition	0x1*
0	PACKET_HANDLING_EN_PACKET	PACKET_HANDLING_EN_PACKET_DISABLE	Disable the packet handler	0x0
		PACKET_HANDLING_EN_PACKET_ENABLE	Enable the packet handler	0x1*

5.9.0.12 RF_SYNC_PATTERN

Bit Field	Read/Write	Field Name	Description
31:0	RW	PATTERN	Pattern (sync word) to be inserted or recognized (banked)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:0	PATTERN	PATTERN_DEFAULT		0x8E89BED6*

5.9.0.13 RF_REG0C

Bit Field	Read/Write	Field Name	Description
31:16	RW	ADDRESS_ADDRESS	Address of the node (banked)
11	RW	ADDRESS_CONF_ADDRESS_LEN	Address length selection (banked)
10	RW	ADDRESS_CONF_EN_ADDRESS_RX_BR	Broadcast address detection in Rx mode (banked)
9	RW	ADDRESS_CONF_EN_ADDRESS_	Address detection in Rx mode (banked)

Bit Field	Read/Write	Field Name	Description
		RX	
8	RW	ADDRESS_CONF_EN_ADDRESS_TX	Address insertion in Tx mode (banked)
7:0	RW	PREAMBLE_LENGTH_PREAMBLE_LEN	Length of the preamble -1 (banked)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:16	ADDRESS_ADDRESS	ADDRESS_ADDRESS_DEFAULT		0x0*
11	ADDRESS_CONF_ADDRESS_LEN	ADDRESS_CONF_ADDRESS_LEN_8	Address length is 8 bits	0x0*
		ADDRESS_CONF_ADDRESS_LEN_16	Address length is 16 bits	0x1
10	ADDRESS_CONF_EN_ADDRESS_RX_BR	ADDRESS_CONF_EN_ADDRESS_RX_BR_DISABLE	Disable the broadcast address detection on Rx	0x0*
		ADDRESS_CONF_EN_ADDRESS_RX_BR_ENABLE	Enable the broadcast address detection on Rx	0x1
9	ADDRESS_CONF_EN_ADDRESS_RX	ADDRESS_CONF_EN_ADDRESS_RX_DISABLE	Disable the address detection on Rx	0x0*
		ADDRESS_CONF_EN_ADDRESS_RX_ENABLE	Enable the address detection on Rx	0x1
8	ADDRESS_CONF_EN_ADDRESS_TX	ADDRESS_CONF_EN_ADDRESS_TX_DISABLE	Disable the address insertion on Tx	0x0*
		ADDRESS_CONF_EN_ADDRESS_TX_ENABLE	Enable the address insertion on Tx	0x1
7:0	PREAMBLE_LENGTH_PREAMBLE_LEN	PREAMBLE_LENGTH_PREAMBLE_LEN_DEFAULT		0x0*

5.9.0.14 RF_PACKET_EXTRA

Bit Field	Read/Write	Field Name	Description
29:28	RW	CONV_CODES_CONF_STOP_WORD_LEN	Length of the stop word (banked)
27:26	RW	CONV_CODES_CONF_CC_VITERBI_LEN	Set the memory length of the Viterbi decoder (banked)
25	RW	CONV_CODES_CONF_CC_EN_TX_STOP	Stop word at the end of the transmission (banked)
24	RW	CONV_CODES_CONF_EN_CONV_	Convolutional codes (banked)

Bit Field	Read/Write	Field Name	Description
		CODE	
22	RW	PACKET_EXTRA_FIFO_REWIND	Rewind the FIFO to the initial stage at the end of a Tx transmission (banked)
21	RW	PACKET_EXTRA_BLE_ PREAMBLE	Handle the preamble directly in Tx mode (PREAMBLE register is not used) according to the BLE standard (banked)
20	RW	PACKET_EXTRA_PKT_INFO_ PRE_NPOST	Packet information sampling (banked)
19:18	RW	PACKET_EXTRA_PATTERN_ MAX_ERR	Unsigned value that specifies the maximum number of errors in the pattern recognition (banked)
17:16	RW	PACKET_EXTRA_PATTERN_ WORD_LEN	Pattern word length (banked)
15:0	RW	ADDRESS_BROADCAST_ ADDRESS_BR	Broadcast address (banked)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
29:28	CONV_CODES_CONF_ STOP_WORD_LEN	CONV_CODES_CONF_STOP_WORD_ LEN_DEFAULT	Same as the pattern word length	0x0*
27:26	CONV_CODES_CONF_ CC_VITERBI_LEN	CONV_CODES_CONF_CC_ VITERBI_LEN_5	5	0x0
		CONV_CODES_CONF_CC_ VITERBI_LEN_10	10	0x1
		CONV_CODES_CONF_CC_ VITERBI_LEN_20	20	0x2*
		CONV_CODES_CONF_CC_ VITERBI_LEN_30	30	0x3
25	CONV_CODES_CONF_ CC_EN_TX_STOP	CONV_CODES_CONF_CC_EN_TX_ STOP_DISABLE	Disable the stop word at the end of the transmission	0x0*
		CONV_CODES_CONF_CC_EN_TX_ STOP_ENABLE	Enable the stop word at the end of the transmission	0x1
24	CONV_CODES_CONF_ EN_CONV_CODE	CONV_CODES_CONF_EN_CONV_ CODE_DISABLE	Disable the convolutional codes	0x0*
		CONV_CODES_CONF_EN_CONV_ CODE_ENABLE	Enable the convolutional codes	0x1
22	PACKET_EXTRA_FIFO_ REWIND	PACKET_EXTRA_FIFO_REWIND_ DISABLE	FIFO is not rewind at the end of Tx	0x0*
		PACKET_EXTRA_FIFO_REWIND_	FIFO is rewind at the end of Tx	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		ENABLE		
21	PACKET_EXTRA_BLE_PREAMBLE	PACKET_EXTRA_BLE_PREAMBLE_DISABLE	Preamble register is used	0x0
		PACKET_EXTRA_BLE_PREAMBLE_ENABLE	Preamble register is not used	0x1*
20	PACKET_EXTRA_PKT_INFO_PRE_NPOST	PACKET_EXTRA_PKT_INFO_PRE_NPOST_SYNC	Sample packet information at the sync word detection	0x0*
		PACKET_EXTRA_PKT_INFO_PRE_NPOST_END	Sample packet information at the end of the packet	0x1
19:18	PACKET_EXTRA_PATTERN_MAX_ERR	PACKET_EXTRA_PATTERN_MAX_ERR_DEFAULT		0x0*
17:16	PACKET_EXTRA_PATTERN_WORD_LEN	PACKET_EXTRA_PATTERN_WORD_LEN_8	8 bits	0x0
		PACKET_EXTRA_PATTERN_WORD_LEN_16	16 bits	0x1
		PACKET_EXTRA_PATTERN_WORD_LEN_24	24 bits	0x2
		PACKET_EXTRA_PATTERN_WORD_LEN_32	32 bits	0x3*
15:0	ADDRESS_BROADCAST_ADDRESS_BR	ADDRESS_BROADCAST_ADDRESS_BR_DEFAULT		0x0*

5.9.0.15 RF_CRC_POLYNOMIAL

Bit Field	Read/Write	Field Name	Description
31:0	RW	CRC_POLY	CRC polynomial (banked)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:0	CRC_POLY	CRC_POLY_DEFAULT	It is coded using the Koopman notation, i.e. the nth bit codes the (n+1) coefficient.	0x80032D*

5.9.0.16 RF_CRC_RST

Bit Field	Read/Write	Field Name	Description
31:0	RW	CRC_RST	CRC reset value (banked)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:0	CRC_RST	CRC_RST_DEFAULT		0x555555*

5.9.0.17 RF_REG10

Bit Field	Read/Write	Field Name	Description
25:21	RW	CONV_CODES_PUNCT_CC_PUNCT_1	Puncture of the second convolutional code (banked)
20:16	RW	CONV_CODES_PUNCT_CC_PUNCT_0	Puncture of the first convolutional code (banked)
11	RW	FRAC_CONF_TX_FRAC_GAIN	Additional gain for fractional data-rates in Tx mode (banked)
10	RW	FRAC_CONF_RX_FRAC_GAIN	Additional gain for fractional data-rates in Rx mode (banked)
9	RW	FRAC_CONF_TX_EN_FRAC	Fractional data-rates in Tx mode (banked)
8	RW	FRAC_CONF_RX_EN_FRAC	Fractional data-rates in Rx mode (banked)
7:4	RW	CONV_CODES_POLY_CC_POLY_1	Second convolutional code (banked)
3:0	RW	CONV_CODES_POLY_CC_POLY_0	First convolutional code (banked)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
25:21	CONV_CODES_PUNCT_CC_PUNCT_1	CONV_CODES_PUNCT_CC_PUNCT_1_DEFAULT		0x1*
20:16	CONV_CODES_PUNCT_CC_PUNCT_0	CONV_CODES_PUNCT_CC_PUNCT_0_DEFAULT		0x1*
11	FRAC_CONF_TX_FRAC_GAIN	FRAC_CONF_TX_FRAC_GAIN_DISABLE	Disable the additional gain for fractional data-rates in Tx mode	0x0*
		FRAC_CONF_TX_FRAC_GAIN_ENABLE	Enable the additional gain for fractional data-rates in Tx mode	0x1
10	FRAC_CONF_RX_FRAC_GAIN	FRAC_CONF_RX_FRAC_GAIN_DISABLE	Disable the additional gain for fractional data-rates in Rx mode	0x0*
		FRAC_CONF_RX_FRAC_GAIN_ENABLE	Enable the additional gain for fractional data-rates in Rx mode	0x1
9	FRAC_CONF_TX_EN_FRAC	FRAC_CONF_TX_EN_FRAC_DISABLE	Disable the fractional data-rates in Tx mode	0x0*
		FRAC_CONF_TX_EN_FRAC_ENABLE	Enable the fractional data-rates in Tx mode	0x1
8	FRAC_CONF_RX_EN_	FRAC_CONF_RX_EN_FRAC_	Disable the fractional data-rates in Rx	0x0*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
	FRAC	DISABLE	mode	
		FRAC_CONF_RX_EN_FRAC_ENABLE	Enable the fractional data-rates in Rx mode	0x1
7:4	CONV_CODES_POLY_CC_POLY_1	CONV_CODES_POLY_CC_POLY_1_DEFAULT		0xD*
3:0	CONV_CODES_POLY_CC_POLY_0	CONV_CODES_POLY_CC_POLY_0_DEFAULT		0xF*

5.9.0.18 RF_REG11

Bit Field	Read/Write	Field Name	Description
31	RW	FILTER_GAIN_LIN_FILTER	Enable the linear filtering (banked)
30	RW	FILTER_GAIN_LOW_LIN_GAIN	Reduce the total gain by two if the linear gain is set (banked)
29:27	RW	FILTER_GAIN_GAIN_M	Mantissa of the final stage gain of the matched filter (banked)
26:24	RW	FILTER_GAIN_GAIN_E	Exponent of the final stage gain of the matched filter (banked)
23:20	RW	TX_MULT_TX_MULT_EXP	Exponent of the Tx multiplier (banked)
19:16	RW	TX_MULT_TX_MULT_MAN	Mantissa of the Tx multiplier (banked)
15:12	RW	TX_FRAC_CONF_TX_FRAC_DEN	Denominator of the fractional data-rate in Tx mode (banked)
11:8	RW	TX_FRAC_CONF_TX_FRAC_NUM	Numerator of the fractional data-rate in Tx mode (banked)
7:4	RW	RX_FRAC_CONF_RX_FRAC_DEN	Denominator of the fractional data-rate in Rx mode (banked)
3:0	RW	RX_FRAC_CONF_RX_FRAC_NUM	Numerator of the fractional data-rate in Rx mode (banked)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31	FILTER_GAIN_LIN_FILTER	FILTER_GAIN_LIN_FILTER_DISABLE	Disable the linear filtering	0x0*
		FILTER_GAIN_LIN_FILTER_ENABLE	Enable the linear filtering	0x1
30	FILTER_GAIN_LOW_LIN_GAIN	FILTER_GAIN_LOW_LIN_GAIN_DISABLE	Disable total gain reduction	0x0*
		FILTER_GAIN_LOW_LIN_GAIN_ENABLE	Enable total gain reduction	0x1
29:27	FILTER_GAIN_GAIN_M	FILTER_GAIN_GAIN_M_DEFAULT		0x0*
26:24	FILTER_GAIN_GAIN_E	FILTER_GAIN_GAIN_E_DEFAULT		0x0*
23:20	TX_MULT_TX_MULT_	TX_MULT_TX_MULT_EXP_		0x2*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
	EXP	DEFAULT		
19:16	TX_MULT_TX_MULT_MAN	TX_MULT_TX_MULT_MAN_DEFAULT		0x9*
15:12	TX_FRAC_CONF_TX_FRAC_DEN	TX_FRAC_CONF_TX_FRAC_DEN_DEFAULT		0x0*
11:8	TX_FRAC_CONF_TX_FRAC_NUM	TX_FRAC_CONF_TX_FRAC_NUM_DEFAULT		0x0*
7:4	RX_FRAC_CONF_RX_FRAC_DEN	RX_FRAC_CONF_RX_FRAC_DEN_DEFAULT		0x0*
3:0	RX_FRAC_CONF_RX_FRAC_NUM	RX_FRAC_CONF_RX_FRAC_NUM_DEFAULT		0x0*

5.9.0.19 RF_TX_PULSE_SHAPE_1

Bit Field	Read/Write	Field Name	Description
31:24	RW	TX_PULSE_SHAPE_1_TX_COEF4	Tx pulse shape coefficient 4 (banked)
23:16	RW	TX_PULSE_SHAPE_1_TX_COEF3	Tx pulse shape coefficient 3 (banked)
15:8	RW	TX_PULSE_SHAPE_1_TX_COEF2	Tx pulse shape coefficient 2 (banked)
7:0	RW	TX_PULSE_SHAPE_1_TX_COEF1	Tx pulse shape coefficient 1 (banked)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:24	TX_PULSE_SHAPE_1_TX_COEF4	TX_PULSE_SHAPE_1_TX_COEF4_DEFAULT		0x0*
23:16	TX_PULSE_SHAPE_1_TX_COEF3	TX_PULSE_SHAPE_1_TX_COEF3_DEFAULT		0x0*
15:8	TX_PULSE_SHAPE_1_TX_COEF2	TX_PULSE_SHAPE_1_TX_COEF2_DEFAULT		0x0*
7:0	TX_PULSE_SHAPE_1_TX_COEF1	TX_PULSE_SHAPE_1_TX_COEF1_DEFAULT		0x0*

5.9.0.20 RF_TX_PULSE_SHAPE_2

Bit Field	Read/Write	Field Name	Description
31:24	RW	TX_PULSE_SHAPE_2_TX_COEF8	Tx pulse shape coefficient 8 (banked)
23:16	RW	TX_PULSE_SHAPE_2_TX_COEF7	Tx pulse shape coefficient 7 (banked)
15:8	RW	TX_PULSE_SHAPE_2_TX_COEF6	Tx pulse shape coefficient 6 (banked)
7:0	RW	TX_PULSE_SHAPE_2_TX_COEF5	Tx pulse shape coefficient 5 (banked)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:24	TX_PULSE_SHAPE_2_TX_COEF8	TX_PULSE_SHAPE_2_TX_COEF8_DEFAULT		0x2*
23:16	TX_PULSE_SHAPE_2_TX_COEF7	TX_PULSE_SHAPE_2_TX_COEF7_DEFAULT		0x1*
15:8	TX_PULSE_SHAPE_2_TX_COEF6	TX_PULSE_SHAPE_2_TX_COEF6_DEFAULT		0x0*
7:0	TX_PULSE_SHAPE_2_TX_COEF5	TX_PULSE_SHAPE_2_TX_COEF5_DEFAULT		0x0*

5.9.0.21 RF_TX_PULSE_SHAPE_3

Bit Field	Read/Write	Field Name	Description
31:24	RW	TX_PULSE_SHAPE_3_TX_COEF12	Tx pulse shape coefficient 12 (banked)
23:16	RW	TX_PULSE_SHAPE_3_TX_COEF11	Tx pulse shape coefficient 11 (banked)
15:8	RW	TX_PULSE_SHAPE_3_TX_COEF10	Tx pulse shape coefficient 10 (banked)
7:0	RW	TX_PULSE_SHAPE_3_TX_COEF9	Tx pulse shape coefficient 9 (banked)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:24	TX_PULSE_SHAPE_3_TX_COEF12	TX_PULSE_SHAPE_3_TX_COEF12_DEFAULT		0x36*
23:16	TX_PULSE_SHAPE_3_TX_COEF11	TX_PULSE_SHAPE_3_TX_COEF11_DEFAULT		0x20*
15:8	TX_PULSE_SHAPE_3_TX_COEF10	TX_PULSE_SHAPE_3_TX_COEF10_DEFAULT		0x10*
7:0	TX_PULSE_SHAPE_3_TX_COEF9	TX_PULSE_SHAPE_3_TX_COEF9_DEFAULT		0x7*

5.9.0.22 RF_TX_PULSE_SHAPE_4

Bit Field	Read/Write	Field Name	Description
31:24	RW	TX_PULSE_SHAPE_4_TX_COEF16	Tx pulse shape coefficient 16 (banked)
23:16	RW	TX_PULSE_SHAPE_4_TX_COEF15	Tx pulse shape coefficient 15 (banked)
15:8	RW	TX_PULSE_SHAPE_4_TX_COEF14	Tx pulse shape coefficient 14 (banked)
7:0	RW	TX_PULSE_SHAPE_4_TX_COEF13	Tx pulse shape coefficient 13 (banked)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:24	TX_PULSE_SHAPE_4_TX_COEF16	TX_PULSE_SHAPE_4_TX_COEF16_DEFAULT		0x7D*
23:16	TX_PULSE_SHAPE_4_TX_COEF15	TX_PULSE_SHAPE_4_TX_COEF15_DEFAULT		0x75*
15:8	TX_PULSE_SHAPE_4_TX_COEF14	TX_PULSE_SHAPE_4_TX_COEF14_DEFAULT		0x66*
7:0	TX_PULSE_SHAPE_4_TX_COEF13	TX_PULSE_SHAPE_4_TX_COEF13_DEFAULT		0x4F*

5.9.0.23 RF_FRONTEND

Bit Field	Read/Write	Field Name	Description
25:16	RW	RX_IF_DIG_IF_DIG	IF frequency (banked)
14:11	RW	FRONTEND_RESAMPLE_PH_GAIN	Gain of the phase resampling block (banked)
10:8	RW	FRONTEND_RESAMPLE_RSSI_	Gain of the decimator in the RSSI resampling block (banked)

Bit Field	Read/Write	Field Name	Description
		G2	
7:6	RW	FRONTEND_RESAMPLE_RSSI_G1	Gain of the interpolator in the RSSI resampling block (banked)
5	RW	FRONTEND_EN_RESAMPLE_RSSI	RSSI resampling (banked)
4	RW	FRONTEND_EN_RESAMPLE_PHADC	Phase resampling (banked)
3:0	RW	FRONTEND_DIV_PHADC	Unsigned value that specifies the divider to obtain the phase ADC clock and RSSI (banked)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
25:16	RX_IF_DIG_IF_DIG	RX_IF_DIG_IF_DIG_DEFAULT	Signed value equal to f_IF/f_phADC*1024	0x40*
14:11	FRONTEND_RESAMPLE_PH_GAIN	FRONTEND_RESAMPLE_PH_GAIN_DEFAULT		0x6*
10:8	FRONTEND_RESAMPLE_RSSI_G2	FRONTEND_RESAMPLE_RSSI_G2_DEFAULT		0x0*
7:6	FRONTEND_RESAMPLE_RSSI_G1	FRONTEND_RESAMPLE_RSSI_G1_DEFAULT		0x0*
5	FRONTEND_EN_RESAMPLE_RSSI	FRONTEND_EN_RESAMPLE_RSSI_DISABLE	Disable the RSSI resampling	0x0*
		FRONTEND_EN_RESAMPLE_RSSI_ENABLE	Enable the RSSI resampling	0x1
4	FRONTEND_EN_RESAMPLE_PHADC	FRONTEND_EN_RESAMPLE_PHADC_DISABLE	Disable the phase resampling	0x0
		FRONTEND_EN_RESAMPLE_PHADC_ENABLE	Enable the phase resampling	0x1*
3:0	FRONTEND_DIV_PHADC	FRONTEND_DIV_PHADC_DEFAULT		0x0*

5.9.0.24 RF_RX_PULSE_SHAPE

Bit Field	Read/Write	Field Name	Description
31:28	RW	RX_PULSE_SHAPE_RX_COEF8	Rx pulse shape coefficient 8 (banked)
27:24	RW	RX_PULSE_SHAPE_RX_COEF7	Rx pulse shape coefficient 7 (banked)
23:20	RW	RX_PULSE_SHAPE_RX_COEF6	Rx pulse shape coefficient 6 (banked)

Bit Field	Read/Write	Field Name	Description
19:16	RW	RX_PULSE_SHAPE_RX_COEF5	Rx pulse shape coefficient 5 (banked)
15:12	RW	RX_PULSE_SHAPE_RX_COEF4	Rx pulse shape coefficient 4 (banked)
11:8	RW	RX_PULSE_SHAPE_RX_COEF3	Rx pulse shape coefficient 3 (banked)
7:4	RW	RX_PULSE_SHAPE_RX_COEF2	Rx pulse shape coefficient 2 (banked)
3:0	RW	RX_PULSE_SHAPE_RX_COEF1	Rx pulse shape coefficient 1 (banked)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:28	RX_PULSE_SHAPE_RX_COEF8	RX_PULSE_SHAPE_RX_COEF8_DEFAULT		0xF*
27:24	RX_PULSE_SHAPE_RX_COEF7	RX_PULSE_SHAPE_RX_COEF7_DEFAULT		0xE*
23:20	RX_PULSE_SHAPE_RX_COEF6	RX_PULSE_SHAPE_RX_COEF6_DEFAULT		0xC*
19:16	RX_PULSE_SHAPE_RX_COEF5	RX_PULSE_SHAPE_RX_COEF5_DEFAULT		0xA*
15:12	RX_PULSE_SHAPE_RX_COEF4	RX_PULSE_SHAPE_RX_COEF4_DEFAULT		0x7*
11:8	RX_PULSE_SHAPE_RX_COEF3	RX_PULSE_SHAPE_RX_COEF3_DEFAULT		0x4*
7:4	RX_PULSE_SHAPE_RX_COEF2	RX_PULSE_SHAPE_RX_COEF2_DEFAULT		0x2*
3:0	RX_PULSE_SHAPE_RX_COEF1	RX_PULSE_SHAPE_RX_COEF1_DEFAULT		0x1*

5.9.0.25 RF_REG18

Bit Field	Read/Write	Field Name	Description
28	RW	DELAY_LINE_CONF_MULTI_SYNC	Detect multiple syncs (banked)
27:25	RW	DELAY_LINE_CONF_DL_ISI_THR	Threshold bias for ISI compensation in the delay line sync word comparator (banked)
22	RW	DELAY_LINE_CONF_EN_SYNC_OK_DELAY_LINE	Use pattern_ok signal in delay line to synchronize the deserializer (banked)
21:20	RW	DELAY_LINE_CONF_MAX_ERR_IN_DL_SYNC	Set the maximum errors in the delay line sync detection (banked)
19	RW	DELAY_LINE_CONF_EN_NOT_CAUSAL	Non causal processing (banked)

Bit Field	Read/Write	Field Name	Description
18:16	RW	DELAY_LINE_CONF_NC_SEL_OUT	Select the output position for the non causal processing (banked)
15:8	RW	FSK_FCR_AMP_1_FSK_FCR_AMP1	FSK amplitude low (banked)
6:4	RW	CARRIER_RECOVERY_EXTRA_FREQ_LIMIT_MAN	Mantissa of the carrier recovery frequency limit (banked)
2:0	RW	CARRIER_RECOVERY_EXTRA_FREQ_LIMIT_EXP	Exponent of the carrier recovery frequency limit (banked)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
28	DELAY_LINE_CONF_MULTI_SYNC	DELAY_LINE_CONF_MULTI_SYNC_DISABLE	The delay line detects only a single sync	0x0*
		DELAY_LINE_CONF_MULTI_SYNC_ENABLE	The delay line detects multiple syncs	0x1
27:25	DELAY_LINE_CONF_DL_ISI_THR	DELAY_LINE_CONF_DL_ISI_THR_DEFAULT		0x1*
22	DELAY_LINE_CONF_EN_SYNC_OK_DELAY_LINE	DELAY_LINE_CONF_EN_SYNC_OK_DELAY_LINE_DISABLE	Don't use the pattern_ok signal in delay line to synchronize the deserializer	0x0
		DELAY_LINE_CONF_EN_SYNC_OK_DELAY_LINE_ENABLE	Use the pattern_ok signal in delay line to synchronize the deserializer	0x1*
21:20	DELAY_LINE_CONF_MAX_ERR_IN_DL_SYNC	DELAY_LINE_CONF_MAX_ERR_IN_DL_SYNC_DEFAULT		0x0*
19	DELAY_LINE_CONF_EN_NOT_CAUSAL	DELAY_LINE_CONF_EN_NOT_CAUSAL_DISABLE	Disable the non causal processing	0x0*
		DELAY_LINE_CONF_EN_NOT_CAUSAL_ENABLE	Enable the non causal processing	0x1
18:16	DELAY_LINE_CONF_NC_SEL_OUT	DELAY_LINE_CONF_NC_SEL_OUT_4	4 symbols	0x0*
		DELAY_LINE_CONF_NC_SEL_OUT_6	6 symbols	0x1
		DELAY_LINE_CONF_NC_SEL_OUT_8	8 symbols	0x2
		DELAY_LINE_CONF_NC_SEL_OUT_12	12 symbols	0x3
		DELAY_LINE_CONF_NC_SEL_OUT_16	16 symbols	0x4

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		DELAY_LINE_CONF_NC_SEL_OUT_24	24 symbols	0x5
		DELAY_LINE_CONF_NC_SEL_OUT_32	32 symbols	0x6
		DELAY_LINE_CONF_NC_SEL_OUT_40	40 symbols	0x7
15:8	FSK_FCR_AMP_1_FSK_FCR_AMP1	FSK_FCR_AMP_1_FSK_FCR_AMP1_DEFAULT		0x1B*
6:4	CARRIER_RECOVERY_EXTRA_FREQ_LIMIT_MAN	CARRIER_RECOVERY_EXTRA_FREQ_LIMIT_MAN_DEFAULT	Unsigned value	0x5*
2:0	CARRIER_RECOVERY_EXTRA_FREQ_LIMIT_EXP	CARRIER_RECOVERY_EXTRA_FREQ_LIMIT_EXP_DEFAULT	Signed value	0x0*

5.9.0.26 RF_REG19

Bit Field	Read/Write	Field Name	Description
30	RW	RSSI_BANK_EN_RSSI_DITHER	Speed on the RSSI triangular dithering signal (banked)
29	RW	RSSI_BANK_FAST_RSSI	RSSI filtering speed (banked)
28	RW	RSSI_BANK_EN_FAST_PRE_SYNC	Fast mode switching during the preamble reception (banked)
27:24	RW	RSSI_BANK_TAU_RSSI_FILTERING	Time constant of the RSSI filtering block (banked)
20	RW	DECISION_USE_VIT_SOFT	Viterbi soft decoding (banked)
19:18	RW	DECISION_VITERBI_LEN	Set the Viterbi path length (banked)
17	RW	DECISION_VITERBI_POW_NLIN	Viterbi algorithm uses power instead of amplitude to evaluate the error on the path (banked)
16	RW	DECISION_EN_VITERBI_GFSK	Viterbi algorithm for the GFSK decoding (banked)
15:8	RW	FSK_FCR_AMP_3_FSK_FCR_AMP3	FSK amplitude high (banked)
7:0	RW	FSK_FCR_AMP_2_FSK_FCR_AMP2	FSK amplitude mid (banked)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
30	RSSI_BANK_EN_RSSI_DITHER	RSSI_BANK_EN_RRSI_DITHER_DISABLE	Disable the RSSI filtering (use minimum value of RSSI even)	0x0*
		RSSI_BANK_EN_RRSI_DITHER_ENABLE	Enable the RSSI filtering	0x1
29	RSSI_BANK_FAST_RSSI	RSSI_BANK_FAST_RSSI_NORMAL	Normal RSSI filtering	0x0*
		RSSI_BANK_FAST_RSSI_FAST	Fast RSSI filtering (8x faster)	0x1
28	RSSI_BANK_EN_FAST_PRE_SYNC	RSSI_BANK_EN_FAST_PRE_SYNC_DISABLE	Don't switch the fast modes during the preamble reception	0x0
		RSSI_BANK_EN_FAST_PRE_SYNC_ENABLE	Switch the fast modes during the preamble reception	0x1*
27:24	RSSI_BANK_TAU_RSSI_FILTERING	RSSI_BANK_TAU_RSSI_FILTERING_4	4 symbols	0x0
		RSSI_BANK_TAU_RSSI_FILTERING_8	8 symbols	0x1*
		RSSI_BANK_TAU_RSSI_FILTERING_16	16 symbols	0x2
		RSSI_BANK_TAU_RSSI_FILTERING_32	32 symbols	0x3
		RSSI_BANK_TAU_RSSI_FILTERING_64	64 symbols	0x4
		RSSI_BANK_TAU_RSSI_FILTERING_128	128 symbols	0x5
		RSSI_BANK_TAU_RSSI_FILTERING_256	256 symbols	0x6
		RSSI_BANK_TAU_RSSI_FILTERING_512	512 symbols	0x7
		RSSI_BANK_TAU_RSSI_FILTERING_1024	1024 symbols	0x8
20	DECISION_USE_VIT_SOFT	DECISION_USE_VIT_SOFT_DISABLE	Don't use the Viterbi soft decoding	0x0*
		DECISION_USE_VIT_SOFT_ENABLE	Use the Viterbi soft decoding	0x1
19:18	DECISION_VITERBI_LEN	DECISION_VITERBI_LEN_1	1 bit	0x0
		DECISION_VITERBI_LEN_2	2 bits	0x1
		DECISION_VITERBI_LEN_4	4 bits	0x2*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		DECISION_VITERBI_LEN_8	8 bits	0x3
17	DECISION_VITERBI_POW_NLIN	DECISION_VITERBI_POW_NLIN_DISABLE	Viterbi algorithm uses amplitude to evaluate the error on the path	0x0
		DECISION_VITERBI_POW_NLIN_ENABLE	Viterbi algorithm uses power to evaluate the error on the path	0x1*
16	DECISION_EN_VITERBI_GFSK	DECISION_EN_VITERBI_GFSK_DISABLE	Disable the Viterbi algorithm for the GFSK decoding	0x0
		DECISION_EN_VITERBI_GFSK_ENABLE	Enable the Viterbi algorithm for the GFSK decoding	0x1*
15:8	FSK_FCR_AMP_3_FSK_FCR_AMP3	FSK_FCR_AMP_3_FSK_FCR_AMP3_DEFAULT	In 4FSK is the high amplitude (+/-3), in FSK w/ ISI it specifies the highest amplitude (generally it corresponds to a sequence 1-1-1)	0x44*
7:0	FSK_FCR_AMP_2_FSK_FCR_AMP2	FSK_FCR_AMP_2_FSK_FCR_AMP2_DEFAULT	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)	0x30*

5.9.0.27 RF_REG1A

Bit Field	Read/Write	Field Name	Description
28:24	RW	PA_PWR_PA_PWR	Signed value that sets the PA power
22	RW	RSSI_BANK_ALT_USE_RSSI_ALT	Use alternative RRSI configuration (banked)
21	RW	RSSI_BANK_ALT_FAST_RSSI_ALT	RSSI filtering speed (banked)
19:16	RW	RSSI_BANK_ALT_TAU_RSSI_FILTERING_ALT	Time constant of the RSSI filtering block (banked)
15:0	RW	CORRECT_CFREQ_IF_CORRECT_CFREQ_IF	Unsigned value that specifies the IF for the Rx mode (banked)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
28:24	PA_PWR_PA_PWR	PA_PWR_PA_PWR_DEFAULT		0xC*
22	RSSI_BANK_ALT_USE_RSSI_ALT	RSSI_BANK_ALT_USE_RSSI_INITIAL	RSSI_AVG, RSSI_MAX and RSSI_MIN will point to initial RSSI configuration	0x0*
		RSSI_BANK_ALT_USE_RSSI_ALTERNATE	RSSI_AVG, RSSI_MAX and RSSI_MIN will point to alternative RSSI	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
			configuration	
21	RSSI_BANK_ALT_ FAST_RSSI_ALT	RSSI_BANK_ALT_FAST_RSSI_ NORMAL	Normal RSSI filtering	0x0*
		RSSI_BANK_ALT_FAST_RSSI_ FAST	Fast RSSI filtering (8x faster)	0x1
19:16	RSSI_BANK_ALT_TAU_ RSSI_FILTERING_ALT	RSSI_BANK_ALT_TAU_RSSI_ FILTERING_ALT_4	4 symbols	0x0
		RSSI_BANK_ALT_TAU_RSSI_ FILTERING_ALT_8	8 symbols	0x1
		RSSI_BANK_ALT_TAU_RSSI_ FILTERING_ALT_16	16 symbols	0x2
		RSSI_BANK_ALT_TAU_RSSI_ FILTERING_ALT_32	32 symbols	0x3*
		RSSI_BANK_ALT_TAU_RSSI_ FILTERING_ALT_64	64 symbols	0x4
		RSSI_BANK_ALT_TAU_RSSI_ FILTERING_ALT_128	128 symbols	0x5
		RSSI_BANK_ALT_TAU_RSSI_ FILTERING_ALT_256	256 symbols	0x6
		RSSI_BANK_ALT_TAU_RSSI_ FILTERING_ALT_512	512 symbols	0x7
		RSSI_BANK_ALT_TAU_RSSI_ FILTERING_ALT_1024	1024 symbols	0x8
15:0	CORRECT_CFREQ_IF_ CORRECT_CFREQ_IF	CORRECT_CFREQ_IF_CORRECT_ CFREQ_IF_DEFAULT		0x1555*

5.9.0.28 RF_REG1B

Bit Field	Read/Write	Field Name	Description
31	RW	PLL_BANK_EN_LOW_CHP_ BIAS_TX	Set the en_low_chp_bias bit in Tx mode (banked)
30	RW	PLL_BANK_EN_LOW_CHP_ BIAS_RX	Set the en_low_chp_bias bit in Rx mode (banked)
29:28	RW	PLL_BANK_PLL_FILTER_RES_ TRIM_TX	Modify the value of the loop filter resistor R2 when bit 5 is high in Tx mode (banked)
27:24	RW	PLL_BANK_IQ_PLL_0_TX	Charge pump bias for Tx case (banked)

Bit Field	Read/Write	Field Name	Description
22	RW	PLL_BANK_LOW_DR_TX	Enable low data-rate mode in Tx mode (banked)
21:20	RW	PLL_BANK_PLL_FILTER_RES_TRIM_RX	Modify the value of the loop filter resistor R2 when bit 5 is high in Rx mode (banked)
19:16	RW	PLL_BANK_IQ_PLL_0_RX	Charge pump bias for Rx (banked)
15	RW	ANACLK_USE_NEW_ANACK	Use the new analog clock generator (banked)
13:12	RW	ANACLK_DIV_CK_RSSI	Set the master clock divider for the RSSI clock (banked)
11:10	RW	ANACLK_DIV_CK_FILT	Set the master clock divider for the channel filter clock (banked)
9:8	RW	ANACLK_DIV_CK_PHADC	Set the master clock divider for the phase ADC clock (banked)
7:4	RW	ANACLK_DIV_RSSI	Unsigned value that specifies the division factor for the clock controlling the RSSI (banked)
3:0	RW	ANACLK_DIV_FILT	Unsigned value that specifies the division factor for the clock controlling the channel filter (banked)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31	PLL_BANK_EN_LOW_CHP_BIAS_TX	PLL_BANK_EN_LOW_CHP_BIAS_TX_RESET	In Tx mode EN_LOW_CHP_BIAS bit is reset	0x0*
		PLL_BANK_EN_LOW_CHP_BIAS_TX_SET	In Tx mode EN_LOW_CHP_BIAS bit is set	0x1
30	PLL_BANK_EN_LOW_CHP_BIAS_RX	PLL_BANK_EN_LOW_CHP_BIAS_RX_RESET	In Rx mode EN_LOW_CHP_BIAS bit is reset	0x0
		PLL_BANK_EN_LOW_CHP_BIAS_RX_SET	In Rx mode EN_LOW_CHP_BIAS bit is set	0x1*
29:28	PLL_BANK_PLL_FILTER_RES_TRIM_TX	PLL_BANK_PLL_FILTER_RES_TRIM_TX_0	Normal resistor	0x0
		PLL_BANK_PLL_FILTER_RES_TRIM_TX_1	Normal resistor + 23%	0x1
		PLL_BANK_PLL_FILTER_RES_TRIM_TX_2	Normal resistor + 30%	0x2
		PLL_BANK_PLL_FILTER_RES_TRIM_TX_3	Normal resistor + 70%	0x3*
27:24	PLL_BANK_IQ_PLL_0_TX	PLL_BANK_IQ_PLL_0_TX_DEFAULT		0x4*
22	PLL_BANK_LOW_DR_TX	PLL_BANK_LOW_DR_TX_DISABLE	Tx works not in low data rate	0x0*
		PLL_BANK_LOW_DR_TX_ENABLE	Tx works in low data rate	0x1
21:20	PLL_BANK_PLL_	PLL_BANK_PLL_FILTER_RES_	Normal resistor	0x0*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
	FILTER_RES_TRIM_RX	TRIM_RX_0		
		PLL_BANK_PLL_FILTER_RES_TRIM_RX_1	Normal resistor + 23%	0x1
		PLL_BANK_PLL_FILTER_RES_TRIM_RX_2	Normal resistor + 30%	0x2
		PLL_BANK_PLL_FILTER_RES_TRIM_RX_3	Normal resistor + 70%	0x3
19:16	PLL_BANK_IQ_PLL_0_RX	PLL_BANK_IQ_PLL_0_RX_DEFAULT		0xB*
15	ANACLK_USE_NEW_ANACK	ANACLK_USE_NEW_ANACK_DISABLE	Do not use the analog clock generator	0x0*
		ANACLK_USE_NEW_ANACK_ENABLE	Use the analog clock generator	0x1
13:12	ANACLK_DIV_CK_RSSI	ANACLK_DIV_CK_RSSI_3	Division by 3	0x0*
		ANACLK_DIV_CK_RSSI_2	Division by 2	0x1
		ANACLK_DIV_CK_RSSI_1	Division by 1	0x2
11:10	ANACLK_DIV_CK_FILT	ANACLK_DIV_CK_FILT_3	Division by 3	0x0*
		ANACLK_DIV_CK_FILT_2	Division by 2	0x1
		ANACLK_DIV_CK_FILT_1	Division by 1	0x2
9:8	ANACLK_DIV_CK_PHADC	ANACLK_DIV_CK_PHADC_3	Division by 3	0x0*
		ANACLK_DIV_CK_PHADC_2	Division by 2	0x1
		ANACLK_DIV_CK_PHADC_1	Division by 1	0x2
7:4	ANACLK_DIV_RSSI	ANACLK_DIV_RSSI_DEFAULT		0x1*
3:0	ANACLK_DIV_FILT	ANACLK_DIV_FILT_DEFAULT		0x5*

5.9.0.29 RF_RSSI_CTRL

Bit Field	Read/Write	Field Name	Description
31:30	RW	RSSI_CTRL_AGC_DECAY_TAU	Time constant of the decay speed
29	RW	RSSI_CTRL_AGC_USE_LNA	AGC algorithm uses LNA bias
28	RW	RSSI_CTRL_AGC_MODE	AGC algorithm selection
27:26	RW	RSSI_CTRL_AGC_WAIT	Set the wait time of the AGC after switching between state
25	RW	RSSI_CTRL_PAYLOAD_BLOCKS_AGC	AGC payload blocking

Bit Field	Read/Write	Field Name	Description
24	RW	RSSI_CTRL_BYPASS_AGC	AGC algorithm bypass
20:16	RW	PA_PWR_OFFSET_PA_PWR_OFFSET	Signed value that sets the PA power (banked)
12:8	RW	FILTER_BIAS_IQ_FI_BW	Bias for the bandwidth of the channel filter (banked)
4:0	RW	FILTER_BIAS_IQ_FI_FC	Bias for the central frequency of the channel filter (banked)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:30	RSSI_CTRL_AGC_DECAY_TAU	RSSI_CTRL_AGC_DECAY_TAU_DEFAULT	High values corresponds to a slow decay	0x3*
29	RSSI_CTRL_AGC_USE_LNA	RSSI_CTRL_AGC_USE_LNA_DISABLE	AGC algorithm doesn't use the LNA bias	0x0
		RSSI_CTRL_AGC_USE_LNA_ENABLE	AGC algorithm uses the LNA bias	0x1*
28	RSSI_CTRL_AGC_MODE	RSSI_CTRL_AGC_MODE_DISABLE	Old AGC algorithm	0x0
		RSSI_CTRL_AGC_MODE_ENABLE	New AGC algorithm	0x1*
27:26	RSSI_CTRL_AGC_WAIT	RSSI_CTRL_AGC_WAIT_0	Don't wait	0x0
		RSSI_CTRL_AGC_WAIT_1	Wait 1x RSSI filtering period	0x1
		RSSI_CTRL_AGC_WAIT_2	Wait 2x RSSI filtering period	0x2
		RSSI_CTRL_AGC_WAIT_3	Wait 3x RSSI filtering period	0x3*
25	RSSI_CTRL_PAYLOAD_BLOCKS_AGC	RSSI_CTRL_PAYLOAD_BLOCKS_AGC_DISABLE	AGC is not blocked during the payload	0x0
		RSSI_CTRL_PAYLOAD_BLOCKS_AGC_ENABLE	AGC is blocked during the payload	0x1*
24	RSSI_CTRL_BYPASS_AGC	RSSI_CTRL_BYPASS_AGC_DISABLE	AGC algorithm is not bypassed	0x0*
		RSSI_CTRL_BYPASS_AGC_ENABLE	AGC algorithm is bypassed	0x1
20:16	PA_PWR_OFFSET_PA_PWR_OFFSET	PA_PWR_OFFSET_PA_PWR_OFFSET_DEFAULT		0x0*
12:8	FILTER_BIAS_IQ_FI_BW	FILTER_BIAS_IQ_FI_BW_DEFAULT		0x14*
4:0	FILTER_BIAS_IQ_FI_FC	FILTER_BIAS_IQ_FI_FC_DEFAULT		0xB*

5.9.0.30 RF_REG1D

Bit Field	Read/Write	Field Name	Description
31:28	RW	AGC_PEAK_DET_PEAK_DET_TAU	Time constant of the peak detector monostable circuit
27:26	RW	AGC_PEAK_DET_PEAK_DET_THR_LOW	Threshold for the low level of the peak detector
25	RW	AGC_PEAK_DET_PEAK_DET_THR_HIGH	Threshold for the high level of the peak detector
24	RW	AGC_PEAK_DET_EN_AGC_PEAK	Enable AGC peak detector
23:16	RW	AGC_THR_HIGH_AGC_THR_HIGH	AGC threshold high level (banked)
15:8	RW	AGC_THR_LOW_AGC_THR_LOW	AGC threshold low level (banked)
7:4	RW	ATT_CTRL_ATT_CTRL_MAX	Maximum attenuation level in AGC algorithm
3:0	RW	ATT_CTRL_SET_RX_ATT_CTRL	Attenuation level if the AGC is bypassed

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:28	AGC_PEAK_DET_PEAK_DET_TAU	AGC_PEAK_DET_PEAK_DET_TAU_DEFAULT		0x7*
27:26	AGC_PEAK_DET_PEAK_DET_THR_LOW	AGC_PEAK_DET_PEAK_DET_THR_LOW_DEFAULT	0 => 0, 1 => 1, 2 => 2, 3 => N.A.	0x0*
25	AGC_PEAK_DET_PEAK_DET_THR_HIGH	AGC_PEAK_DET_PEAK_DET_THR_HIGH_0	Threshold 2 for the high level of the peak detector	0x0*
		AGC_PEAK_DET_PEAK_DET_THR_HIGH_1	Threshold 3 for the high level of the peak detector	0x1
24	AGC_PEAK_DET_EN_AGC_PEAK	AGC_PEAK_DET_EN_AGC_PEAK_DISABLE	Disable the AGC peak detector	0x0
		AGC_PEAK_DET_EN_AGC_PEAK_ENABLE	Enable the AGC peak detector	0x1*
23:16	AGC_THR_HIGH_AGC_THR_HIGH	AGC_THR_HIGH_AGC_THR_HIGH_DEFAULT		0x69*
15:8	AGC_THR_LOW_AGC_THR_LOW	AGC_THR_LOW_AGC_THR_LOW_DEFAULT		0x40*
7:4	ATT_CTRL_ATT_CTRL_MAX	ATT_CTRL_ATT_CTRL_MAX_DEFAULT		0xB*
3:0	ATT_CTRL_SET_RX_ATT_CTRL	ATT_CTRL_SET_RX_ATT_CTRL_DEFAULT		0x0*

5.9.0.31 RF_AGC_LUT1

Bit Field	Read/Write	Field Name	Description
31:22	RW	AGC_LUT_1_AGC_LEVEL_2_LO	AGC values level 2 (LSB)
21:11	RW	AGC_LUT_1_AGC_LEVEL_1	AGC values level 1
10:0	RW	AGC_LUT_1_AGC_LEVEL_0	AGC values level 0

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:22	AGC_LUT_1_AGC_LEVEL_2_LO	AGC_LUT_1_AGC_LEVEL_2_LO_DEFAULT		0x280*
21:11	AGC_LUT_1_AGC_LEVEL_1	AGC_LUT_1_AGC_LEVEL_1_DEFAULT		0x80*
10:0	AGC_LUT_1_AGC_LEVEL_0	AGC_LUT_1_AGC_LEVEL_0_DEFAULT		0x0*

5.9.0.32 RF_AGC_LUT2

Bit Field	Read/Write	Field Name	Description
31:23	RW	AGC_LUT_2_AGC_LEVEL_5_LO	AGC values level 5 (LSB)
22:12	RW	AGC_LUT_2_AGC_LEVEL_4	AGC values level 4
11:1	RW	AGC_LUT_2_AGC_LEVEL_3	AGC values level 3
0	RW	AGC_LUT_2_AGC_LEVEL_2_HI	AGC values level 2 (MSB)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:23	AGC_LUT_2_AGC_LEVEL_5_LO	AGC_LUT_2_AGC_LEVEL_5_LO_DEFAULT		0x84*
22:12	AGC_LUT_2_AGC_LEVEL_4	AGC_LUT_2_AGC_LEVEL_4_DEFAULT		0x284*
11:1	AGC_LUT_2_AGC_LEVEL_3	AGC_LUT_2_AGC_LEVEL_3_DEFAULT		0x480*
0	AGC_LUT_2_AGC_LEVEL_2_HI	AGC_LUT_2_AGC_LEVEL_2_HI_DEFAULT		0x0*

5.9.0.33 RF_AGC_LUT3

Bit Field	Read/Write	Field Name	Description
31:24	RW	AGC_LUT_3_AGC_LEVEL_8_LO	AGC values level 8 (LSB)
23:13	RW	AGC_LUT_3_AGC_LEVEL_7	AGC values level 7
12:2	RW	AGC_LUT_3_AGC_LEVEL_6	AGC values level 6
1:0	RW	AGC_LUT_3_AGC_LEVEL_5_HI	AGC values level 5 (MSB)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:24	AGC_LUT_3_AGC_LEVEL_8_LO	AGC_LUT_3_AGC_LEVEL_8_LO_DEFAULT		0x9D*
23:13	AGC_LUT_3_AGC_LEVEL_7	AGC_LUT_3_AGC_LEVEL_7_DEFAULT		0x495*
12:2	AGC_LUT_3_AGC_LEVEL_6	AGC_LUT_3_AGC_LEVEL_6_DEFAULT		0x485*
1:0	AGC_LUT_3_AGC_LEVEL_5_HI	AGC_LUT_3_AGC_LEVEL_5_HI_DEFAULT		0x2*

5.9.0.34 RF_AGC_LUT4

Bit Field	Read/Write	Field Name	Description
31:25	RW	AGC_LUT_4_AGC_LEVEL_11_LO	AGC values level 11 (LSB)
24:14	RW	AGC_LUT_4_AGC_LEVEL_10	AGC values level 10
13:3	RW	AGC_LUT_4_AGC_LEVEL_9	AGC values level 9
2:0	RW	AGC_LUT_4_AGC_LEVEL_8_HI	AGC values level 8 (MSB)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:25	AGC_LUT_4_AGC_LEVEL_11_LO	AGC_LUT_4_AGC_LEVEL_11_LO_DEFAULT		0x7F*
24:14	AGC_LUT_4_AGC_LEVEL_10	AGC_LUT_4_AGC_LEVEL_10_DEFAULT		0x4FF*
13:3	AGC_LUT_4_AGC_LEVEL_9	AGC_LUT_4_AGC_LEVEL_9_DEFAULT		0x49F*
2:0	AGC_LUT_4_AGC_LEVEL_8_HI	AGC_LUT_4_AGC_LEVEL_8_HI_DEFAULT		0x4*

5.9.0.35 RF_AGC_LUT5

Bit Field	Read/Write	Field Name	Description
26:25	RW	IEEE802154_OPTS_CNT_LIM_802154	Set the number of samples to wait before increasing the threshold
24:22	RW	IEEE802154_OPTS_CNT_OK_INC_802154	Set the increment to the counter that indicates that the correlators peaks are coherent
21	RW	IEEE802154_OPTS_USE_OS_802154	Enable the new algorithm working in the oversampled domain for the demodulation of the IEEE 802.15.4 protocol
20	RW	IEEE802154_OPTS_EN_DW_TEST	Tx data-whitening before the convolutional code block
18:16	RW	IEEE802154_OPTS_C2B_THR	Threshold of the chip2bit correlator of the IEEE 802.15.4 protocol
13:12	RW	DATA_STREAMS_BER_CLK_MODE	Set the clock output mode for BER mode or RW mode
10	RW	DATA_STREAMS_RX_DATA_NOT_SAMPLED	Signal rx_data in test modes sampling
9	RW	DATA_STREAMS_PHASE_GREY	Phase signal encoding
8	RW	DATA_STREAMS_TX_IN_CLK_TOGGLE	Input clock
3:0	RW	AGC_LUT_5_AGC_LEVEL_11_HI	AGC values level 11 (MSB)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
26:25	IEEE802154_OPTS_CNT_LIM_802154	IEEE802154_OPTS_CNT_LIM_802154_DEFAULT		0x2*
24:22	IEEE802154_OPTS_CNT_OK_INC_802154	IEEE802154_OPTS_CNT_OK_INC_802154_DEFAULT		0x4*
21	IEEE802154_OPTS_USE_OS_802154	IEEE802154_OPTS_USE_OS_802154_DISABLE	Disable the Tx data-whitening	0x0
		IEEE802154_OPTS_USE_OS_802154_ENABLE	Enable the Tx data-whitening	0x1*
20	IEEE802154_OPTS_EN_DW_TEST	IEEE802154_OPTS_EN_DW_TEST_DISABLE	Disable the Tx data-whitening	0x0*
		IEEE802154_OPTS_EN_DW_TEST_ENABLE	Enable the Tx data-whitening	0x1
18:16	IEEE802154_OPTS_C2B_THR	IEEE802154_OPTS_C2B_THR_DEFAULT		0x4*
13:12	DATA_STREAMS_BER_	DATA_STREAMS_BER_CLK_MODE_	data change on falling edge	0x0*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
	CLK_MODE	FALLING		
		DATA_STREAMS_BER_CLK_MODE_RISING	Data change on rising edge	0x1
		DATA_STREAMS_BER_CLK_MODE_TOGGLE	Clock signal is a toggled signal	0x2
		DATA_STREAMS_BER_CLK_MODE_RECOVERY	Enable signal from clock recovery	0x3
10	DATA_STREAMS_RX_DATA_NOT_SAMPLED	DATA_STREAMS_RX_DATA_NOT_SAMPLED_DISABLE	The signal rx_data in test modes is sampled	0x0*
		DATA_STREAMS_RX_DATA_NOT_SAMPLED_ENABLE	The signal rx_data in test modes is not sampled	0x1
9	DATA_STREAMS_PHASE_GREY	DATA_STREAMS_PHASE_GREY_DISABLE	The phase signal on the test bus is not grey encoded	0x0*
		DATA_STREAMS_PHASE_GREY_ENABLE	The phase signal on the test bus is grey encoded	0x1
8	DATA_STREAMS_TX_IN_CLK_TOGGLE	DATA_STREAMS_TX_IN_CLK_TOGGLE_DISABLE	Input clock is not a toggling signal	0x0*
		DATA_STREAMS_TX_IN_CLK_TOGGLE_ENABLE	input clock is a toggling signal	0x1
3:0	AGC_LUT_5_AGC_LEVEL_11_HI	AGC_LUT_5_AGC_LEVEL_11_HI_DEFAULT		0xE*

5.9.0.36 RF_AGC_ATT1

Bit Field	Read/Write	Field Name	Description
31:30	RW	AGC_ATT_1_AGC_ATT_AB_LO	AGC attenuation step 10/11 (LSB)
29:27	RW	AGC_ATT_1_AGC_ATT_9A	AGC attenuation step 9/10
26:24	RW	AGC_ATT_1_AGC_ATT_89	AGC attenuation step 8/9
23:21	RW	AGC_ATT_1_AGC_ATT_78	AGC attenuation step 7/8
20:18	RW	AGC_ATT_1_AGC_ATT_67	AGC attenuation step 6/7
17:15	RW	AGC_ATT_1_AGC_ATT_56	AGC attenuation step 5/6
14:12	RW	AGC_ATT_1_AGC_ATT_45	AGC attenuation step 4/5
11:9	RW	AGC_ATT_1_AGC_ATT_34	AGC attenuation step 3/4
8:6	RW	AGC_ATT_1_AGC_ATT_23	AGC attenuation step 2/3
5:3	RW	AGC_ATT_1_AGC_ATT_12	AGC attenuation step 1/2
2:0	RW	AGC_ATT_1_AGC_ATT_01	AGC attenuation step 0/1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:30	AGC_ATT_1_AGC_ATT_AB_LO	AGC_ATT_1_AGC_ATT_AB_LO_DEFAULT		0x3*
29:27	AGC_ATT_1_AGC_ATT_9A	AGC_ATT_1_AGC_ATT_9A_DEFAULT		0x5*
26:24	AGC_ATT_1_AGC_ATT_89	AGC_ATT_1_AGC_ATT_89_DEFAULT		0x3*
23:21	AGC_ATT_1_AGC_ATT_78	AGC_ATT_1_AGC_ATT_78_DEFAULT		0x4*
20:18	AGC_ATT_1_AGC_ATT_67	AGC_ATT_1_AGC_ATT_67_DEFAULT		0x3*
17:15	AGC_ATT_1_AGC_ATT_56	AGC_ATT_1_AGC_ATT_56_DEFAULT		0x2*
14:12	AGC_ATT_1_AGC_ATT_45	AGC_ATT_1_AGC_ATT_45_DEFAULT		0x2*
11:9	AGC_ATT_1_AGC_ATT_34	AGC_ATT_1_AGC_ATT_34_DEFAULT		0x2*
8:6	AGC_ATT_1_AGC_ATT_23	AGC_ATT_1_AGC_ATT_23_DEFAULT		0x1*
5:3	AGC_ATT_1_AGC_ATT_12	AGC_ATT_1_AGC_ATT_12_DEFAULT		0x1*
2:0	AGC_ATT_1_AGC_ATT_01	AGC_ATT_1_AGC_ATT_01_DEFAULT		0x4*

5.9.0.37 RF_AGC_ATT2

Bit Field	Read/Write	Field Name	Description
31:28	RW	TIMINGS_3_T_DLL	Time needed by the DLL blocks to switch on
27:24	RW	TIMINGS_3_T_PLL_TX	Time needed by the PLL blocks in Tx mode to switch on
23:20	RW	TIMINGS_2_T_SUBBAND_TX	Time needed by the subband algorithm to calibrate in Tx mode
19:16	RW	TIMINGS_2_T_TX_RF	Time needed by the RF blocks to switch on in Tx mode
14:12	RW	TIMINGS_1_T_GRANULARITY_TX	Define the granularity of the timer in Tx mode
10:8	RW	TIMINGS_1_T_GRANULARITY_RX	Define the granularity of the timer in Rx mode
1	RW	AGC_ATT_2_AGC_ATT_1DB	Attenuation steps
0	RW	AGC_ATT_2_AGC_ATT_AB_HI	AGC attenuation step 10/11 (MSB)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:28	TIMINGS_3_T_DLL	TIMINGS_3_T_DLL_DEFAULT		0x2*
27:24	TIMINGS_3_T_PLL_TX	TIMINGS_3_T_PLL_TX_DEFAULT		0x2*
23:20	TIMINGS_2_T_ SUBBAND_TX	TIMINGS_2_T_SUBBAND_TX_ DEFAULT		0xC*
19:16	TIMINGS_2_T_TX_RF	TIMINGS_2_T_TX_RF_DEFAULT		0x1*
14:12	TIMINGS_1_T_ GRANULARITY_TX	TIMINGS_1_T_GRANULARITY_ TX_DEFAULT	The granularity is given by (2^(t_ granularity-2)) x 1us	0x3*
10:8	TIMINGS_1_T_ GRANULARITY_RX	TIMINGS_1_T_GRANULARITY_ RX_DEFAULT	The granularity is given by (2^(t_ granularity)) x 1us)	0x5*
1	AGC_ATT_2_AGC_ATT_ 1DB	AGC_ATT_2_AGC_ATT_1DB_ DISABLE	Attenuation is not specified by 1dB steps from 4dB to 11dB	0x0*
		AGC_ATT_2_AGC_ATT_1DB_ ENABLE	Attenuation is specified by 1dB steps from 4dB to 11dB	0x1
0	AGC_ATT_2_AGC_ATT_ AB_HI	AGC_ATT_2_AGC_ATT_AB_HI_ DISABLE	AGC_ATT_2_AGC_ATT_AB MSB is 0	0x0
		AGC_ATT_2_AGC_ATT_AB_HI_ ENABLE	AGC_ATT_2_AGC_ATT_AB MSB is 1	0x1*

5.9.0.38 RF_REG25

Bit Field	Read/Write	Field Name	Description
31	RW	TIMEOUT_EN_RX_TIMEOUT	Timeout of the Rx when the system is on FSM mode
30:28	RW	TIMEOUT_T_TIMEOUT_GR	Granularity of the timer in timeout Rx mode
27:24	RW	TIMEOUT_T_RX_TIMEOUT	Time that has to occur before the timeout
21	RW	TIMING_FAST_RX_EN_FAST_ RX_TXFILT	Filter Tx configuration for the fast Rx PLL
20	RW	TIMING_FAST_RX_EN_FAST_ RX	Fast Rx PLL
19:16	RW	TIMING_FAST_RX_T_RX_ FAST_CHP	Time to switch off the fast CHP in Rx mode
15:12	RW	TIMINGS_5_T_RX_RF	Time needed by the RF blocks to switch on in Rx mode
11:8	RW	TIMINGS_5_T_RX_BB	Time needed by the BB blocks to switch on in Rx mode
7:4	RW	TIMINGS_4_T_SUBBAND_RX	Time needed by the subband algorithm to calibrate in Rx mode
3:0	RW	TIMINGS_4_T_PLL_RX	Time needed by the PLL blocks to switch on in Rx mode

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31	TIMEOUT_EN_RX_TIMEOUT	TIMEOUT_EN_RX_TIMEOUT_DISABLE	Disable the timeout of the Rx	0x0*
		TIMEOUT_EN_RX_TIMEOUT_ENABLE	Enable the timeout of the Rx	0x1
30:28	TIMEOUT_T_TIMEOUT_GR	TIMEOUT_T_TIMEOUT_GR_DEFAULT		0x0*
27:24	TIMEOUT_T_RX_TIMEOUT	TIMEOUT_T_RX_TIMEOUT_DEFAULT		0x0*
21	TIMING_FAST_RX_EN_FAST_RX_TXFILT	TIMING_FAST_RX_EN_FAST_RX_TXFILT_DISABLE	Disable filter Tx configuration for the fast Rx PLL	0x0*
		TIMING_FAST_RX_EN_FAST_RX_TXFILT_ENABLE	Enable filter Tx configuration for the fast Rx PLL	0x1
20	TIMING_FAST_RX_EN_FAST_RX	TIMING_FAST_RX_EN_FAST_RX_DISABLE	Disable the fast Rx PLL	0x0*
		TIMING_FAST_RX_EN_FAST_RX_ENABLE	Enable the fast Rx PLL	0x1
19:16	TIMING_FAST_RX_T_RX_FAST_CHP	TIMING_FAST_RX_T_RX_FAST_CHP_DEFAULT		0x0*
15:12	TIMINGS_5_T_RX_RF	TIMINGS_5_T_RX_RF_DEFAULT		0x0*
11:8	TIMINGS_5_T_RX_BB	TIMINGS_5_T_RX_BB_DEFAULT		0x1*
7:4	TIMINGS_4_T_SUBBAND_RX	TIMINGS_4_T_SUBBAND_RX_DEFAULT		0x5*
3:0	TIMINGS_4_T_PLL_RX	TIMINGS_4_T_PLL_RX_DEFAULT		0x1*

5.9.0.39 RF_BIAS_0_2

Bit Field	Read/Write	Field Name	Description
31:28	RW	BIAS_2_IQ_RXTX_6	VCOM_MX bias
27:24	RW	BIAS_2_IQ_RXTX_5	VCOM_LO bias
23:20	RW	BIAS_1_IQ_RXTX_3	PrePA Casc bias
19:16	RW	BIAS_1_IQ_RXTX_2	PrePA In bias
15:12	RW	BIAS_0_IQ_RXTX_1	PA backoff bias
11:8	RW	BIAS_0_IQ_RXTX_0	PA bias

Bit Field	Read/Write	Field Name	Description
7	RW	INTERFACE_CONF_EN_SYNC_IFACE	Interfaces resynchronization
6:4	RW	INTERFACE_CONF_APB_WAIT_STATE	Select the number of wait states during the APB transaction
1:0	RW	INTERFACE_CONF_SPI_SELECT	Select the SPI mode

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:28	BIAS_2_IQ_RXTX_6	BIAS_2_IQ_RXTX_6_DEFAULT		0x3*
27:24	BIAS_2_IQ_RXTX_5	BIAS_2_IQ_RXTX_5_DEFAULT		0x8*
23:20	BIAS_1_IQ_RXTX_3	BIAS_1_IQ_RXTX_3_DEFAULT		0x6*
19:16	BIAS_1_IQ_RXTX_2	BIAS_1_IQ_RXTX_2_DEFAULT		0x6*
15:12	BIAS_0_IQ_RXTX_1	BIAS_0_IQ_RXTX_1_DEFAULT		0x7*
11:8	BIAS_0_IQ_RXTX_0	BIAS_0_IQ_RXTX_0_DEFAULT		0x3*
7	INTERFACE_CONF_EN_SYNC_IFACE	INTERFACE_CONF_EN_SYNC_IFACE_DISABLE	Interfaces are not resynchronized if the clock is available	0x0*
		INTERFACE_CONF_EN_SYNC_IFACE_ENABLE	Interfaces are resynchronized if the clock is available	0x1
6:4	INTERFACE_CONF_APB_WAIT_STATE	INTERFACE_CONF_APB_WAIT_STATE_DEFAULT		0x0*
1:0	INTERFACE_CONF_SPI_SELECT	INTERFACE_CONF_SPI_SELECT_LEGACY_SPI	Legacy SPI	0x0*
		INTERFACE_CONF_SPI_SELECT_ADVANCED_SPI	Advanced SPI	0x1
		INTERFACE_CONF_SPI_SELECT_BLIM4SME_SPI	BLIM4SME SPI	0x2

5.9.0.40 RF_BIAS_3_6

Bit Field	Read/Write	Field Name	Description
31:28	RW	BIAS_6_IQ_BB_0	ACD_O bias
27:24	RW	BIAS_6_IQ_PLL_3	DLL bias
23:20	RW	BIAS_5_IQ_PLL_4_RX	VCO bias for Rx mode
19:16	RW	BIAS_5_IQ_PLL_4_TX	VCO bias for Tx mode

Bit Field	Read/Write	Field Name	Description
15:12	RW	BIAS_4_IQ_PLL_2	Sub-band comparator bias
11:8	RW	BIAS_4_IQ_PLL_1	Dynamic divider bias
7:4	RW	BIAS_3_IQ_RXTX_8	IFA ctrl_c bias
3:0	RW	BIAS_3_IQ_RXTX_7	IFA ctrl_r bias

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:28	BIAS_6_IQ_BB_0	BIAS_6_IQ_BB_0_DEFAULT		0x7*
27:24	BIAS_6_IQ_PLL_3	BIAS_6_IQ_PLL_3_DEFAULT		0x7*
23:20	BIAS_5_IQ_PLL_4_RX	BIAS_5_IQ_PLL_4_RX_ DEFAULT		0x8*
19:16	BIAS_5_IQ_PLL_4_TX	BIAS_5_IQ_PLL_4_TX_ DEFAULT		0xA*
15:12	BIAS_4_IQ_PLL_2	BIAS_4_IQ_PLL_2_DEFAULT		0x7*
11:8	BIAS_4_IQ_PLL_1	BIAS_4_IQ_PLL_1_DEFAULT		0x4*
7:4	BIAS_3_IQ_RXTX_8	BIAS_3_IQ_RXTX_8_DEFAULT		0x7*
3:0	BIAS_3_IQ_RXTX_7	BIAS_3_IQ_RXTX_7_DEFAULT		0x7*

5.9.0.41 RF_BIAS_7_9

Bit Field	Read/Write	Field Name	Description
31:28	RW	BIAS_9_IQ_BB_6	Peak detector threshold bias 0
27:24	RW	BIAS_9_IQ_BB_5	Peak detector bias
23:20	RW	SWCAP_FSM_SB_CAP_RX	VCO subband selection (FSM in Rx mode)
19:16	RW	SWCAP_FSM_SB_CAP_TX	VCO subband selection (FSM in Tx mode)
15:12	RW	BIAS_8_IQ_BB_4	RSSI_D bias
11:8	RW	BIAS_8_IQ_BB_3	RSSI_G bias
7:4	RW	BIAS_7_IQ_BB_2	ACD_L bias
3:0	RW	BIAS_7_IQ_BB_1	ACD_C bias

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:28	BIAS_9_IQ_BB_6	BIAS_9_IQ_BB_6_DEFAULT		0x9*
27:24	BIAS_9_IQ_BB_5	BIAS_9_IQ_BB_5_DEFAULT		0x5*
23:20	SWCAP_FSM_SB_CAP_ RX	SWCAP_FSM_SB_CAP_RX_ DEFAULT		0x0*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
19:16	SWCAP_FSM_SB_CAP_TX	SWCAP_FSM_SB_CAP_TX_DEFAULT		0x0*
15:12	BIAS_8_IQ_BB_4	BIAS_8_IQ_BB_4_DEFAULT		0x9*
11:8	BIAS_8_IQ_BB_3	BIAS_8_IQ_BB_3_DEFAULT		0xF*
7:4	BIAS_7_IQ_BB_2	BIAS_7_IQ_BB_2_DEFAULT		0x6*
3:0	BIAS_7_IQ_BB_1	BIAS_7_IQ_BB_1_DEFAULT		0x6*

5.9.0.42 RF_BIAS_10_12

Bit Field	Read/Write	Field Name	Description
30	RW	SD_MASH_MASH_DITHER_TYPE	Enable the new dithering scheme
29	RW	SD_MASH_MASH_ENABLE	Enable the sigma delta mash
28	RW	SD_MASH_MASH_DITHER	Enable dithering on the sigma delta mash
27:25	RW	SD_MASH_MASH_ORDER	Order of the sigma delta mash
24	RW	SD_MASH_MASH_RSTB	Reset of the sigma delta mash (active low)
23:20	RW	BIAS_12_LNA_AGC_BIAS_3	LNA bias for AGC level 3
19:16	RW	BIAS_12_LNA_AGC_BIAS_2	LNA bias for AGC level 2
15:12	RW	BIAS_11_LNA_AGC_BIAS_1	LNA bias for AGC level 1
11:8	RW	BIAS_11_LNA_AGC_BIAS_0	LNA bias for AGC level 0
7:4	RW	BIAS_10_IQ_BB_8	Peak detector threshold bias 1
3:0	RW	BIAS_10_IQ_BB_7	Peak detector threshold bias 2

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
30	SD_MASH_MASH_DITHER_TYPE	SD_MASH_MASH_DITHER_TYPE_0	Disable the new dithering scheme	0x0*
		SD_MASH_MASH_DITHER_TYPE_ENABLE	Enable the new dithering scheme	0x1
29	SD_MASH_MASH_ENABLE	SD_MASH_MASH_ENABLE_DISABLE	Disable the sigma delta mash	0x0*
		SD_MASH_MASH_ENABLE_ENABLE	Enable the sigma delta mash	0x1
28	SD_MASH_MASH_DITHER	SD_MASH_MASH_DITHER_DISABLE	Disable dithering on the sigma delta mash	0x0
		SD_MASH_MASH_DITHER_ENABLE	Enable dithering on the sigma delta mash	0x1*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
27:25	SD_MASH_MASH_ ORDER	SD_MASH_MASH_ORDER_DEFAULT		0x3*
24	SD_MASH_MASH_RSTB	SD_MASH_MASH_RSTB_RESET	Reset the sigma delta mash	0x0
		SD_MASH_MASH_RSTB_NO_RESET	Do not reset the sigma delta mash	0x1*
23:20	BIAS_12_LNA_AGC_ BIAS_3	BIAS_12_LNA_AGC_BIAS_3_ DEFAULT		0x6*
19:16	BIAS_12_LNA_AGC_ BIAS_2	BIAS_12_LNA_AGC_BIAS_2_ DEFAULT		0x7*
15:12	BIAS_11_LNA_AGC_ BIAS_1	BIAS_11_LNA_AGC_BIAS_1_ DEFAULT		0x8*
11:8	BIAS_11_LNA_AGC_ BIAS_0	BIAS_11_LNA_AGC_BIAS_0_ DEFAULT		0x9*
7:4	BIAS_10_IQ_BB_8	BIAS_10_IQ_BB_8_DEFAULT		0x0*
3:0	BIAS_10_IQ_BB_7	BIAS_10_IQ_BB_7_DEFAULT		0x6*

5.9.0.43 RF_REG2A

Bit Field	Read/Write	Field Name	Description
27:24	RW	SD_MASH_MASK_MASH_MASK	Mask the n LSB of the fractional part of the MASH (debug only)
19	RW	BIAS_EN_2_EN_PTAT	Enable PTAT
18:16	RW	BIAS_EN_2_EN_BIAS_BB_HI	Bias enable for BB (same order as biases)
15:12	RW	BIAS_EN_1_EN_BIAS_BB_LO	Bias enable for BB (same order as biases)
11:7	RW	BIAS_EN_1_EN_BIAS_PLL	Bias enable for PLL (same order as biases)
6:0	RW	BIAS_EN_1_EN_BIAS_RXTX	Bias enable for RxTx (same order as biases)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
27:24	SD_MASH_MASK_ MASH_MASK	SD_MASH_MASK_MASH_MASK_ DEFAULT		0x0*
19	BIAS_EN_2_EN_PTAT	BIAS_EN_2_EN_PTAT_DISABLE	Disable PTAT	0x0
		BIAS_EN_2_EN_PTAT_ENABLE	Enable PTAT	0x1*
18:16	BIAS_EN_2_EN_BIAS_ BB_HI	BIAS_EN_2_EN_BIAS_BB_HI_ DEFAULT		0x0*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
15:12	BIAS_EN_1_EN_BIAS_BB_LO	BIAS_EN_1_EN_BIAS_BB_LO_DEFAULT		0x0*
11:7	BIAS_EN_1_EN_BIAS_PLL	BIAS_EN_1_EN_BIAS_PLL_DEFAULT		0x0*
6:0	BIAS_EN_1_EN_BIAS_RXTX	BIAS_EN_1_EN_BIAS_RXTX_DEFAULT		0x0*

5.9.0.44 RF_PLL_CTRL

Bit Field	Read/Write	Field Name	Description
26	RW	PLL_CTRL_DISABLE_CHP_SBS	Charge-pump disabling during sub-band selection (FLL and frequency ratios)
25	RW	PLL_CTRL_PLL_RX_48MEG	PLL frequency
24	RW	PLL_CTRL_SWCAP_TX_SAME_RX	Registers for Rx and Tx modes swcap in case of swcap_fsm=1
23	RW	PLL_CTRL_SWCAP_FSM	Selection of the swcap_fsm register
22	RW	PLL_CTRL_DLL_RSTB	Reset signal of the DLL (active low)
21:18	RW	PLL_CTRL_VCO_SUBBAND_TRIM	VCO sub-band selection bits
17	RW	PLL_CTRL_SUB_SEL_OFFS_EN	Add offset to sub-band selection comparator
16	RW	PLL_CTRL_DIV2_CLKVCO_TEST_EN	VCO signal divided by the programmable divider
15	RW	PLL_CTRL_VCODIV_CLK_TEST_EN	Output on GPIO the VCO signal divided by the programmable divider
13	RW	PLL_CTRL_CHP_DEAD_ZONE_EN	Charge-pump dead zone
12:11	RW	PLL_CTRL_CHP_CURR_OFF_TRIM_TX	Charge-pump offset current values selection bits in Tx mode
10:9	RW	PLL_CTRL_CHP_CURR_OFF_TRIM_RX	Charge-pump offset current values selection bits in Rx mode
8	RW	PLL_CTRL_HIGH_BW_FILTER_EN_TX	PLL filter high bandwidth needed in Tx mode
7	RW	PLL_CTRL_HIGH_BW_FILTER_EN_RX	PLL filter high bandwidth needed in Rx mode
6	RW	PLL_CTRL_FAST_CHP_EN_TX	High current output of the charge-pump for PLL Tx high bandwidth mode
5	RW	PLL_CTRL_FAST_CHP_EN_RX	High current output of the charge-pump for PLL Rx high bandwidth mode

Bit Field	Read/Write	Field Name	Description
4:3	RW	PLL_CTRL_CHP_MODE_TRIM	Select the frequency inside sub-band selection
2	RW	PLL_CTRL_CHP_CMC_EN	Common mode control block of the charge-pump
1	RW	PLL_CTRL_CHP_CURR_OFF_EN_TX	Charge-pump offset current in Tx mode
0	RW	PLL_CTRL_CHP_CURR_OFF_EN_RX	Charge-pump offset current in Rx mode

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
26	PLL_CTRL_DISABLE_CHP_SBS	PLL_CTRL_2_DISABLE_CHP_SBS_DISABLE	Charge-pump is enabled during the sub-band selection	0x0*
		PLL_CTRL_2_DISABLE_CHP_SBS_ENABLE	Charge-pump is disabled during the sub-band selection	0x1
25	PLL_CTRL_PLL_RX_48MEG	PLL_CTRL_2_PLL_RX_48MEG_24	PLL is set to 24MHz in Rx	0x0
		PLL_CTRL_2_PLL_RX_48MEG_48	PLL is set to 48MHz in Rx	0x1*
24	PLL_CTRL_SWCAP_TX_SAME_RX	PLL_CTRL_2_SWCAP_TX_SAME_RX_DISABLE	The register for Rx and Tx swcap is not the same	0x0*
		PLL_CTRL_2_SWCAP_TX_SAME_RX_ENABLE	The register for Rx and Tx swcap is the same	0x1
23	PLL_CTRL_SWCAP_FSM	PLL_CTRL_2_SWCAP_FSM_DISABLE	Don't use the swcap_fsm register as reference for the sub-band selection	0x0
		PLL_CTRL_2_SWCAP_FSM_ENABLE	Use the swcap_fsm register as reference for the sub-band selection	0x1*
22	PLL_CTRL_DLL_RSTB	PLL_CTRL_2_DLL_RSTB_RESET	Reset the DLL	0x0
		PLL_CTRL_2_DLL_RSTB_NO_RESET	Don't reset the DLL	0x1*
21:18	PLL_CTRL_VCO_SUBBAND_TRIM	PLL_CTRL_VCO_SUBBAND_TRIM_DEFAULT		0x0*
17	PLL_CTRL_SUB_SEL_OFFS_EN	PLL_CTRL_1_SUB_SEL_OFFS_EN_DISABLE	Don't add offset to sub-band selection comparator	0x0*
		PLL_CTRL_1_SUB_SEL_OFFS_EN_ENABLE	Add offset to sub-band selection comparator	0x1
16	PLL_CTRL_DIV2_CLKVCO_TEST_EN	PLL_CTRL_1_DIV2_CLKVCO_TEST_EN_1	Division ratio set to 1	0x0*
		PLL_CTRL_1_DIV2_CLKVCO_TEST_EN_2	Division ratio set to 2 (before to be outputted to ck_div_test)	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
15	PLL_CTRL_VCODIV_CLK_TEST_EN	PLL_CTRL_1_VCODIV_CLK_TEST_EN_DISABLE	Disable to output on GPIO the VCO signal divided by the programmable divider	0x0*
		PLL_CTRL_ENABLE_VCODIV_CLK_TEST_EN_ENABLE	Enable to output on GPIO the VCO signal divided by the programmable divider	0x1
13	PLL_CTRL_CHP_DEAD_ZONE_EN	PLL_CTRL_1_CHP_DEAD_ZONE_EN_DISABLE	Disable charge-pump dead zone	0x0*
		PLL_CTRL_ENABLE_CHP_DEAD_ZONE_EN_ENABLE	Enable charge-pump dead zone	0x1
12:11	PLL_CTRL_CHP_CURR_OFF_TRIM_TX	PLL_CTRL_1_CHP_CURR_OFF_TRIM_TX_15	d_phi=15	0x0
		PLL_CTRL_1_CHP_CURR_OFF_TRIM_TX_22	d_phi=22.5	0x1
		PLL_CTRL_1_CHP_CURR_OFF_TRIM_TX_30	d_phi = 30	0x2
		PLL_CTRL_1_CHP_CURR_OFF_TRIM_TX_60	d_phi = 60	0x3*
10:9	PLL_CTRL_CHP_CURR_OFF_TRIM_RX	PLL_CTRL_1_CHP_CURR_OFF_TRIM_RX_15	d_phi=15	0x0
		PLL_CTRL_1_CHP_CURR_OFF_TRIM_RX_22	d_phi=22.5	0x1
		PLL_CTRL_1_CHP_CURR_OFF_TRIM_RX_30	d_phi = 30	0x2
		PLL_CTRL_1_CHP_CURR_OFF_TRIM_RX_60	d_phi = 60	0x3*
8	PLL_CTRL_HIGH_BW_FILTER_EN_TX	PLL_CTRL_HIGH_BW_FILTER_EN_TX_DISABLE	Disable the PLL filter high bandwidth needed in TX	0x0
		PLL_CTRL_HIGH_BW_FILTER_EN_TX_ENABLE	Enable the PLL filter high bandwidth needed in TX	0x1*
7	PLL_CTRL_HIGH_BW_FILTER_EN_RX	PLL_CTRL_HIGH_BW_FILTER_EN_RX_DISABLE	Disable the PLL filter high bandwidth needed in RX	0x0*
		PLL_CTRL_HIGH_BW_FILTER_EN_RX_ENABLE	Enable the PLL filter high bandwidth needed in RX	0x1
6	PLL_CTRL_FAST_CHP_EN_TX	PLL_CTRL_FAST_CHP_EN_TX_DISABLE	Disable the high current output of the charge-pump in TX	0x0
		PLL_CTRL_FAST_CHP_EN_TX_	Enable the high current output of the	0x1*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		ENABLE	charge-pum in TX	
5	PLL_CTRL_FAST_CHP_EN_RX	PLL_CTRL_FAST_CHP_EN_RX_DISABLE	Disable the high current output of the charge-pump in RX	0x0*
		PLL_CTRL_FAST_CHP_EN_RX_ENABLE	Enable the high current output of the charge-pump in RX	0x1
4:3	PLL_CTRL_CHP_MODE_TRIM	PLL_CTRL_CHP_MODE_TRIM_MIN_FREQ	Minimum frequency inside sub-band selection	0x0*
		PLL_CTRL_CHP_MODE_TRIM_MED_FREQ	Medium frequency inside sub-band selection	0x1
		PLL_CTRL_CHP_MODE_TRIM_MAX_FREQ	Maximum frequency inside sub-band selection	0x2
2	PLL_CTRL_CHP_CMC_EN	PLL_CTRL_CHP_CMC_EN_DISABLE	Disable the common mode control block of the charge-pump	0x0
		PLL_CTRL_CHP_CMC_EN_ENABLE	Enable the common mode control block of the charge-pump	0x1*
1	PLL_CTRL_CHP_CURR_OFF_EN_TX	PLL_CTRL_CHP_CURR_OFF_EN_TX_DISABLE	Disable the charge-pump offset current in TX	0x0
		PLL_CTRL_CHP_CURR_OFF_EN_TX_ENABLE	Enable the charge-pump offset current in TX	0x1*
0	PLL_CTRL_CHP_CURR_OFF_EN_RX	PLL_CTRL_CHP_CURR_OFF_EN_RX_DISABLE	Disable the charge-pump offset current in RX	0x0*
		PLL_CTRL_CHP_CURR_OFF_EN_RX_ENABLE	Enable the charge-pump offset current in RX	0x1

5.9.0.45 RF_DLL_CTRL

Bit Field	Read/Write	Field Name	Description
31:29	RW	RSSI_TUN_1_RSSI_TUN_GAIN	RSSI tuning for gain
28:24	RW	RSSI_TUN_1_RSSI_ODD_OFFSET	RSSI tuning for odd stages (offset to the even triangular wave)
23:20	RW	RSSI_TUN_1_RSSI_EVEN_MAX	RSSI tuning for even stages (maximum value of the triangular wave)
19:16	RW	RSSI_TUN_1_RSSI_EVEN_MIN	RSSI tuning for even stages (minimum value of the triangular wave)
12	RW	DLL_CTRL_CK_LAST_SEL_DELAY	Last SEL delay
11	RW	DLL_CTRL_CK_FIRST_SEL_DELAY	First SEL delay

Bit Field	Read/Write	Field Name	Description
10	RW	DLL_CTRL_CK_EXT_SEL	Input clock selection
9	RW	DLL_CTRL_CK_DIG_EN	Alternate ck_dig pin to output the PLL reference clock signal
8	RW	DLL_CTRL_CK_TEST_EN	Output on GPIO the PLL reference clock signal via ck_test pin
7	RW	DLL_CTRL_TOO_FAST_ENB	Lock range phase detector
6	RW	DLL_CTRL_LOCKED_DET_EN	Reference frequency multiplier locked detector
5	RW	DLL_CTRL_LOCKED_AUTO_CHECK_EN	Frequency multiplier is out of lock (usually because some input clocks from ck_xtal or ck_ext are missing)
4	RW	DLL_CTRL_FAST_ENB	Disable fast mode locking of the reference frequency multiplier
3:2	RW	DLL_CTRL_CK_SEL_TX	Selection of the clock used as frequency reference of the PLL in Tx mode (also to ck_test and ck_dig outputs)
1:0	RW	DLL_CTRL_CK_SEL_RX	Selection of the clock used as frequency reference of the PLL in Rx mode (also to ck_test and ck_dig outputs)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:29	RSSI_TUN_1_RSSI_TUN_GAIN	RSSI_TUN_1_RSSI_TUN_GAIN_DEFAULT		0x1*
28:24	RSSI_TUN_1_RSSI_ODD_OFFSET	RSSI_TUN_1_RSSI_ODD_OFFSET_DEFAULT		0x4*
23:20	RSSI_TUN_1_RSSI_EVEN_MAX	RSSI_TUN_1_RSSI_EVEN_MAX_DEFAULT		0x1*
19:16	RSSI_TUN_1_RSSI_EVEN_MIN	RSSI_TUN_1_RSSI_EVEN_MIN_DEFAULT		0x1*
12	DLL_CTRL_CK_LAST_SEL_DELAY	DLL_CTRL_CK_LAST_SEL_DELAY_0		0x0*
		DLL_CTRL_CK_LAST_SEL_DELAY_1		0x1
11	DLL_CTRL_CK_FIRST_SEL_DELAY	DLL_CTRL_CK_FIRST_SEL_DELAY_0		0x0*
		DLL_CTRL_CK_FIRST_SEL_DELAY_1		0x1
10	DLL_CTRL_CK_EXT_SEL	DLL_CTRL_CK_EXT_SEL_XTAL	Input clock comes from ck_xtal pin	0x0*
		DLL_CTRL_CK_EXT_SEL_EXT	Input clock comes from ck_ext pin	0x1
9	DLL_CTRL_CK_DIG_EN	DLL_CTRL_CK_DIG_EN_NOMINAL	Don't use the alternate ck_dig pin to output the PLL reference clock signal	0x0*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		DLL_CTRL_CK_DIG_EN_ALTERNATE	Use the alternate ck_dig pin to output the PLL reference clock signal	0x1
8	DLL_CTRL_CK_TEST_EN	DLL_CTRL_CK_TEST_EN_0	Don't output on GPIO the PLL reference clock signal via ck_test pin	0x0*
		DLL_CTRL_CK_TEST_EN_ENABLE	Output on GPIO the PLL reference clock signal via ck_test pin	0x1
7	DLL_CTRL_TOO_FAST_ENB	DLL_CTRL_TOO_FAST_ENB_DISABLE	Enable auxiliary wide lock range phase detector when fast mode locking is enabled (fast_enb = 0)	0x0*
		DLL_CTRL_TOO_FAST_ENB_ENABLE	The narrow lock range phase detector is enabled and bit 2 (fast_enb) must be high to avoid false frequency lock (slow mode locking)	0x1
6	DLL_CTRL_LOCKED_DET_EN	DLL_CTRL_LOCKED_DET_EN_DISABLE	Disable reference frequency multiplier locked detector	0x0
		DLL_CTRL_LOCKED_DET_EN_ENABLE	Enable reference frequency multiplier locked detector	0x1*
5	DLL_CTRL_LOCKED_AUTO_CHECK_EN	DLL_CTRL_LOCKED_AUTO_CHECK_EN_DISABLE	Manual reset should be performed via dll_rstb input(see Table 3) to relock the frequency multiplier	0x0
		DLL_CTRL_LOCKED_AUTO_CHECK_EN_ENABLE	Frequency multiplier will try to lock again automatically	0x1*
4	DLL_CTRL_FAST_ENB	DLL_CTRL_FAST_ENB_ENABLE	Fast mode locking of the reference frequency multiplier	0x0*
		DLL_CTRL_FAST_ENB_DISABLE		0x1
3:2	DLL_CTRL_CK_SEL_TX	DLL_CTRL_CK_SEL_TX_0	ref = ck_xtal ot ck_ext (if bit 8 is high)	0x0
		DLL_CTRL_CK_SEL_TX_1	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)	0x1*
		DLL_CTRL_CK_SEL_TX_2	ref = same as ck_sel = 01 but output frequency divided by 2 (used in normal RX mode when dll_en = 0)	0x2
		DLL_CTRL_CK_SEL_TX_3	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)	0x3
1:0	DLL_CTRL_CK_SEL_RX	DLL_CTRL_CK_SEL_RX_0	ref = ck_xtal ot ck_ext (if bit 8 is high)	0x0*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		DLL_CTRL_CK_SEL_RX_1	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)	0x1
		DLL_CTRL_CK_SEL_RX_2	ref = same as ck_sel = 01 but output frequency divided by 2 (used in normal RX mode when dll_en = 0)	0x2
		DLL_CTRL_CK_SEL_RX_3	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)	0x3

5.9.0.46 RF_REG2D

Bit Field	Read/Write	Field Name	Description
29:28	RW	PA_CONF_SW_CN	Harmonic 2 notch tuning
27	RW	PA_CONF_TX_SWITCHPA	Switch PA
26	RW	PA_CONF_TX_0DBM	Select between PPA and PA
25	RW	PA_CONF_LIN_RAMP	PA ramp-up linearization
24	RW	PA_CONF_MIN_PA_PWR	Set the minimum power during the PA ramp-up
23	RW	CTRL_RX_SWITCH_LP	Switch the low-pass filter in the Rx chain
22	RW	CTRL_RX_USE_PEAK_ DETECTOR	Peak detector powering
21	RW	CTRL_RX_START_MIX_ON_CAL	Mixer enabling
20:16	RW	CTRL_RX_CTRL_RX	Rx control
15	RW	CTRL_ADC_PHADC_THERM_ OUT_EN	Enable the buffers of phase ADC thermometric code (banked)
14:13	RW	CTRL_ADC_PHADC_DELLATCH	Phase ADC delay latch trimming (banked)
12:8	RW	CTRL_ADC_CTRL_ADC	Phase ADC control (banked)
6:5	RW	RSSI_TUN_2_RSSI_TRI_CK_ DIV	Speed on the RSSI triangular dithering signal (cf reg RSSI_TUN)
4	RW	RSSI_TUN_2_RSSI_ONE_CK_ RSSI_PHADC	RSSI and phase ADC clocks sharing
3	RW	RSSI_TUN_2_RSSI_FULL	RSSI full scale
2	RW	RSSI_TUN_2_RSSI_1DB	LSB resolution
1:0	RW	RSSI_TUN_2_RSSI_PRE_ATT	Pre attenuation of the RSSI signal

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
29:28	PA_CONF_SW_CN	PA_CONF_SW_CN_DEFAULT		0x0*
27	PA_CONF_TX_SWITCHPA	PA_CONF_TX_SWITCHPA_DISABLE	RF Tx timing	0x0*
		PA_CONF_TX_SWITCHPA_ENABLE	Enable the PA only with the digital block	0x1
26	PA_CONF_TX_0DBM	PA_CONF_TX_DISABLEDBM_DISABLE	Only use the PPA is used (-20dBm)	0x0
		PA_CONF_TX_DISABLEDBM_ENABLE	Enable the PA	0x1*
25	PA_CONF_LIN_RAMP	PA_CONF_LIN_RAMP_DISABLE	Don't linearize the PA ramp-up by having not equal ramp-up step delays	0x0*
		PA_CONF_LIN_RAMP_ENABLE	Linearize the PA ramp-up by having not equal ramp-up step delays	0x1
24	PA_CONF_MIN_PA_PWR	PA_CONF_MIN_PA_PWR_M3	The ramp-up starts at -3	0x0
		PA_CONF_MIN_PA_PWR_M1	The ramp-up starts at -1	0x1*
23	CTRL_RX_SWITCH_LP	CTRL_RX_SWITCH_LP_DISABLE	Do not switch the low-pass filter in the Rx chain	0x0*
		CTRL_RX_SWITCH_LP_ENABLE	Switch the low-pass filter in the Rx chain	0x1
22	CTRL_RX_USE_PEAK_DETECTOR	CTRL_RX_USE_PEAK_DETECTOR_DISABLE	The peak detector is not powered on during the Rx by the FSM	0x0
		CTRL_RX_USE_PEAK_DETECTOR_ENABLE	The peak detector is powered on during the Rx by the FSM	0x1*
21	CTRL_RX_START_MIX_ON_CAL	CTRL_RX_START_MIX_ON_CAL_DISABLE	The mixer is disabled during the sub-band selection phase	0x0*
		CTRL_RX_START_MIX_ON_CAL_ENABLE	The mixer is enabled during the sub-band selection phase	0x1
20:16	CTRL_RX_CTRL_RX	CTRL_RX_CTRL_RX_DEFAULT	Rx control	0xF*
15	CTRL_ADC_PHADC_THERM_OUT_EN	CTRL_ADC_PHADC_THERM_OUT_EN_DISABLE	Disable thermometric code	0x0
		CTRL_ADC_PHADC_THERM_OUT_EN_ENABLE	Enable thermometric code	0x1*
14:13	CTRL_ADC_PHADC_DELLATCH	CTRL_ADC_PHADC_DELLATCH_DEFAULT		0x1*
12:8	CTRL_ADC_CTRL_ADC	CTRL_ADC_CTRL_ADC_DEFAULT	bits(1:0) => phase ADC reset delay, bits (3:2) phase ADC clock delay, bit(4) phase ADC latch idle	0x5*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
6:5	RSSI_TUN_2_RSSI_TRI_CK_DIV	RSSI_TUN_2_RSSI_TRI_CK_DIV_2	2 RSSI clk periods	0x0*
		RSSI_TUN_2_RSSI_TRI_CK_DIV_4	4 RSSI clk periods	0x1
		RSSI_TUN_2_RSSI_TRI_CK_DIV_8	8 RSSI clk periods	0x2
		RSSI_TUN_2_RSSI_TRI_CK_DIV_16	16 RSSI clk periods	0x3
4	RSSI_TUN_2_RSSI_ONE_CK_RSSI_PHADC	RSSI_TUN_2_RSSI_ONE_CK_RSSI_PHADC_DISABLE	RSSI and the phase ADC don't share the same clock	0x0*
		RSSI_TUN_2_RSSI_ONE_CK_RSSI_PHADC_ENABLE	RSSI and the phase ADC share the same clock	0x1
3	RSSI_TUN_2_RSSI_FULL	RSSI_TUN_2_RSSI_FULL_DISABLE	RSSI full scale is not used	0x0
		RSSI_TUN_2_RSSI_FULL_ENABLE	RSSI full scale is used (10bits)	0x1*
2	RSSI_TUN_2_RSSI_1DB	RSSI_TUN_2_RSSI_1DB_0P5	LSB is 0.5dB	0x0*
		RSSI_TUN_2_RSSI_1DB_1	LSB is 1dB	0x1
1:0	RSSI_TUN_2_RSSI_PRE_ATT	RSSI_TUN_2_RSSI_PRE_ATT_DEFAULT		0x3*

5.9.0.47 RF_REG2E

Bit Field	Read/Write	Field Name	Description
31:24	RW	XTAL_TRIM_XTAL_TRIM_INIT	Initial trimming of the XTAL
23:16	RW	XTAL_TRIM_XTAL_TRIM	Trimming of the XTAL
12	RW	ENABLES_SEPARATE_PPA_CASC	PA cascode bit
11:6	RW	ENABLES_EN_RXTX	Enable signals
5:0	RW	ENABLES_EN_BB	Enable signals for the BB

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:24	XTAL_TRIM_XTAL_TRIM_INIT	XTAL_TRIM_XTAL_TRIM_INIT_DEFAULT	5MSB thermometric, 3LSB direct	0x60*
23:16	XTAL_TRIM_XTAL_TRIM	XTAL_TRIM_XTAL_TRIM_DEFAULT	5MSB thermometric, 3LSB direct	0x60*
12	ENABLES_SEPARATE_PPA_CASC	ENABLES_SEPARATE_PPA_CASC_DISABLE	Enable PPA cascode bit is dependent of the en PA	0x0*
		ENABLES_SEPARATE_PPA_CASC_ENABLE	Disable PPA cascode bit is independent from the en PA	0x1
11:6	ENABLES_EN_RXTX	ENABLES_EN_RXTX_0	LNA	0x0*
		ENABLES_EN_RXTX_1	LNA	0x1
		ENABLES_EN_RXTX_2	IFA	0x2
		ENABLES_EN_RXTX_3	TX	0x3
		ENABLES_EN_RXTX_4	PA	0x4
		ENABLES_EN_RXTX_5	PPA casc	0x5
5:0	ENABLES_EN_BB	ENABLES_EN_BB_0	Filter,	0x0*
		ENABLES_EN_BB_1	Filter central frequency bias	0x1
		ENABLES_EN_BB_2	Filter bandwidth bias	0x2
		ENABLES_EN_BB_3	ADC	0x3
		ENABLES_EN_BB_4	RSSI	0x4
		ENABLES_EN_BB_5	Peak detector	0x5

5.9.0.48 RF_XTAL_CTRL

Bit Field	Read/Write	Field Name	Description
31:28	RW	XTAL_CTRL_XO_THR_HIGH	High threshold for XTAL trimming
27:24	RW	XTAL_CTRL_XO_THR_LOW	Low threshold for XTAL trimming
23:22	RW	XTAL_CTRL_XO_A_S_CURR_SEL_HIGH	Value of after_startup_curr_sel when level is higher than xo_thr_high
21:20	RW	XTAL_CTRL_XO_A_S_CURR_SEL_LOW	Value of after_startup_curr_sel when level is lower than xo_thr_low
19	RW	XTAL_CTRL_LOW_CLK_READY_TH_EN	clk_ready threshold
18	RW	XTAL_CTRL_XTAL_CTRL_BYPASS	Bypass the XTAL control algorithm
17	RW	XTAL_CTRL_DIG_CLK_IN_SEL	Clock selection for the digital block

Bit Field	Read/Write	Field Name	Description
16	RW	XTAL_CTRL_XO_EN_B_REG	XTAL oscillator enable
15:14	RW	XTAL_CTRL_XTAL_CKDIV	XTAL trimming speed
13	RW	XTAL_CTRL_CLK_OUT_EN_B	Output clock to go to main IP
12	RW	XTAL_CTRL_REG_VALUE_SEL	Control bits of xtal_reg
11:10	RW	XTAL_CTRL_AFTERSTARTUP_CURR_SEL	Selection of the current before amplitude stabilization but after starting-up in active transistors of the core oscillator
9:8	RW	XTAL_CTRL_STARTUP_CURR_SEL	Selection of the starting-up current in active transistors of the core oscillator
7	RW	XTAL_CTRL_INV_CLK_DIG	Invert clock on clk_dig output
6	RW	XTAL_CTRL_INV_CLK_PLL	Invert clock on clk_pll output
5	RW	XTAL_CTRL_FORCE_CLK_READY	Force output clocks on clk_pll, clk_dig and clk_out
4	RW	XTAL_CTRL_CLK_DIG_EN_B	Disable the output clock to go to digital (clk_dig output stay low)
3	RW	XTAL_CTRL_BUFF_EN_B	XTAL buffer disabling
2	RW	XTAL_CTRL_HP_MODE	Bias current increase in the clock buffer
1	RW	XTAL_CTRL_LP_MODE	Bias current decrease in the clock buffer
0	RW	XTAL_CTRL_EXT_CLK_MODE	Use XTAL pads as external clock input

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:28	XTAL_CTRL_XO_THR_HIGH	XTAL_CTRL_XO_THR_HIGH_DEFAULT		0xC*
27:24	XTAL_CTRL_XO_THR_LOW	XTAL_CTRL_XO_THR_LOW_DEFAULT		0x3*
23:22	XTAL_CTRL_XO_A_S_CURR_SEL_HIGH	XTAL_CTRL_XO_A_S_CURR_SEL_HIGH_DEFAULT		0x2*
21:20	XTAL_CTRL_XO_A_S_CURR_SEL_LOW	XTAL_CTRL_XO_A_S_CURR_SEL_LOW_DEFAULT		0x0*
19	XTAL_CTRL_LOW_CLK_READY_TH_EN	XTAL_CTRL_LOW_CLK_READY_TH_EN_NOMINAL	The clk_ready threshold is nominal	0x0*
		XTAL_CTRL_LOW_CLK_READY_TH_EN_LOW	The clk_ready threshold is set to a lower value	0x1
18	XTAL_CTRL_XTAL_CTRL_BYPASS	XTAL_CTRL_XTAL_CTRL_BYPASS_DISABLE	Don't bypass the Xtal control algorithm	0x0*
		XTAL_CTRL_XTAL_CTRL_	Bypass the Xtal control algorithm	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		BYPASS_ENABLE		
17	XTAL_CTRL_DIG_CLK_IN_SEL	XTAL_CTRL_DIG_CLK_IN_SEL_XTAL	Select the internal xtal	0x0*
		XTAL_CTRL_DIG_CLK_IN_SEL_CLK_IN	Select the clk_in_dig signal	0x1
16	XTAL_CTRL_XO_EN_B_REG	XTAL_CTRL_ENABLE_OSCILLATOR	Xtal oscillator enable	0x0
		XTAL_CTRL_DISABLE_OSCILLATOR	Xtal oscillator disable	0x1*
15:14	XTAL_CTRL_XTAL_CKDIV	XTAL_CTRL_XTAL_CKDIV_0	43 us	0x0*
		XTAL_CTRL_XTAL_CKDIV_1	85 us	0x1
		XTAL_CTRL_XTAL_CKDIV_2	171 us	0x2
		XTAL_CTRL_XTAL_CKDIV_3	341 us	0x3
13	XTAL_CTRL_CLK_OUT_EN_B	XTAL_CTRL_CLK_OUT_EN_B_ENABLE	Enable the output clock to go to main IP	0x0*
		XTAL_CTRL_CLK_OUT_EN_B_DISABLE	Disable the output clock to go to main IP (clk_out output stay low)	0x1
12	XTAL_CTRL_REG_VALUE_SEL	XTAL_CTRL_REG_VALUE_SEL_EXTERNAL	Main ctrl signals are used instead of corresponding ctrl signal or some control bits of xtal_reg	0x0*
		XTAL_CTRL_REG_VALUE_SEL_INTERNAL	Corresponding ctrl signal and some control bits of xtal_reg are used instead of main ctrl signals.	0x1
11:10	XTAL_CTRL_AFTERSTARTUP_CURR_SEL	XTAL_CTRL_AFTERSTARTUP_CURR_SEL_0	0.15 mA	0x0
		XTAL_CTRL_AFTERSTARTUP_CURR_SEL_1	0.24 mA	0x1*
		XTAL_CTRL_AFTERSTARTUP_CURR_SEL_2	0.40 mA	0x2
		XTAL_CTRL_AFTERSTARTUP_CURR_SEL_3	0.61 mA	0x3
9:8	XTAL_CTRL_STARTUP_CURR_SEL	XTAL_CTRL_STARTUP_CURR_SEL_0	0.41 mA	0x0
		XTAL_CTRL_STARTUP_CURR_SEL_1	0.59 mA	0x1*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		XTAL_CTRL_STARTUP_CURR_SEL_2	0.88 mA	0x2
		XTAL_CTRL_STARTUP_CURR_SEL_3	1.24 mA	0x3
7	XTAL_CTRL_INV_CLK_DIG	XTAL_CTRL_INV_CLK_DIG_DISABLE	Don't invert clock on clk_dig output	0x0*
		XTAL_CTRL_INV_CLK_DIG_ENABLE	Invert clock on clk_dig output	0x1
6	XTAL_CTRL_INV_CLK_PLL	XTAL_CTRL_INV_CLK_PLL_DISABLE	Don't invert clock on clk_pll output	0x0*
		XTAL_CTRL_INV_CLK_PLL_ENABLE	Invert clock on clk_pll output	0x1
5	XTAL_CTRL_FORCE_CLK_READY	XTAL_CTRL_FORCE_CLK_READY_DISABLE	Don't force output clocks on clk_pll, clk_dig and clk_out	0x0*
		XTAL_CTRL_FORCE_CLK_READY_ENABLE	Force output clocks on clk_pll, clk_dig and clk_out and bypass the xtal internal clock detector	0x1
4	XTAL_CTRL_CLK_DIG_EN_B	XTAL_CTRL_CLK_DIG_EN_B_DISABLE	Enable the output clock to go to digital	0x0*
		XTAL_CTRL_CLK_DIG_EN_B_ENABLE	Disable the output clock to go to digital (clk_dig output stay low)	0x1
3	XTAL_CTRL_BUFF_EN_B	XTAL_CTRL_BUFF_EN_B_DISABLE	The xtal buffer is enabled	0x0*
		XTAL_CTRL_BUFF_EN_B_ENABLE	The xtal buffer is disabled	0x1
2	XTAL_CTRL_HP_MODE	XTAL_CTRL_HP_MODE_NOMINAL	The bias current in the clock buffer is nominal	0x0*
		XTAL_CTRL_HP_MODE_HIGH	The bias current in the clock buffer is increased	0x1
1	XTAL_CTRL_LP_MODE	XTAL_CTRL_LP_MODE_NOMINAL	The bias current in the clock buffer is nominal	0x0*
		XTAL_CTRL_LP_MODE_HIGH	The bias current in the clock buffer is reduced compared to normal operatio	0x1
0	XTAL_CTRL_EXT_CLK_MODE	XTAL_CTRL_EXT_CLK_MODE_DISABLE	Don not use xtal_p (and eventually xtal_n) as external clock input(s)	0x0*
		XTAL_CTRL_EXT_CLK_MODE_ENABLE	Use xtal_p (and eventually xtal_n) as external clock input(s)	0x1

5.9.0.49 RF_SUBBAND

Bit Field	Read/Write	Field Name	Description
31:24	RW	SUBBAND_OFFSET_SB_OFFSET_RX	Offset to add in frequency count in order to compensate the offset of the varicap
23:16	RW	SUBBAND_OFFSET_SB_OFFSET	Offset to add in frequency count in order to compensate the offset of the varicap
15:12	RW	SWCAP_LIM_SB_MAX_VAL	Maximum subband value in linear search subband (freq and comp)
11:8	RW	SWCAP_LIM_SB_MIN_VAL	Minimum subband value in linear search subband (freq and comp)
7	RW	SUBBAND_CONF_SB_FLL_MODE	FLL mode for the subband selection
6	RW	SUBBAND_CONF_SB_INV_BAND	Invert the meaning of sb_high and sb_low
5:4	RW	SUBBAND_CONF_SB_FREQ_CNT	The length to count in frequency mode
3:2	RW	SUBBAND_CONF_SB_WAIT_T	Time to wait to the PLL to settle
1:0	RW	SUBBAND_CONF_SB_MODE	Sub-band algorithm mode

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:24	SUBBAND_OFFSET_SB_OFFSET_RX	SUBBAND_OFFSET_SB_OFFSET_RX_DEFAULT		0xF1*
23:16	SUBBAND_OFFSET_SB_OFFSET	SUBBAND_OFFSET_SB_OFFSET_DEFAULT		0xD0*
15:12	SWCAP_LIM_SB_MAX_VAL	SWCAP_LIM_SB_MAX_VAL_DEFAULT		0xF*
11:8	SWCAP_LIM_SB_MIN_VAL	SWCAP_LIM_SB_MIN_VAL_DEFAULT		0x0*
7	SUBBAND_CONF_SB_FLL_MODE	SUBBAND_CONF_SB_FLL_MODE_DISABLE	Disable the FLL mode for the subband selection	0x0
		SUBBAND_CONF_SB_FLL_MODE_ENABLE	Enable the FLL mode for the subband selection (overrides other settings)	0x1*
6	SUBBAND_CONF_SB_INV_BAND	SUBBAND_CONF_SB_INV_BAND_DISABLE	Don't invert the meaning of sb_high and sb_low	0x0*
		SUBBAND_CONF_SB_INV_BAND_ENABLE	Invert the meaning of sb_high and sb_low	0x1
5:4	SUBBAND_CONF_SB_FREQ_CNT	SUBBAND_CONF_SB_FREQ_CNT_256	256 (Rx: 10.7us, Tx: 2.13us)	0x0*
		SUBBAND_CONF_SB_FREQ_CNT_512	512 (Rx: 21.3us, Tx: 4.26us)	0x1
		SUBBAND_CONF_SB_FREQ_CNT_	1024 (Rx: 42.7us, Tx: 8.53us)	0x2

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		1024		
		SUBBAND_CONF_SB_FREQ_CNT_4096	4096 (Rx: 171us, Tx: 34.1us)	0x3
3:2	SUBBAND_CONF_SB_WAIT_T	SUBBAND_CONF_SB_WAIT_T_8	Rx 8us, Tx 2us	0x0*
		SUBBAND_CONF_SB_WAIT_T_12	Rx 12us, Tx 3us	0x1
		SUBBAND_CONF_SB_WAIT_T_16	Rx 16us, Tx 4us	0x2
		SUBBAND_CONF_SB_WAIT_T_24	Rx 24us, Tx 6u	0x3
1:0	SUBBAND_CONF_SB_MODE	SUBBAND_CONF_SB_MODE_0	SAR w/ comparators	0x0*
		SUBBAND_CONF_SB_MODE_1	linear w/ comparators	0x1
		SUBBAND_CONF_SB_MODE_2	SAR w/ frequency ratios	0x2
		SUBBAND_CONF_SB_MODE_3	linear w/ frequency ratios	0x3

5.9.0.50 RF_REG31

Bit Field	Read/Write	Field Name	Description
31:30	RW	RSSI_DETECT_RSSI_DET_CR_LEN	Number of samples to estimate the carrier offset (banked)
29:28	RW	RSSI_DETECT_RSSI_DET_WAIT	Symbols to wait after the RSSI detection (banked)
27:26	RW	RSSI_DETECT_RSSI_DET_DIFF_LL	Set the distance between the actual value and the subtracted one (banked)
25	RW	RSSI_DETECT_EN_ABS_RSSI_DETECT	Absolute RSSI detection (banked)
24	RW	RSSI_DETECT_EN_DIFF_RSSI_DETECT	Differential RSSI detection (banked)
23	RW	SUBBAND_CORR_SUBBAND_CORR_EN	Subband correction
22:20	RW	SUBBAND_CORR_SUBBAND_CORR_RX	Subband correction in Rx
18:16	RW	SUBBAND_CORR_SUBBAND_CORR_TX	Subband correction in Tx
11	RW	TXRX_CONF_INV_CLK_PLL_TX	Invert PLL clock when the radio is in Tx mode
10	RW	TXRX_CONF_INV_CLK_DIG_TX	Invert digital clock when the radio is in Tx mode

Bit Field	Read/Write	Field Name	Description
9:8	RW	TXRX_CONF_SB_WAIT_T_TX	Xor value to apply to sb_wait_t (register SUBBAND_CONF) when the radio is in Tx mode
7	RW	PA_RAMPUP_FULL_PA_RAMPUP	PA rampup configuration
6:4	RW	PA_RAMPUP_DEL_PA_RAMPUP	Time to wait to start the ramp-up after the PA enable is detected
3:2	RW	PA_RAMPUP_TAU_PA_RAMPUP	Time constant of the ramp-up/ramp-down
1	RW	PA_RAMPUP_EN_PA_RAMPDOWN	PA ramp-down
0	RW	PA_RAMPUP_EN_PA_RAMPUP	PA ramp-up linearization

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:30	RSSI_DETECT_RSSI_DET_CR_LEN	RSSI_DETECT_RSSI_DET_CR_LEN_32	32 samples	0x0*
		RSSI_DETECT_RSSI_DET_CR_LEN_64	64 samples	0x1
		RSSI_DETECT_RSSI_DET_CR_LEN_128	128 samples	0x2
		RSSI_DETECT_RSSI_DET_CR_LEN_256	256 samples	0x3
29:28	RSSI_DETECT_RSSI_DET_WAIT	RSSI_DETECT_RSSI_DET_WAIT_0	0 symbol	0x0*
		RSSI_DETECT_RSSI_DET_WAIT_1	1 symbol	0x1
		RSSI_DETECT_RSSI_DET_WAIT_2	2 symbols	0x2
		RSSI_DETECT_RSSI_DET_WAIT_4	4 symbols	0x3
27:26	RSSI_DETECT_RSSI_DET_DIFF_LL	RSSI_DETECT_RSSI_DET_DIFF_LL_1	1 sample	0x0*
		RSSI_DETECT_RSSI_DET_DIFF_LL_2	2 samples	0x1
		RSSI_DETECT_RSSI_DET_DIFF_LL_3	3 samples	0x2
		RSSI_DETECT_RSSI_DET_DIFF_LL_4	4 samples	0x3
25	RSSI_DETECT_EN_ABS_RSSI_DETECT	RSSI_DETECT_EN_ABS_RSSI_DETECT_DISABLE	Disable the absolute RSSI detection	0x0*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		RSSI_DETECT_EN_ABS_RSSI_DETECT_ENABLE	Enable the absolute RSSI detection	0x1
24	RSSI_DETECT_EN_DIFF_RSSI_DETECT	RSSI_DETECT_EN_DIFF_RSSI_DETECT_DISABLE	Disable the differential RSSI detection	0x0*
		RSSI_DETECT_EN_DIFF_RSSI_DETECT_ENABLE	Enable the differential RSSI detection	0x1
23	SUBBAND_CORR_SUBBAND_CORR_EN	SUBBAND_CORR_SUBBAND_CORR_EN_DISABLE	Disable the subband correction	0x0*
		SUBBAND_CORR_SUBBAND_CORR_EN_ENABLE	Enable the subband correction	0x1
22:20	SUBBAND_CORR_SUBBAND_CORR_RX	SUBBAND_CORR_SUBBAND_CORR_RX_DEFAULT		0x0*
18:16	SUBBAND_CORR_SUBBAND_CORR_TX	SUBBAND_CORR_SUBBAND_CORR_TX_DEFAULT		0x0*
11	TXRX_CONF_INV_CLK_PLL_TX	TXRX_CONF_INV_CLK_PLL_TX_DISABLE		0x0*
		TXRX_CONF_INV_CLK_PLL_TX_ENABLE		0x1
10	TXRX_CONF_INV_CLK_DIG_TX	TXRX_CONF_INV_CLK_DIG_TX_DISABLE		0x0*
		TXRX_CONF_INV_CLK_DIG_TX_ENABLE		0x1
9:8	TXRX_CONF_SB_WAIT_T_TX	TXRX_CONF_SB_WAIT_T_TX_DEFAULT		0x0*
7	PA_RAMPUP_FULL_PA_RAMPUP	PA_RAMPUP_FULL_PA_RAMPUP_DISABLE	The PA rampup doesn't use the PA backoff enable bit	0x0
		PA_RAMPUP_FULL_PA_RAMPUP_ENABLE	The PA rampup uses the PA backoff enable bit (from -40 dBm)	0x1*
6:4	PA_RAMPUP_DEL_PA_RAMPUP	PA_RAMPUP_DEL_PA_RAMPUP_DEFAULT		0x4*
3:2	PA_RAMPUP_TAU_PA_RAMPUP	PA_RAMPUP_TAU_PA_RAMPUP_DEFAULT		0x0*
1	PA_RAMPUP_EN_PA_RAMPDOWN	PA_RAMPUP_EN_PA_RAMPDOWN_DISABLE	Disable the PA ramp-down	0x0

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		PA_RAMPUP_EN_PA_RAMPDOWN_ENABLE	Enable the PA ramp-down	0x1*
0	PA_RAMPUP_EN_PA_RAMPUP	PA_RAMPUP_EN_PA_RAMPUP_DISABLE	Disable the PA ramp-up	0x0
		PA_RAMPUP_EN_PA_RAMPUP_ENABLE	Enable the PA ramp-up	0x1*

5.9.0.51 RF_DEMOD_CTRL

Bit Field	Read/Write	Field Name	Description
31	RW	SYNC_WORD_CORR_EN_SYNC_WORD_CORR	Sync word bias correction with RSSI detection (banked)
29:24	RW	SYNC_WORD_CORR_SYNC_WORD_BIAS	Set the sync word bias (banked)
23:16	RW	RSSI_DETECT_ABS_THR_RSSI_DET_ABS_THR	Threshold used for absolute RSSI detection
15:8	RW	RSSI_DETECT_DIFF_THR_RSSI_DET_DIFF_THR	Threshold used for differential RSSI detection
7	RW	DEMOD_CTRL_DL_SYNC_NO_DATA	No data going through the demodulator, until the delay line detects the sync word (banked)
6	RW	DEMOD_CTRL_EN_DELLINE_SYNC_DET	Sync word detection in the delay line (banked)
5	RW	DEMOD_CTRL_RSSI_DET_FILT	Additional filtering on the RSSI value (banked)
4	RW	DEMOD_CTRL_EN_FAST_CLK_RECOV	Clock recovery during the resto of the preamble (banked)
3	RW	DEMOD_CTRL_EN_MIN_MAX_MF	Min max algo after the matched filter (banked)
2	RW	DEMOD_CTRL_EN_PRE_SYNC	Sync detection on the non-delayed path (banked)
1	RW	DEMOD_CTRL_BLOCK_RSSI_DET	RSSI detection during the slow-down period (banked)
0	RW	DEMOD_CTRL_EARLY_FINE_RECOV	Early fine recovery after the packet detection or pre-sync (banked)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31	SYNC_WORD_CORR_EN_SYNC_WORD_CORR	SYNC_WORD_CORR_EN_SYNC_WORD_CORR_DISABLE	Disable the sync word bias correction with RSSI detection	0x0
		SYNC_WORD_CORR_EN_SYNC_WORD_CORR_ENABLE	Enable the sync word bias correction with RSSI detection	0x1*
29:24	SYNC_WORD_CORR_SYNC_WORD_BIAS	SYNC_WORD_CORR_SYNC_WORD_BIAS_DEFAULT	Without the phase ADC rescaler, it's 8*mod_idx	0x8*
23:16	RSSI_DETECT_ABS_THR_RSSI_DET_ABS_THR	RSSI_DETECT_ABS_THR_RSSI_DET_ABS_THR_DEFAULT		0x0*
15:8	RSSI_DETECT_DIFF_THR_RSSI_DET_DIFF_THR	RSSI_DETECT_DIFF_THR_RSSI_DET_DIFF_THR_DEFAULT		0x0*
7	DEMOD_CTRL_DL_SYNC_NO_DATA	DEMOD_CTRL_DL_SYNC_NO_DATA_DISABLE		0x0
		DEMOD_CTRL_DL_SYNC_NO_DATA_ENABLE		0x1*
6	DEMOD_CTRL_EN_DELLINE_SYNC_DET	DEMOD_CTRL_EN_DELLINE_SYNC_DET_DISABLE	Disable the sync word detection in the delay line	0x0
		DEMOD_CTRL_EN_DELLINE_SYNC_DET_ENABLE	Enable the sync word detection in the delay line	0x1*
5	DEMOD_CTRL_RSSI_DET_FILT	DEMOD_CTRL_RSSI_DET_FILT_DISABLE	Dont' add an additional filtering on the RSSI value	0x0*
		DEMOD_CTRL_RSSI_DET_FILT_ENABLE	Add an additional filtering on the RSSI value	0x1
4	DEMOD_CTRL_EN_FAST_CLK_RECOV	DEMOD_CTRL_EN_FAST_CLK_RECOV_NOMINAL	Keep nominal clock recovery during the resto of the preamble	0x0*
		DEMOD_CTRL_EN_FAST_CLK_RECOV_SPEED	Speed up the clock recovery during the resto of the preamble	0x1
3	DEMOD_CTRL_EN_MIN_MAX_MF	DEMOD_CTRL_EN_MIN_MAX_MF_DISABLE	Disable the min max algo after the matched filter	0x0*
		DEMOD_CTRL_EN_MIN_MAX_MF_ENABLE	Enable the min max algo after the matched filter	0x1
2	DEMOD_CTRL_EN_PRE_SYNC	DEMOD_CTRL_EN_PRE_SYNC_DISABLE	Disable the sync detection on the non-delayed path	0x0*
		DEMOD_CTRL_EN_PRE_SYNC_ENABLE	Enable the sync detection on the non-delayed path	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
1	DEMOD_CTRL_BLOCK_RSSI_DET	DEMOD_CTRL_BLOCK_RSSI_DET_DISABLE	Keep the rssi detection during the slow-down period	0x0*
		DEMOD_CTRL_BLOCK_RSSI_DET_ENABLE	Block the rssi detection during the slow-down period	0x1
0	DEMOD_CTRL_EARLY_FINE_RECOV	DEMOD_CTRL_EARLY_FINE_RECOV_DISABLE	Disable the early fine recovery	0x0*
		DEMOD_CTRL_EARLY_FINE_RECOV_ENABLE	Enable the early fine recovery	0x1

5.9.0.52 RF_REG33

Bit Field	Read/Write	Field Name	Description
26:24	RW	CK_DIV_1_6_CK_DIV_1_6	Clock division factor for ck_div_1_6
23:16	RW	SPARES_SPARES	Spare bits
14	RW	PADS_PE_DS_GPIO_DS	Increased drive strength of the digital pads
13	RW	PADS_PE_DS_GPIO_PE	Pull-up of the GPIO pads
12	RW	PADS_PE_DS_NRESET_PE	Pull-up of the NRESET pads
11	RW	PADS_PE_DS_SPI_MISO_PE	Pull-up of the SPI MISO pads
10	RW	PADS_PE_DS_SPI_MOSI_PE	Pull-up of the SPI MOSI pads
9	RW	PADS_PE_DS_SPI_SCLK_PE	Pull-up of the SPI CLK pads
8	RW	PADS_PE_DS_SPI_CS_N_PE	Pull-up of the SPI CSN pads
7:6	RW	SUBBAND_FLL_SB_FLL_DITHER	Select the dithering
5:4	RW	SUBBAND_FLL_SB_FLL_CIC_TAU	Set the CIC decimator factor
3	RW	SUBBAND_FLL_SB_FLL_PH_4_N8	Phases in the frequency detector
2:0	RW	SUBBAND_FLL_SB_FLL_WAIT	Set the number of CIC samples before stopping the FLL

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
26:24	CK_DIV_1_6_CK_DIV_1_6	CK_DIV_1_6_NO_CLOCK	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
14	PADS_PE_DS_GPIO_DS	PADS_PE_DS_GPIO_DS_DISABLE	Disable the increased drive strength of the digital pads	0x0*
		PADS_PE_DS_GPIO_DS_ENABLE	Enable the increased drive strength of the digital pads	0x1
13	PADS_PE_DS_GPIO_PE	PADS_PE_DS_GPIO_PE_DISABLE	Disable the pull-up of the GPIO pads	0x0*
		PADS_PE_DS_GPIO_PE_ENABLE	Enable the pull-up of the GPIO pads	0x1
12	PADS_PE_DS_NRESET_PE	PADS_PE_DS_NRESET_PE_DISABLE	Disable the pull-up of the NRESET pads	0x0*
		PADS_PE_DS_NRESET_PE_ENABLE	Enable the pull-up of the NRESET pads	0x1
11	PADS_PE_DS_SPI_MISO_PE	PADS_PE_DS_SPI_MISO_PE_DISABLE	Disable the pull-up of the SPI MISO pads	0x0*
		PADS_PE_DS_SPI_MISO_PE_ENABLE	Enable the pull-up of the SPI MISO pads	0x1
10	PADS_PE_DS_SPI_MOSI_PE	PADS_PE_DS_SPI_MOSI_PE_DISABLE	Disable the pull-up of the SPI MOSI pads	0x0*
		PADS_PE_DS_SPI_MOSI_PE_ENABLE	Enable the pull-up of the SPI MOSI pads	0x1
9	PADS_PE_DS_SPI_SCLK_PE	PADS_PE_DS_SPI_SCLK_PE_DISABLE	Disable the pull-up of the SPI CLK pads	0x0*
		PADS_PE_DS_SPI_SCLK_PE_ENABLE	Enable the pull-up of the SPI CLK pads	0x1
8	PADS_PE_DS_SPI_CS_N_PE	PADS_PE_DS_SPI_CS_N_PE_DISABLE	Disable the pull-up of the SPI CSN pads	0x0*
		PADS_PE_DS_SPI_CS_N_PE_ENABLE	Enable the pull-up of the SPI CSN pads	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
7:6	SUBBAND_FLL_SB_FLL_DITHER	SUBBAND_FLL_SB_FLL_DITHER_OFF	No dithering	0x0*
		SUBBAND_FLL_SB_FLL_DITHER_PN9	PN9 positive	0x1
		SUBBAND_FLL_SB_FLL_DITHER_PN10	PN10 negative	0x2
		SUBBAND_FLL_SB_FLL_DITHER_PN9_PN10	PN9+PN10	0x3
5:4	SUBBAND_FLL_SB_FLL_CIC_TAU	SUBBAND_FLL_SB_FLL_CIC_TAU_16	Decimate by 16	0x0
		SUBBAND_FLL_SB_FLL_CIC_TAU_32	Decimate by 32	0x1
		SUBBAND_FLL_SB_FLL_CIC_TAU_64	Decimate by 64	0x2
		SUBBAND_FLL_SB_FLL_CIC_TAU_128	Decimate by 128	0x3*
3	SUBBAND_FLL_SB_FLL_PH_4_N8	SUBBAND_FLL_SB_FLL_PH_4_N8_8	Use 8 phases in the frequency detector	0x0*
		SUBBAND_FLL_SB_FLL_PH_4_N8_4	Use 4 phases in the frequency detector	0x1
2:0	SUBBAND_FLL_SB_FLL_WAIT	SUBBAND_FLL_SB_FLL_WAIT_DEFAULT		0x3*

5.9.0.53 RF_REG34

Bit Field	Read/Write	Field Name	Description
29:24	RW	CLK_RECOVERY_CLK_RECOV_CORR	Number of samples that covers the clock recovery correlator
23:16	RW	CLK_RECOVERY_CLK_AB_LIMIT	Time constant for switch the clock phase if chosen wrong in clk recovery algorithm
15	RW	TX_PRE_DIST_EN_PRE_DIST	Tx pre-distortion filter (banked)
13:8	RW	TX_PRE_DIST_PRE_DIST_B0	Coefficient b0 of the Tx pre-distortion filter (banked)
5:0	RW	TX_PRE_DIST_PRE_DIST_A0	Coefficient a0 of the Tx pre-distortion filter (banked)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
29:24	CLK_RECOVERY_CLK_RECOV_CORR	CLK_RECOVERY_CLK_RECOV_CORR_DEFAULT		0x4*
23:16	CLK_RECOVERY_CLK_AB_LIMIT	CLK_RECOVERY_CLK_AB_LIMIT_DEFAULT		0x80*
15	TX_PRE_DIST_EN_PRE_DIST	TX_PRE_DIST_EN_PRE_DIST_DISABLE	Disable the Tx pre-distortion filter	0x0
		TX_PRE_DIST_EN_PRE_DIST_ENABLE	Enable the Tx pre-distortion filter	0x1*
13:8	TX_PRE_DIST_PRE_DIST_B0	TX_PRE_DIST_PRE_DIST_B0_DEFAULT	The coefficient value is the (pre_dist_b0+64)/128	0x2E*
5:0	TX_PRE_DIST_PRE_DIST_A0	TX_PRE_DIST_PRE_DIST_A0_DEFAULT	The coefficient value is the (pre_dist_a0+64)/128	0x2F*

5.9.0.54 RF_BLE_LR

Bit Field	Read/Write	Field Name	Description
30:24	RW	BLR_SYNC_THRESHOLD_BLE_SYNC_THR	Threshold for the BLR sync word detector
19:16	RW	BLR_PREAMBLE_BLE_PRE_THR	Threshold for the BLR preamble detector
15	RW	BLE_LONG_RANGE_BLR_PUT_RI_FIFO	During the reception the RI (rate indicator) is put into the Rx FIFO (banked)
14	RW	BLE_LONG_RANGE_BLR500_NO_ROUGH	Rough recovery is stopped during the 500kbps payloads of BLR packets (banked)
13	RW	BLE_LONG_RANGE_BLR_LIN_FILTER	Matched filter (banked)
12	RW	BLE_LONG_RANGE_EN_BLR_FLUSH	Viterbi path 0 flushing at the end of the packet (banked)
11	RW	BLE_LONG_RANGE_BLR_USE_EXT_LEN	BLR_PKT_LEN for flushing out the Viterbi (banked)
10	RW	BLE_LONG_RANGE_DISABLE_BLR_TX	Long Range feature in Tx mode (banked)
9	RW	BLE_LONG_RANGE_BLR_500_N125	Data rate selection (banked)
8	RW	BLE_LONG_RANGE_EN_BLR	BLE long range mode (banked)
4	RW	HW_TRIGGER_HW_TRIG_GPIO	HW trigger is mapped on the GPIO instead of the Tx_on signal 0x0
3	RW	HW_TRIGGER_HW_TRIG_SUBBAND	Activate the sub-band selection during the Tx activation

Bit Field	Read/Write	Field Name	Description
2	RW	HW_TRIGGER_HW_TRIG_TX_NRX	Activate the Tx mode
1	RW	HW_TRIGGER_HW_TRIG_LOW	Set the trigger polarity
0	RW	HW_TRIGGER_HW_TRIG_ACTIVE	Enable HW trigger

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
30:24	BLR_SYNC_THRESHOLD_BLE_SYNC_THR	BLR_SYNC_THRESHOLD_BLE_SYNC_THR_DEFAULT	Unsigned value smaller than 64	0x38*
19:16	BLR_PREAMBLE_BLE_PRE_THR	BLR_PREAMBLE_BLE_PRE_THR_DEFAULT	Unsigned value	0x1*
15	BLE_LONG_RANGE_BLR_PUT_RI_FIFO	BLE_LONG_RANGE_BLR_PUT_RI_FIFO_DISABLE	Do not put RI in the RX FIFO	0x0
		BLE_LONG_RANGE_BLR_PUT_RI_FIFO_ENABLE	Put RI in the RX FIFO	0x1*
14	BLE_LONG_RANGE_BLR500_NO_ROUGH	BLE_LONG_RANGE_BLR500_NO_ROUGH_NO_STOP	Rough recovery is not stopped	0x0
		BLE_LONG_RANGE_BLR500_NO_ROUGH_STOP	Rough recovery is stopped	0x1*
13	BLE_LONG_RANGE_BLR_LIN_FILTER	BLE_LONG_RANGE_BLR_LIN_FILTER_DISABLE	The matched filter is not linear in BLR mode	0x0
		BLE_LONG_RANGE_BLR_LIN_FILTER_ENABLE	The matched filter is linear in BLR mode	0x1*
12	BLE_LONG_RANGE_EN_BLR_FLUSH	BLE_LONG_RANGE_EN_BLR_FLUSH_DISABLE	The Viterbi path 0 is not flushed at 2Mbps	0x0
		BLE_LONG_RANGE_EN_BLR_FLUSH_ENABLE	The Viterbi path 0 is flushed at 2Mbps	0x1*
11	BLE_LONG_RANGE_BLR_USE_EXT_LEN	BLE_LONG_RANGE_BLR_USE_EXT_LEN_NOT_USED	The value in BLR_PKT_LEN will not be used for the flush out of the Viterbi at the end of the packet	0x0*
		BLE_LONG_RANGE_BLR_USE_EXT_LEN_USED	The value in BLR_PKT_LEN will be used for the flush out of the Viterbi at the end of the packet	0x1
10	BLE_LONG_RANGE_DISABLE_BLR_TX	BLE_LONG_RANGE_DISABLE_BLR_TX_ENABLE	Long Range feature is enabled in Tx mode	0x0*
		BLE_LONG_RANGE_DISABLE_	Long Range feature is disabled in Tx	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		BLR_TX_DISABLE	mode	
9	BLE_LONG_RANGE_BLR_500_N125	BLE_LONG_RANGE_BLR_500_N125_125	125 kbps mode is used	0x0*
		BLE_LONG_RANGE_BLR_500_N125_500	500 kbps mode is used	0x1
8	BLE_LONG_RANGE_EN_BLR	BLE_LONG_RANGE_EN_BLR_DISABLE	Disable the BLE long range mode	0x0*
		BLE_LONG_RANGE_EN_BLR_ENABLE	Enable the BLE long range mode	0x1
4	HW_TRIGGER_HW_TRIG_GPIO	HW_TRIGGER_HW_TRIG_GPIO_TX_ON	HW trigger mapped on TX_ON	0x0*
		HW_TRIGGER_HW_TRIG_GPIO	HW trigger mapped on GPIO	0x1
3	HW_TRIGGER_HW_TRIG_SUBBAND	HW_TRIGGER_HW_TRIG_SUBBAND_NOT_ACTIVE	HW trigger does not active sub-band selection	0x0*
		HW_TRIGGER_HW_TRIG_SUBBAND_ACTIVE	HW trigger actives sub-band selection	0x1
2	HW_TRIGGER_HW_TRIG_TX_NRX	HW_TRIGGER_HW_TRIG_TX_NRX_RX	HW trigger actives RX mode	0x0*
		HW_TRIGGER_HW_TRIG_TX_NRX_TX	HW trigger actives TX mode	0x1
1	HW_TRIGGER_HW_TRIG_LOW	HW_TRIGGER_HW_TRIG_LOW_LOW	HW trigger active low	0x0*
		HW_TRIGGER_HW_TRIG_LOW_HIGH	HW trigger active high	0x1
0	HW_TRIGGER_HW_TRIG_ACTIVE	HW_TRIGGER_HW_TRIG_ACTIVE_DISABLE	Disable HW trigger	0x0*
		HW_TRIGGER_HW_TRIG_ACTIVE_ENABLE	Enable HW trigger	0x1

5.9.0.55 RF_REG36

Bit Field	Read/Write	Field Name	Description
30:28	RW	IQ_SPARES_EN_BIAS_SPARE	Enable for IQ spares
27:24	RW	IQ_SPARES_IQ_SPARE_2	Spare bias 2
23:20	RW	IQ_SPARES_IQ_SPARE_1	Spare bias 1
19:16	RW	IQ_SPARES_IQ_SPARE_0	Spare bias 0

Bit Field	Read/Write	Field Name	Description
8	RW	MISC_ISO_VDDA	Isolate VDDA signals
5	RW	BLR_DEMAPPER_BLR_SEND_DECODED_RI	Fully decode the rate indicator
4	RW	BLR_DEMAPPER_BLR_USE_EXT_VIT_GFSK	500kbps BLR uses the Viterbi GFSK decision
3:2	RW	BLR_DEMAPPER_BLR_500_DPHASE	Set the distance between samples for the phase to frequency conversion in S2 mode
1	RW	BLR_DEMAPPER_BLR_500_LOW_GAIN	Set the low gain in S2 mode
0	RW	BLR_DEMAPPER_BLR_125_LOW_GAIN	Set the low gain in S8 mode

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
30:28	IQ_SPARES_EN_BIAS_SPARE	IQ_SPARES_EN_BIAS_SPARE_DEFAULT		0x0*
27:24	IQ_SPARES_IQ_SPARE_2	IQ_SPARES_IQ_SPARE_2_DEFAULT		0x0*
23:20	IQ_SPARES_IQ_SPARE_1	IQ_SPARES_IQ_SPARE_1_DEFAULT		0x0*
19:16	IQ_SPARES_IQ_SPARE_0	IQ_SPARES_IQ_SPARE_0_DEFAULT		0x0*
8	MISC_ISO_VDDA	MISC_ISO_VDDA_NOT_ISOLATE	Do not isolate VDDA signals	0x0*
		MISC_ISO_VDDA_ISOLATE	Isolate VDDA signals	0x1
5	BLR_DEMAPPER_BLR_SEND_DECODED_RI	BLR_DEMAPPER_BLR_SEND_DECODED_RI_DISABLE	Rate indicator is not fully decoded and decision made is sent instead	0x0*
		BLR_DEMAPPER_BLR_SEND_DECODED_RI_ENABLE	Rate indicator is fully decoded	0x1
4	BLR_DEMAPPER_BLR_USE_EXT_VIT_GFSK	BLR_DEMAPPER_BLR_USE_EXT_VIT_GFSK_DISABLE	Do not use the Viterbi GFSK decision	0x0
		BLR_DEMAPPER_BLR_USE_EXT_VIT_GFSK_ENABLE	Use the Viterbi GFSK decision	0x1*
3:2	BLR_DEMAPPER_BLR_500_DPHASE	BLR_DEMAPPER_BLR_500_DPHASE_2	2 samples	0x0
		BLR_DEMAPPER_BLR_500_DPHASE_4	4 samples	0x1
		BLR_DEMAPPER_BLR_500_	6 samples	0x2

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		DPHASE_6		
		BLR_DEMAPPER_BLR_500_DPHASE_8	8 samples	0x3*
1	BLR_DEMAPPER_BLR_500_LOW_GAIN	BLR_DEMAPPER_BLR_500_LOW_GAIN_DISABLE	Do not set low gain in S2 mode	0x0*
		BLR_DEMAPPER_BLR_500_LOW_GAIN_ENABLE	Set low gain in S2 mode	0x1
0	BLR_DEMAPPER_BLR_125_LOW_GAIN	BLR_DEMAPPER_BLR_125_LOW_GAIN_DISABLE	Do not set low gain in S8 mode	0x0*
		BLR_DEMAPPER_BLR_125_LOW_GAIN_ENABLE	Set low gain in S8 mode	0x1

5.9.0.56 RF_PROT_TIMER

Bit Field	Read/Write	Field Name	Description
31	RW	PROT_TIMER_CONF_EN_PROT_TIMER	Enable the protocol timer
29:27	RW	PROT_TIMER_CONF_PT_T_STP_1	Configure the time stamp 1
26:24	RW	PROT_TIMER_CONF_PT_T_STP_0	Configure the time stamp 0
22	RW	STAGING_PS_NZ_START_BIT	Select the frequency offset
21	RW	STAGING_PS_NZ_START	Start the pulse shaper with a +/- 250 kHz frequency offset
20	RW	STAGING_DEL_PA_RAMPDW	Delay the PA ramp-down by 4.5 us
19	RW	STAGING_PEAK_DET_TH_SHIFT	Peak detector threshold shift
18:17	RW	STAGING_AGC_DERIV_LVL	Select the AGC derivative level
16	RW	STAGING_AGC_USE_DERIV	AGC algorithm uses the derivative information to accelerate the AGC settling
15:8	RW	BLE_DTM_BLE_DTM_LEN	Set the BLE DTM packet length
7	RW	BLE_DTM_EN_BLE_DTM	Enable the BLE DTM automatic packets
3:0	RW	BLE_DTM_BLE_DTM_PKT_TYPE	Set the BLE DTM packet type (see Bluetooth specification)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31	PROT_TIMER_CONF_EN_PROT_TIMER	PROT_TIMER_CONF_EN_PROT_TIMER_DISABLE	Disable protocol timer	0x0*
		PROT_TIMER_CONF_EN_PROT_TIMER_ENABLE	Enable protocol timer	0x1
29:27	PROT_TIMER_CONF_PT_T_STP_1	PROT_TIMER_CONF_PT_T_STP_1_DEFAULT		0x0*
26:24	PROT_TIMER_CONF_PT_T_STP_0	PROT_TIMER_CONF_PT_T_STP_0_NO_STAMP	No time stamp	0x0*
		PROT_TIMER_CONF_PT_T_STP_0_TIMER_TRIGGER	Protocol timer trigger	0x1
		PROT_TIMER_CONF_PT_T_STP_0_TRIGGER_TX_STOP	TX stop trigger	0x4
		PROT_TIMER_CONF_PT_T_STP_0_TRIGGER_RX_SYNC	RX sync trigger	0x5
		PROT_TIMER_CONF_PT_T_STP_0_TRIGGER_RX_STOP	RX stop trigger	0x6
		PROT_TIMER_CONF_PT_T_STP_0_TRIGGER_RX_RECEIVED	RX received trigger	0x7
22	STAGING_PS_NZ_START_BIT	STAGING_PS_NZ_START_BIT_0	Select bit 0 (-250 kHz offset)	0x0*
		STAGING_PS_NZ_START_BIT_1	Select bit 1 (+250 kHz offset)	0x1
21	STAGING_PS_NZ_START	STAGING_PS_NZ_START_NO_OFFSET	Do not start the pulse shaper with a frequency offset	0x0*
		STAGING_PS_NZ_START_OFFSET	Start the pulse shaper with a frequency offset	0x1
20	STAGING_DEL_PA_RAMPDW	STAGING_DEL_PA_RAMPDW_NO_DELAY	Do not delay PA ramp down	0x0*
		STAGING_DEL_PA_RAMPDW_DELAY	Delay PA ramp down	0x1
19	STAGING_PEAK_DET_TH_SHIFT	STAGING_PEAK_DET_TH_SHIFT_NO_SHIFT	Do not shift peak detector	0x0*
		STAGING_PEAK_DET_TH_SHIFT_SHIFT	Shift peak detector	0x1
18:17	STAGING_AGC_DERIV_LVL	STAGING_AGC_DERIV_LVL_16	Level 16	0x0
		STAGING_AGC_DERIV_LVL_24	Level 24	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		STAGING_AGC_DERIV_LVL_32	Level 32	0x2*
		STAGING_AGC_DERIV_LVL_48	Level 48	0x3
16	STAGING_AGC_USE_DERIV	STAGING_AGC_USE_DERIV_NOT_USED	Do not use derivative information	0x0*
		STAGING_AGC_USE_DERIV_USED	Use derivative information	0x1
15:8	BLE_DTM_BLE_DTM_LEN	BLE_DTM_BLE_DTM_LEN_DEFAULT		0x25*
7	BLE_DTM_EN_BLE_DTM	BLE_DTM_EN_BLE_DTM_DISABLE	Disable BLE DTM automatic packets	0x0*
		BLE_DTM_EN_BLE_DTM_ENABLE	Enable BLE DTM automatic packets	0x1
3:0	BLE_DTM_BLE_DTM_PKT_TYPE	BLE_DTM_BLE_DTM_PKT_TYPE_DEFAULT		0x0*

5.9.0.57 RF_CTE_OPTS

Bit Field	Read/Write	Field Name	Description
29	RW	CTE_OPTS_RECT_PS_CTE	Use rectangular pulse shape during the CTE
28	RW	CTE_OPTS_USE_CTE_WO_CP	Enable the CTE without reading or inserting the CP
27	RW	CTE_OPTS_CTE_AMPL	Enable the usage of the RSSI values to adapt the amplitude of the IQ signal based to the RSSI value
26	RW	CTE_OPTS_DF_AOA_SLOT_TIME	Indicate the switching/sampling slot period for AoA
25	RW	CTE_OPTS_CP_INSERT	Force the CP bit in the packet header to 1
24	RW	CTE_OPTS_EN_READ_CP	CP bit is read in the packet header (BLE standard)
23:16	RW	CTE_OPTS_CTE_INFO	Set the CTEInfo field in the packet header while cp_insert is set to 1
14:10	RW	ASK_MOD_ASK_MAX	Set the maximum value for the ASK modulation
9:5	RW	ASK_MOD_ASK_MIN	Set the minimum value for the ASK modulation
4:1	RW	ASK_MOD_ASK_CNT	Set the how long to count for the ASK modulation
0	RW	ASK_MOD_EN_RSSI_ASK	PA will perform an ASK modulation

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
29	CTE_OPTS_RECT_PS_CTE	CTE_OPTS_RECT_PS_CTE_DISABLE	Disable CTE with rectangular pulse shaping	0x0*
		CTE_OPTS_RECT_PS_CTE_ENABLE	Enable CTE with rectangular pulse shaping	0x1
28	CTE_OPTS_USE_CTE_WO_CP	CTE_OPTS_USE_CTE_WO_CP_DISABLE	Disable CTE without reading CP	0x0*
		CTE_OPTS_USE_CTE_WO_CP_ENABLE	Enable CTE without reading CP	0x1
27	CTE_OPTS_CTE_AMPL	CTE_OPTS_CTE_AMPL_DISABLE	Disable RSSI for IQ signal validation	0x0*
		CTE_OPTS_CTE_AMPL_ENABLE	Enable RSSI for IQ signal validation	0x1
26	CTE_OPTS_DF_AOA_SLOT_TIME	CTE_OPTS_DF_AOA_SLOT_TIME_1	1us switching period	0x0*
		CTE_OPTS_DF_AOA_SLOT_TIME_2	2us switching period	0x1
25	CTE_OPTS_CP_INSERT	CTE_OPTS_CP_INSERT_DISABLE	Disable forcing CP bit to 1	0x0*
		CTE_OPTS_CP_INSERT_ENABLE	Enable forcing CP bit to 1	0x1
24	CTE_OPTS_EN_READ_CP	CTE_OPTS_EN_READ_CP_DISABLE	Disable reading CP bit	0x0*
		CTE_OPTS_EN_READ_CP_ENABLE	Enable reading CP bit	0x1
23:16	CTE_OPTS_CTE_INFO	CTE_OPTS_CTE_INFO_DEFAULT		0x0*
14:10	ASK_MOD_ASK_MAX	ASK_MOD_ASK_MAX_DEFAULT		0xC*
9:5	ASK_MOD_ASK_MIN	ASK_MOD_ASK_MIN_DEFAULT		0x0*
4:1	ASK_MOD_ASK_CNT	ASK_MOD_ASK_CNT_DEFAULT		0x7*
0	ASK_MOD_EN_RSSI_ASK	ASK_MOD_EN_RSSI_ASK_DISABLE	Disable PA ASK modulation	0x0*
		ASK_MOD_EN_RSSI_ASK_ENABLE	Enable PA ASK modulation	0x1

5.9.0.58 RF_PT_DELTA_0

Bit Field	Read/Write	Field Name	Description
31:30	RW	PT_DELTA_TS_0_PT_DELTA_T0_MULT	Multiplier for the delta t0
19:0	RW	PT_DELTA_TS_0_PT_DELTA_T0	Delta t0 for the protocol timer

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:30	PT_DELTA_TS_0_PT_DELTA_T0_MULT	PT_DELTA_TS_0_PT_DELTA_T0_MULT_DEFAULT		0x0*
19:0	PT_DELTA_TS_0_PT_DELTA_T0	PT_DELTA_TS_0_PT_DELTA_T0_DEFAULT		0x0*

5.9.0.59 RF_PT_DELTA_1

Bit Field	Read/Write	Field Name	Description
31:30	RW	PT_DELTA_TS_1_PT_DELTA_T1_MULT	Multiplier for the delta t1
19:0	RW	PT_DELTA_TS_1_PT_DELTA_T1	Delta t1 for the protocol timer

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:30	PT_DELTA_TS_1_PT_DELTA_T1_MULT	PT_DELTA_TS_1_PT_DELTA_T1_MULT_DEFAULT		0x0*
19:0	PT_DELTA_TS_1_PT_DELTA_T1	PT_DELTA_TS_1_PT_DELTA_T1_DEFAULT		0x0*

5.9.0.60 RF_CTE_IF

Bit Field	Read/Write	Field Name	Description
25:16	RW	CTE_CTRL_DELAY_TX_DF_DELAY_TX	Delay (in 62.5ns) form the serializer up to the antenna in direction finding (banked)
15	RW	ANTENNA_CONF_DF_IND_PATTERN	Separate the antenna switching pattern from the reference one
14	RW	ANTENNA_CONF_DF_IND_ANTENNA	Make the antenna for DF independent from the rest of the packet
13:8	RW	ANTENNA_CONF_ANT_LUT_M	Number of states used (-1) in the antenna LUT
5	RW	CTE_AUTO_PULL_EXT_IQ_SMP_TYPE	Select the external IQ sample signal qualifier type
4	RW	CTE_AUTO_PULL_IQ_MSB	Select which signal is sent over the MSB in case of a 16bits buffers
3:2	RW	CTE_AUTO_PULL_IQ_DATA_BUS_SIZE	Select the bus data size of IQ signals
1	RW	CTE_AUTO_PULL_CTE_QUAL	Select the CTE data qualifier
0	RW	CTE_AUTO_PULL_EN_CTE_AUTO_PULL	Enable the automatic push of CTE data to an external IP

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
25:16	CTE_CTRL_DELAY_TX_DF_DELAY_TX	CTE_CTRL_DELAY_TX_DF_DELAY_TX_DEFAULT		0x0*
15	ANTENNA_CONF_DF_IND_PATTERN	ANTENNA_CONF_DF_IND_PATTERN_DISABLE	Disable the seperation of the antenna switching pattern	0x0*
		ANTENNA_CONF_DF_IND_PATTERN_ENABLE	Enable the seperation of the antenna switching pattern	0x1
14	ANTENNA_CONF_DF_IND_ANTENNA	ANTENNA_CONF_DF_IND_ANTENNA_DISABLE	Disable the independancy of the antenna	0x0*
		ANTENNA_CONF_DF_IND_ANTENNA_ENABLE	Enable the independancy of the antenna	0x1
13:8	ANTENNA_CONF_ANT_LUT_M	ANTENNA_CONF_ANT_LUT_M_DEFAULT		0x0*
5	CTE_AUTO_PULL_EXT_IQ_SMP_TYPE	CTE_AUTO_PULL_EXT_IQ_SMP_TYPE_PULSE	Pulse	0x0*
		CTE_AUTO_PULL_EXT_IQ_SMP_TYPE_TOGGLE	Toggle	0x1
4	CTE_AUTO_PULL_IQ_MSB	CTE_AUTO_PULL_IQ_MSB_Q	I LSB, Q MSB	0x0*
		CTE_AUTO_PULL_IQ_MSB_I	Q LSB, I MSB	0x1
3:2	CTE_AUTO_PULL_IQ_DATA_BUS_SIZE	CTE_AUTO_PULL_IQ_DATA_BUS_SIZE_4	4 bits	0x0*
		CTE_AUTO_PULL_IQ_DATA_BUS_SIZE_8	8 bits	0x1
		CTE_AUTO_PULL_IQ_DATA_BUS_SIZE_16	16 bits	0x2
1	CTE_AUTO_PULL_CTE_QUAL	CTE_AUTO_PULL_CTE_QUAL_TOGGLE	Toggling signal	0x0*
		CTE_AUTO_PULL_CTE_QUAL_CLOCK	Clock signal	0x1
0	CTE_AUTO_PULL_EN_CTE_AUTO_PULL	CTE_AUTO_PULL_EN_CTE_AUTO_PULL_DISABLE	Disable automatic push of CTE data	0x0*
		CTE_AUTO_PULL_EN_CTE_AUTO_PULL_ENABLE	Enable automatic push of CTE data	0x1

5.9.0.61 RF_CTE_CTRL

Bit Field	Read/Write	Field Name	Description
25:16	RW	CTE_CTRL_DELAY_RX_DF_DELAY_SWITCH_RX	Delay (in 62.5ns) from the antenna up to the deserializer in direction finding (banked)
9:0	RW	CTE_CTRL_DELAY_RX_DF_DELAY_SAMPLE_RX	Delay (in 62.5ns) from the matched filter up to the deserializer in direction finding (banked)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
25:16	CTE_CTRL_DELAY_RX_DF_DELAY_SWITCH_RX	CTE_CTRL_DELAY_RX_DF_DELAY_SWITCH_RX_DEFAULT		0x0*
9:0	CTE_CTRL_DELAY_RX_DF_DELAY_SAMPLE_RX	CTE_CTRL_DELAY_RX_DF_DELAY_SAMPLE_RX_DEFAULT		0x0*

5.9.0.62 RF_AGC_ADVANCED

Bit Field	Read/Write	Field Name	Description
26:16	RW	AGC_SWITCHES_AGC_SHORTS_LUT	Array of values that indicates if the highpass shorts must be set for the AGC state passage from n -> n+1
8	RW	DEBUG_FAKE_IQ_SAMPLES	Generate fake IQ samples
7:4	RW	AGC_ADVANCED_AGC_TAU_SHORTS	Time constant that indicates the time that shorts must be on
3	RW	AGC_ADVANCED_AGC_EN_SHORT_PHADC	Enable the short on the phase ADC highpass filter
2	RW	AGC_ADVANCED_AGC_EN_SHORT_IFA	Enable the short on the IFA highpass filter
1	RW	AGC_ADVANCED_AGC_USE_SHORTS	Enable the usage of the shorts located in the BB path
0	RW	AGC_ADVANCED_AGC_FULL_SPEED	Enable the maximum speed in AGC

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
26:16	AGC_SWITCHES_AGC_SHORTS_LUT	AGC_SWITCHES_AGC_SHORTS_LUT_DEFAULT		0x0*
8	DEBUG_FAKE_IQ_SAMPLES	DEBUG_FAKE_IQ_SAMPLES_DISABLE	Disable fake IQ samples	0x0*
		DEBUG_FAKE_IQ_SAMPLES_ENABLE	Enable fake IQ samples	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
7:4	AGC_ADVANCED_AGC_TAU_SHORTS	AGC_ADVANCED_AGC_TAU_SHORTS_DEFAULT		0x0*
3	AGC_ADVANCED_AGC_EN_SHORT_PHADC	AGC_ADVANCED_AGC_EN_SHORT_PHADC_DISABLE	Disable the short on the phase ADC highpass filter	0x0*
		AGC_ADVANCED_AGC_EN_SHORT_PHADC_ENABLE	Enable the short on the phase ADC highpass filter	0x1
2	AGC_ADVANCED_AGC_EN_SHORT_IFA	AGC_ADVANCED_AGC_EN_SHORT_IFA_DISABLE	Disable the short on the IFA highpass filter	0x0*
		AGC_ADVANCED_AGC_EN_SHORT_IFA_ENABLE	Enable the short on the IFA highpass filter	0x1
1	AGC_ADVANCED_AGC_USE_SHORTS	AGC_ADVANCED_AGC_USE_SHORTS_DISABLE	Disable using shorts located in BB path	0x0*
		AGC_ADVANCED_AGC_USE_SHORTS_ENABLE	Enable using shorts located in BB path	0x1
0	AGC_ADVANCED_AGC_FULL_SPEED	AGC_ADVANCED_AGC_FULL_SPEED_DISABLE	Disable maximum AGC speed	0x0*
		AGC_ADVANCED_AGC_FULL_SPEED_ENABLE	Enable maximum speed AGC	0x1

5.9.0.63 RF_DATA_STREAMING

Bit Field	Read/Write	Field Name	Description
9	RW	DATA_STREAMING_DMA_PHASE_TYPE	Use the phase after the rescaler instead of the raw phase from phase ADC (banked)
8	RW	DATA_STREAMING_DMA_EN_BUS	Enable the DMA bus on the IP interface (banked)
6	RW	DATA_STREAMING_PERIODIC_SAMPLE_AFTER_CTE	Restart sampling after the CTE period (banked)
5	RW	DATA_STREAMING_PERIODIC_SAMPLE_AT_SYNC	Start sampling at the sync detection signal from the delay line (banked)
4	RW	DATA_STREAMING_PERIODIC_SAMPLE_OSR_CLK	Oversample (8x) the reference clock of the periodic sample (banked)
3:1	RW	DATA_STREAMING_PERIODIC_SAMPLE_OSR	Division factor (-1) of for the sampling period of the periodic IQ sampling (banked)
0	RW	DATA_STREAMING_PERIODIC_SAMPLE_EN_IQ	Sample periodically I and Q channels after the matched filter and put into the IQ FIFO (banked)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
9	DATA_STREAMING_DMA_PHASE_TYPE	DATA_STREAMING_DMA_PHASE_TYPE_PHADC	Use the raw phase from the phase ADC	0x0*
		DATA_STREAMING_DMA_PHASE_TYPE_RESCALER	Use the phase after the rescaler	0x1
8	DATA_STREAMING_DMA_EN_BUS	DATA_STREAMING_DMA_BUS_DISABLE	Enable the DMA bus on the IP interface	0x0*
		DATA_STREAMING_DMA_BUS_ENABLE	Disable the DMA bus on the IP interface	0x1
6	DATA_STREAMING_PERIODIC_SAMPLE_AFTER_CTE	DATA_STREAMING_PERIODIC_SAMPLE_AFTER_CTE_DISABLE	Stop sampling after the CTE event	0x0
		DATA_STREAMING_PERIODIC_SAMPLE_AFTER_CTE_ENABLE	Restart sampling after the CTE event	0x1*
5	DATA_STREAMING_PERIODIC_SAMPLE_AT_SYNC	DATA_STREAMING_PERIODIC_SAMPLE_AT_SYNC_DISABLE	Sample at the Rx activation	0x0*
		DATA_STREAMING_PERIODIC_SAMPLE_AT_SYNC_ENABLE	Sample at the sync detection signal from the delay line	0x1
4	DATA_STREAMING_PERIODIC_SAMPLE_OSR_CLK	DATA_STREAMING_PERIODIC_SAMPLE_OSR_CLK_DISABLE	Use the system clock	0x0*
		DATA_STREAMING_PERIODIC_SAMPLE_OSR_CLK_ENABLE	Oversample the reference clock of the periodic sample	0x1
3:1	DATA_STREAMING_PERIODIC_SAMPLE_OSR	DATA_STREAMING_PERIODIC_SAMPLE_OSR_DEFAULT		0x3*
0	DATA_STREAMING_PERIODIC_SAMPLE_EN_IQ	DATA_STREAMING_PERIODIC_SAMPLE_IQ_DISABLE	Disable periodical sampling	0x0*
		DATA_STREAMING_PERIODIC_SAMPLE_IQ_ENABLE	Enable periodical sampling	0x1

5.9.0.64 RF_REVISION

Bit Field	Read/Write	Field Name	Description
31:24	R	CHIP_ID	Remapped register of CHIP_ID

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:24	CHIP_ID	CHIP_ID_DEFAULT	The remapped registers are accessible over SPI only	0x30*

5.9.0.65 RF_FSM_CTRL

Bit Field	Read/Write	Field Name	Description
31:30	R	RXFIFO_STATUS_RX_BIST_ERRORS	Rx FIFO BIST result
31:25	W	RXFIFO_STATUS_RX_BIST	Start the bist test on the Rx FIFO (code 0x5d)
29	R	RXFIFO_STATUS_RX_NEAR_UNDERFLOW	Rx FIFO near underflow
28	R	RXFIFO_STATUS_RX_NEAR_OVERFLOW	Rx FIFO near overflow
27	R	RXFIFO_STATUS_RX_UNDERFLOW	Rx FIFO underflow
26	R	RXFIFO_STATUS_RX_OVERFLOW	Rx FIFO overflow
25	R	RXFIFO_STATUS_RX_FULL	Rx FIFO full
24	R	RXFIFO_STATUS_RX_EMPTY	Rx FIFO empty
24	W	RXFIFO_STATUS_RX_FLUSH	Rx FIFO flush
23:22	R	TXFIFO_STATUS_TX_BIST_ERRORS	Tx FIFO BIST result
23:17	W	TXFIFO_STATUS_TX_BIST	Start the bist test on the Tx FIFO (code 0x5d)
21	R	TXFIFO_STATUS_TX_NEAR_UNDERFLOW	Tx FIFO near underflow
20	R	TXFIFO_STATUS_TX_NEAR_OVERFLOW	Tx FIFO near overflow
19	R	TXFIFO_STATUS_TX_UNDERFLOW	Tx FIFO underflow
18	R	TXFIFO_STATUS_TX_OVERFLOW	Tx FIFO overflow
17	R	TXFIFO_STATUS_TX_FULL	Tx FIFO full
16	R	TXFIFO_STATUS_TX_EMPTY	Tx FIFO empty
16	W	TXFIFO_STATUS_TX_FLUSH	Tx FIFO flush
10	R	FSM_STATUS_TX_NRX	Select Rx or Tx mode
9:8	R	FSM_STATUS_STATUS	Status of the FSM

Bit Field	Read/Write	Field Name	Description
3	W	FSM_MODE_RESET	FSM reset
2	R	FSM_MODE_RX_MODE	Rx status
2	W	FSM_MODE_TX_NRX	Set the radio in Tx or Rx mode
1	R	FSM_MODE_TX_MODE	Tx status
1:0	W	FSM_MODE_MODE	Set the FSM mode
0	R	FSM_MODE_N_IDLE	FSM status

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:30	RXFIFO_STATUS_RX_BIST_ERRORS	RXFIFO_STATUS_RX_BIST_NO_ERROR	No error	0x0*
		RXFIFO_STATUS_RX_BIST_CKBD	Error in checkboard test	0x1
		RXFIFO_STATUS_RX_BIST_ICKBD	Error in inversed checkboard test	0x2
		RXFIFO_STATUS_RX_BIST_DECODER	Error in decoder test	0x3
31:25	RXFIFO_STATUS_RX_BIST	RXFIFO_STATUS_RX_BIST_DEFAULT		0x0
29	RXFIFO_STATUS_RX_NEAR_UNDERFLOW	RXFIFO_STATUS_RX_NO_NEAR_UNDERFLOW	No Rx FIFO near underflow occurred	0x0*
		RXFIFO_STATUS_RX_NEAR_UNDERFLOW	Rx FIFO near underflow occurred	0x1
28	RXFIFO_STATUS_RX_NEAR_OVERFLOW	RXFIFO_STATUS_RX_NO_NEAR_OVERFLOW	No Rx FIFO near overflow occurred	0x0*
		RXFIFO_STATUS_RX_NEAR_OVERFLOW	Rx FIFO near verflow occurred	0x1
27	RXFIFO_STATUS_RX_UNDERFLOW	RXFIFO_STATUS_RX_NO_UNDERFLOW	No Rx FIFO underflow occurred	0x0*
		RXFIFO_STATUS_RX_UNDERFLOW	Rx FIFO underflow occurred	0x1
26	RXFIFO_STATUS_RX_OVERFLOW	RXFIFO_STATUS_RX_NO_OVERFLOW	No Rx FIFO overflow occurred	0x0*
		RXFIFO_STATUS_RX_OVERFLOW	Rx FIFO overflow occurred	0x1
25	RXFIFO_STATUS_RX_FULL	RXFIFO_STATUS_RX_NOT_FULL	Rx FIFO is not full	0x0*
		RXFIFO_STATUS_RX_FULL	Rx FIFO is full	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
24	RXFIFO_STATUS_RX_EMPTY	RXFIFO_STATUS_RX_NOT_EMPTY	Rx FIFO is not empty	0x0*
		RXFIFO_STATUS_RX_EMPTY	Rx FIFO is empty	0x1
24	RXFIFO_STATUS_RX_FLUSH	RXFIFO_STATUS_NO_FLUSH	Don't flush Rx FIFO	0x0
		RXFIFO_STATUS_FLUSH	Flush Rx FIFO	0x1
23:22	TXFIFO_STATUS_TX_BIST_ERRORS	TXFIFO_STATUS_TX_BIST_NO_ERROR	No error	0x0*
		TXFIFO_STATUS_TX_BIST_CKBD	Error in checkboard test	0x1
		TXFIFO_STATUS_TX_BIST_ICKBD	Error in inversed checkboard test	0x2
		TXFIFO_STATUS_TX_BIST_DECODER	Error in decoder test	0x3
23:17	TXFIFO_STATUS_TX_BIST	TXFIFO_STATUS_TX_BIST_DEFAULT		0x0
21	TXFIFO_STATUS_TX_NEAR_UNDERFLOW	TXFIFO_STATUS_TX_NO_NEAR_UNDERFLOW	No Tx FIFO near underflow occurred	0x0*
		TXFIFO_STATUS_TX_NEAR_UNDERFLOW	Tx FIFO near underflow occurred	0x1
20	TXFIFO_STATUS_TX_NEAR_OVERFLOW	TXFIFO_STATUS_TX_NO_NEAR_OVERFLOW	No Tx FIFO near overflow occurred	0x0*
		TXFIFO_STATUS_TX_NEAR_OVERFLOW	Tx FIFO near verflow occurred	0x1
19	TXFIFO_STATUS_TX_UNDERFLOW	TXFIFO_STATUS_TX_NO_UNDERFLOW	No Tx FIFO underflow occurred	0x0*
		TXFIFO_STATUS_TX_UNDERFLOW	Tx FIFO underflow occurred	0x1
18	TXFIFO_STATUS_TX_OVERFLOW	TXFIFO_STATUS_TX_NO_OVERFLOW	No Tx FIFO overflow occurred	0x0*
		TXFIFO_STATUS_TX_OVERFLOW	Tx FIFO overflow occurred	0x1
17	TXFIFO_STATUS_TX_FULL	TXFIFO_STATUS_TX_NOT_FULL	Tx FIFO is not full	0x0*
		TXFIFO_STATUS_TX_FULL	Tx FIFO is full	0x1
16	TXFIFO_STATUS_TX_EMPTY	TXFIFO_STATUS_TX_NOT_EMPTY	Tx FIFO is not empty	0x0*
		TXFIFO_STATUS_TX_EMPTY	Tx FIFO is empty	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
16	TXFIFO_STATUS_TX_FLUSH	TXFIFO_STATUS_NO_FLUSH	Don't flush Tx FIFO	0x0
		TXFIFO_STATUS_FLUSH	Flush Tx FIFO	0x1
10	FSM_STATUS_TX_NRX	FSM_STATUS_RX_NTX	Radio is in Rx mode	0x0*
		FSM_STATUS_TX_NRX	Radio is in Tx mode	0x1
9:8	FSM_STATUS_STATUS	FSM_STATUS_IDLE	Nothing is done	0x0*
		FSM_STATUS_TX	Tx mode	0x1
		FSM_STATUS_RX	Rx mode	0x2
		FSM_STATUS_SUSPEND	Suspend	0x3
3	FSM_MODE_RESET	FSM_MODE_NOT_RESET	Not reset FSM	0x0
		FSM_MODE_RESET	Reset FSM	0x1
2	FSM_MODE_RX_MODE	FSM_MODE_RX_MODE_OFF	No ongoing Rx	0x0*
		FSM_MODE_RX_MODE_ON	Rx is ongoing	0x1
2	FSM_MODE_TX_NRX	FSM_MODE_RX_NTX	Set the radio in Rx	0x0
		FSM_MODE_TX_NRX	Set the radio in Tx	0x1
1	FSM_MODE_TX_MODE	FSM_MODE_TX_MODE_OFF	No ongoing Tx	0x0*
		FSM_MODE_TX_MODE_ON	Tx is ongoing	0x1
1:0	FSM_MODE_MODE	FSM_MODE_MODE_IDLE	Nothing is done	0x0
		FSM_MODE_MODE_ACTIVATE	Activate	0x1
		FSM_MODE_MODE_CAL_PLL	Calibrate the PLL	0x2
		FSM_MODE_MODE_CAL_PLL_TXRX	Calibrate the PLL then Tx/Rx	0x3
0	FSM_MODE_N_IDLE	FSM_MODE_IDLE	FSM is in the Idle mode	0x0*
		FSM_MODE_N_IDLE	FSM is not in the Idle mode	0x1

5.9.0.66 RF_IQFIFO_STATUS

Bit Field	Read/Write	Field Name	Description
24:16	R	TXFIFO_COUNT_TX_COUNT	Number of bytes in the Tx FIFO
15:8	R	IQFIFO_COUNT_IQ_COUNT	Number of bytes in the IQ FIFO
7:6	R	IQFIFO_STATUS_IQ_BIST_ERRORS	IQ FIFO BIST result
7:1	W	IQFIFO_STATUS_IQ_BIST	Start the BIST test on the IQ FIFO (code 0x5d)

Bit Field	Read/Write	Field Name	Description
5	R	IQFIFO_STATUS_IQ_NEAR_UNDERFLOW	IQ FIFO near underflow
4	R	IQFIFO_STATUS_IQ_NEAR_OVERFLOW	IQ FIFO near overflow
3	R	IQFIFO_STATUS_IQ_UNDERFLOW	IQ FIFO underflow
2	R	IQFIFO_STATUS_IQ_OVERFLOW	IQ FIFO overflow
1	R	IQFIFO_STATUS_IQ_FULL	IQ FIFO full
0	R	IQFIFO_STATUS_IQ_EMPTY	IQ FIFO empty
0	W	IQFIFO_STATUS_FLUSH	IQ FIFO flush

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
24:16	TXFIFO_COUNT_TX_COUNT	TXFIFO_COUNT_TX_COUNT_DEFAULT		0x0*
15:8	IQFIFO_COUNT_IQ_COUNT	IQFIFO_COUNT_IQ_COUNT_DEFAULT		0x0*
7:6	IQFIFO_STATUS_IQ_BIST_ERRORS	IQFIFO_STATUS_IQ_BIST_NO_ERROR	No error	0x0*
		IQFIFO_STATUS_IQ_BIST_CKBD	Error in checkboard test	0x1
		IQFIFO_STATUS_IQ_BIST_ICKBD	Error in inversed checkboard test	0x2
		IQFIFO_STATUS_IQ_BIST_DECODER	Error in decoder test	0x3
7:1	IQFIFO_STATUS_IQ_BIST	IQFIFO_STATUS_IQ_BIST_DEFAULT		0x0
5	IQFIFO_STATUS_IQ_NEAR_UNDERFLOW	IQFIFO_STATUS_IQ_NO_NEAR_UNDERFLOW	No Tx FIFO near underflow occurred	0x0*
		IQFIFO_STATUS_IQ_NEAR_UNDERFLOW	Tx FIFO near underflow occurred	0x1
4	IQFIFO_STATUS_IQ_NEAR_OVERFLOW	IQFIFO_STATUS_IQ_NO_NEAR_OVERFLOW	No Tx FIFO near overflow occurred	0x0*
		IQFIFO_STATUS_IQ_NEAR_OVERFLOW	Tx FIFO near overflow occurred	0x1
3	IQFIFO_STATUS_IQ_UNDERFLOW	IQFIFO_STATUS_IQ_NO_UNDERFLOW	No Tx FIFO underflow occurred	0x0*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		IQFIFO_STATUS_IQ_UNDERFLOW	Tx FIFO underflow occurred	0x1
2	IQFIFO_STATUS_IQ_OVERFLOW	IQFIFO_STATUS_IQ_NO_OVERFLOW	No Tx FIFO overflow occurred	0x0*
		IQFIFO_STATUS_IQ_OVERFLOW	Tx FIFO overflow occurred	0x1
1	IQFIFO_STATUS_IQ_FULL	IQFIFO_STATUS_IQ_NOT_FULL	Tx FIFO is not full	0x0*
		IQFIFO_STATUS_IQ_FULL	Tx FIFO is full	0x1
0	IQFIFO_STATUS_IQ_EMPTY	IQFIFO_STATUS_IQ_NOT_EMPTY	Tx FIFO is not empty	0x0*
		IQFIFO_STATUS_IQ_EMPTY	Tx FIFO is empty	0x1
0	IQFIFO_STATUS_FLUSH	IQFIFO_STATUS_NO_FLUSH	Don't flush Tx FIFO	0x0
		IQFIFO_STATUS_FLUSH	Flush Tx FIFO	0x1

5.9.0.67 RF_TXFIFO

Bit Field	Read/Write	Field Name	Description
7:0	W	TXFIFO_TX_DATA	Data to be sent

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
7:0	TXFIFO_TX_DATA	TXFIFO_TX_DATA_DEFAULT		0x0

5.9.0.68 RF_RXFIFO

Bit Field	Read/Write	Field Name	Description
7:0	R	RXFIFO_RX_DATA	Received data

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
7:0	RXFIFO_RX_DATA	RXFIFO_RX_DATA_DEFAULT		0x0*

5.9.0.69 RF_IQFIFO

Bit Field	Read/Write	Field Name	Description
7:0	R	IQFIFO_IQ_DATA	IQ data for AoA or AoD

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
7:0	IQFIFO_IQ_DATA	IQFIFO_IQ_DATA_DEFAULT		0x0*

5.9.0.70 RF_REG45

Bit Field	Read/Write	Field Name	Description
25:16	R	RSSI_AVG_RSSI_AVG	Filtered RSSI value
8:0	R	RXFIFO_COUNT_RX_COUNT	Number of bytes in the Rx FIFO

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
25:16	RSSI_AVG_RSSI_AVG	RSSI_AVG_RSSI_AVG_DEFAULT		0x0*
8:0	RXFIFO_COUNT_RX_ COUNT	RXFIFO_COUNT_RX_COUNT_ DEFAULT		0x0*

5.9.0.71 RF_DESER_STATUS

Bit Field	Read/Write	Field Name	Description
7	R	DESER_STATUS_SIGNAL_ RECEIVING	Deserializer enabling
6	R	DESER_STATUS_SYNC_ DETECTED	Sync word detection
5	R	DESER_STATUS_WAIT_SYNC	Deserializer waiting for the sync word
4	R	DESER_STATUS_IS_ADDRESS_ BR	Received address
3	R	DESER_STATUS_PKT_LEN_ERR	Packet length
2	R	DESER_STATUS_ADDRESS_ERR	Address error
1	R	DESER_STATUS_CRC_ERR	CRC error
0	R	DESER_STATUS_DESER_ FINISH	Deserializer status

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
7	DESER_STATUS_ SIGNAL_RECEIVING	DESER_STATUS_SIGNAL_ RECEIVING_DISABLE	Deserializer is disabled	0x0*
		DESER_STATUS_SIGNAL_ RECEIVING_ENABLE	Deserializer is enabled	0x1
6	DESER_STATUS_ SYNC_DETECTED	DESER_STATUS_SYNC_NOT_ DETECTED	Sync word (pattern) not detected	0x0*
		DESER_STATUS_SYNC_DETECTED	Sync word (pattern) detected	0x1
5	DESER_STATUS_WAIT_ SYNC	DESER_STATUS_WAIT_SYNC_ NOT_WAITING	Deserializer is not waiting the sync word	0x0*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		DESER_STATUS_WAIT_SYNC_WAITING	Deserializer is waiting the sync word	0x1
4	DESER_STATUS_IS_ADDRESS_BR	DESER_STATUS_IS_ADDRESS_BR_DISABLE	The received address is not the broadcast address	0x0*
		DESER_STATUS_IS_ADDRESS_BR_ENABLE	The received address is the broadcast address	0x1
3	DESER_STATUS_PKT_LEN_ERR	DESER_STATUS_PKT_LEN_ERR_NO_ERROR	The packet length is shorter than the maximum acceptable packet length	0x0*
		DESER_STATUS_PKT_LEN_ERR_ERROR	The packet length is longer than the maximum acceptable packet length	0x1
2	DESER_STATUS_ADDRESS_ERR	DESER_STATUS_ADDRESS_ERR_NO_ERROR	No address error detected	0x0*
		DESER_STATUS_ADDRESS_ERR_ERROR	Address error detected	0x1
1	DESER_STATUS_CRC_ERR	DESER_STATUS_CRC_ERR_NO_ERROR	No CRC error detected	0x0*
		DESER_STATUS_CRC_ERR_ERROR	CRC error detected	0x1
0	DESER_STATUS_DESER_FINISH	DESER_STATUS_DESER_FINISH_BUSY	Deserializer has not yet finished	0x0*
		DESER_STATUS_DESER_FINISH_IDLE	Deserializer has finished	0x1

5.9.0.72 RF_BLE_AEC_CCM

Bit Field	Read/Write	Field Name	Description
2	R	BLE_AES_CCM_BLE_AES_MIC_OK	AES CCM MIC error
1	R	BLE_AES_CCM_BLE_AES_DONE_RX	AES CCM packet decoding
0	R	BLE_AES_CCM_BLE_AES_DONE_TX	AES CCM packet encoding

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
2	BLE_AES_CCM_BLE_AES_MIC_OK	BLE_AES_CCM_BLE_AES_MIC_OK_ERROR	MIC has been received with errors	0x0*
		BLE_AES_CCM_BLE_AES_MIC_OK_NO_ERROR	MIC has been received without errors	0x1
1	BLE_AES_CCM_BLE_AES_DONE_RX	BLE_AES_CCM_BLE_AES_DONE_RX_BUSY	BLE AES CCM algorithm has not yet finished the packet decoding	0x0*
		BLE_AES_CCM_BLE_AES_DONE_RX_IDLE	BLE AES CCM algorithm has finished the packet decoding	0x1
0	BLE_AES_CCM_BLE_AES_DONE_TX	BLE_AES_CCM_BLE_AES_DONE_TX_BUSY	BLE AES CCM algorithm has not yet finished the packet encoding	0x0*
		BLE_AES_CCM_BLE_AES_DONE_TX_IDLE	BLE AES CCM algorithm has finished the packet encoding	0x1

5.9.0.73 RF_IRQ_STATUS

Bit Field	Read/Write	Field Name	Description
5	R	IRQ_STATUS_FLAG_RXFIFO	IRQ RXFIFO status
4	R	IRQ_STATUS_FLAG_TXFIFO	IRQ TXFIFO status
3	R	IRQ_STATUS_FLAG_SYNC	IRQ SYNC status
2	R	IRQ_STATUS_FLAG_RECEIVED	IRQ RECEIVED status
1	R	IRQ_STATUS_FLAG_RXSTOP	IRQ RXSTOP status
0	R	IRQ_STATUS_FLAG_TX	IRQ Tx status

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
5	IRQ_STATUS_FLAG_RXFIFO	IRQ_STATUS_FLAG_RXFIFO_IDLE	IRQ RXFIFO is not active	0x0*
		IRQ_STATUS_FLAG_RXFIFO_ACTIVE	IRQ RXFIFO is active	0x1
4	IRQ_STATUS_FLAG_TXFIFO	IRQ_STATUS_FLAG_TXFIFO_IDLE	IRQ TXFIFO is not active	0x0*
		IRQ_STATUS_FLAG_TXFIFO_ACTIVE	IRQ TXFIFO is active	0x1
3	IRQ_STATUS_FLAG_SYNC	IRQ_STATUS_FLAG_SYNC_IDLE	IRQ SYNC is not active	0x0*
		IRQ_STATUS_FLAG_SYNC_	IRQ SYNC is active	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		ACTIVE		
2	IRQ_STATUS_FLAG_RECEIVED	IRQ_STATUS_FLAG_RECEIVED_IDLE	IRQ RECEIVED is not active	0x0*
		IRQ_STATUS_FLAG_RECEIVED_ACTIVE	IRQ RECEIVED is active	0x1
1	IRQ_STATUS_FLAG_RXSTOP	IRQ_STATUS_FLAG_RXSTOP_IDLE	IRQ RXSTOP is not active	0x0*
		IRQ_STATUS_FLAG_RXSTOP_ACTIVE	IRQ RXSTOP is active	0x1
0	IRQ_STATUS_FLAG_TX	IRQ_STATUS_FLAG_TX_IDLE	IRQ TX is not active	0x0*
		IRQ_STATUS_FLAG_TX_ACTIVE	IRQ TX is active	0x1

5.9.0.74 RF_RSSI_MIN_MAX

Bit Field	Read/Write	Field Name	Description
25:16	R	RSSI_MAX_RSSI_MAX	Maximum RSSI value over a filtering period
9:0	R	RSSI_MIN_RSSI_MIN	Minimum RSSI value over a filtering period

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
25:16	RSSI_MAX_RSSI_MAX	RSSI_MAX_RSSI_MAX_DEFAULT		0x0*
9:0	RSSI_MIN_RSSI_MIN	RSSI_MIN_RSSI_MIN_DEFAULT		0x0*

5.9.0.75 RF_REG4A

Bit Field	Read/Write	Field Name	Description
30:28	R	RX_ATT_LEVEL_RX_ATT_LEVEL_PKT_LVL	Rx attenuation level (AGC level) during the packet reception
26:24	R	RX_ATT_LEVEL_RX_ATT_LEVEL	Rx attenuation level (AGC level)
23:16	R	DR_ERR_IND_DR_ERR_IND	Data-rate error indicator
9:0	R	RSSI_PKT_RSSI_PKT	Filtered RSSI value sampled during the packet reception

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
30:28	RX_ATT_LEVEL_RX_ATT_LEVEL_PKT_LVL	RX_ATT_LEVEL_RX_ATT_LEVEL_PKT_LVL_DEFAULT		0x0*
26:24	RX_ATT_LEVEL_RX_ATT_LEVEL	RX_ATT_LEVEL_RX_ATT_LEVEL_DEFAULT		0x0*
23:16	DR_ERR_IND_DR_ERR_IND	DR_ERR_IND_DR_ERR_IND_DEFAULT		0x0*
9:0	RSSI_PKT_RSSI_PKT	RSSI_PKT_RSSI_PKT_DEFAULT		0x0*

5.9.0.76 RF_FEI

Bit Field	Read/Write	Field Name	Description
31:16	R	FEI_PKT_FEI_PKT	Frequency error indicator sampled during the packet reception
15:0	R	FEI_FEI_OUT	Frequency error indicator

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:16	FEI_PKT_FEI_PKT	FEI_PKT_FEI_PKT_DEFAULT		0x0*
15:0	FEI_FEI_OUT	FEI_FEI_OUT_DEFAULT		0x0*

5.9.0.77 RF_REG4C

Bit Field	Read/Write	Field Name	Description
31:24	R	LINK_QUAL_PKT_LINK_QUALITY_PKT	Link quality indicator sampled during the packet reception
23:16	R	LINK_QUAL_LINK_QUALITY	Instantaneous link quality indicator
15:0	R	FEI_AFC_FEI_AFC	Frequency error indicator sampled during the AFC

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:24	LINK_QUAL_PKT_LINK_QUALITY_PKT	LINK_QUAL_PKT_LINK_QUALITY_PKT_DEFAULT		0x0*
23:16	LINK_QUAL_LINK_QUALITY	LINK_QUAL_LINK_QUALITY_DEFAULT		0x0*
15:0	FEI_AFC_FEI_AFC	FEI_AFC_FEI_AFC_DEFAULT		0x0*

5.9.0.78 RF_ANALOG_INFO

Bit Field	Read/Write	Field Name	Description
25:24	W	BLR_READOUT_BLR_RATE	Bluetooth LE long range rate indicator
22:20	R	PEAK_DET_VAL_PEAK_DET_FILT	Distance from the subband center (only available with the FLL method)
18:16	R	PEAK_DET_VAL_PEAK_DET_RAW	Distance from the subband center (only available with the FLL method)
15	R	ANALOG_INFO_POR_VDDA	VDDA LDO disable status
14	R	ANALOG_INFO_PLL_UNLOCK	PLL unlock status
13	R	ANALOG_INFO_XTAL_FINISH	XTAL algorithm status
12	R	ANALOG_INFO_DLL_LOCKED	DLL lock status
11	R	ANALOG_INFO_CLK_DIG_READY	Ready signal of the digital clock
10	R	ANALOG_INFO_CLK_PLL_READY	PLL clock status
9:8	R	ANALOG_INFO_SUBBAND	Status of the subband comparator Hi
7:0	R	SUBBAND_ERR_SB_FLL_ERR	Distance from the subband center (only available with the FLL method)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
25:24	BLR_READOUT_BLR_RATE	BLR_READOUT_BLR_RATE_DEFAULT	No action	0x0
22:20	PEAK_DET_VAL_PEAK_DET_FILT	PEAK_DET_VAL_PEAK_DET_FILT_DEFAULT		0x0*
18:16	PEAK_DET_VAL_PEAK_DET_RAW	PEAK_DET_VAL_PEAK_DET_RAW_DEFAULT		0x0*
15	ANALOG_INFO_POR_VDDA	ANALOG_INFO_POR_VDDA_ENABLE	VDDA LDO rnabled	0x0*
		ANALOG_INFO_POR_VDDA_DISABLE	VDDA LDO disabled	0x1
14	ANALOG_INFO_PLL_UNLOCK	ANALOG_INFO_PLL_LOCKED	PLL is locked	0x0*
		ANALOG_INFO_PLL_UNLOCKED	PLL is unlocked	0x1
13	ANALOG_INFO_XTAL_FINISH	ANALOG_INFO_XTAL_TRIM_RUNNING	Xtal algorithm has not yet finished	0x0*
		ANALOG_INFO_XTAL_TRIM_	Xtal algorithm has finished	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		FINISHED		
12	ANALOG_INFO_DLL_LOCKED	ANALOG_INFO_DLL_UNLOCKED	DLL is unlocked	0x0*
		ANALOG_INFO_DLL_LOCKED	DLL is locked	0x1
11	ANALOG_INFO_CLK_DIG_READY	ANALOG_INFO_CLK_DIG_NOT_READY	Digital clock not ready	0x0*
		ANALOG_INFO_CLK_DIG_READY	Digital clock ready	0x1
10	ANALOG_INFO_CLK_PLL_READY	ANALOG_INFO_CLK_PLL_NOT_READY	PLL clock not ready	0x0*
		ANALOG_INFO_CLK_PLL_READY	PLL clock ready	0x1
9:8	ANALOG_INFO_SUBBAND	ANALOG_INFO_SUBBAND_DEFAULT		0x0*
7:0	SUBBAND_ERR_SB_FLL_ERR	SUBBAND_ERR_SB_FLL_ERR_DEFAULT		0x0*

5.9.0.79 RF_SAMPLE_RSSI

Bit Field	Read/Write	Field Name	Description
0	W	SAMPLE_RSSI	Sample the thermometric RSSI

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
0	SAMPLE_RSSI	SAMPLE_RSSI_SAMPLE_RSSI	Sample the thermometric RSSI	0x1

5.9.0.80 RF_RSSI_THERM

Bit Field	Read/Write	Field Name	Description
29:0	R	RSSI_THERM	Thermometric value of the RSSI

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
29:0	RSSI_THERM	RSSI_THERM_DEFAULT		0x0*

5.9.0.81 RF_LUT_ANTENNA_ARRAY_1

Bit Field	Read/Write	Field Name	Description
31:28	RW	LUT_ANTENNA_ARRAY_1_ ANTENNA_7	Antenna 7 specification
27:24	RW	LUT_ANTENNA_ARRAY_1_ ANTENNA_6	Antenna 6 specification
23:20	RW	LUT_ANTENNA_ARRAY_1_ ANTENNA_5	Antenna 5 specification
19:16	RW	LUT_ANTENNA_ARRAY_1_ ANTENNA_4	Antenna 4 specification
15:12	RW	LUT_ANTENNA_ARRAY_1_ ANTENNA_3	Antenna 3 specification
11:8	RW	LUT_ANTENNA_ARRAY_1_ ANTENNA_2	Antenna 2 specification
7:4	RW	LUT_ANTENNA_ARRAY_1_ ANTENNA_1	Antenna 1 specification
3:0	RW	LUT_ANTENNA_ARRAY_1_ ANTENNA_0	Antenna 0 specification

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:28	LUT_ANTENNA_ ARRAY_1_ANTENNA_7	LUT_ANTENNA_ARRAY_1_ ANTENNA_7_DEFAULT		0x0*
27:24	LUT_ANTENNA_ ARRAY_1_ANTENNA_6	LUT_ANTENNA_ARRAY_1_ ANTENNA_6_DEFAULT		0x0*
23:20	LUT_ANTENNA_ ARRAY_1_ANTENNA_5	LUT_ANTENNA_ARRAY_1_ ANTENNA_5_DEFAULT		0x0*
19:16	LUT_ANTENNA_ ARRAY_1_ANTENNA_4	LUT_ANTENNA_ARRAY_1_ ANTENNA_4_DEFAULT		0x0*
15:12	LUT_ANTENNA_ ARRAY_1_ANTENNA_3	LUT_ANTENNA_ARRAY_1_ ANTENNA_3_DEFAULT		0x3*
11:8	LUT_ANTENNA_ ARRAY_1_ANTENNA_2	LUT_ANTENNA_ARRAY_1_ ANTENNA_2_DEFAULT		0x2*
7:4	LUT_ANTENNA_ ARRAY_1_ANTENNA_1	LUT_ANTENNA_ARRAY_1_ ANTENNA_1_DEFAULT		0x1*
3:0	LUT_ANTENNA_ ARRAY_1_ANTENNA_0	LUT_ANTENNA_ARRAY_1_ ANTENNA_0_DEFAULT		0x0*

5.9.0.82 RF_LUT_ANTENNA_ARRAY_2

Bit Field	Read/Write	Field Name	Description
31:28	RW	LUT_ANTENNA_ARRAY_2_ ANTENNA_15	Antenna 15 specification
27:24	RW	LUT_ANTENNA_ARRAY_2_ ANTENNA_14	Antenna 14 specification
23:20	RW	LUT_ANTENNA_ARRAY_2_ ANTENNA_13	Antenna 13 specification
19:16	RW	LUT_ANTENNA_ARRAY_2_ ANTENNA_12	Antenna 12 specification
15:12	RW	LUT_ANTENNA_ARRAY_2_ ANTENNA_11	Antenna 11 specification
11:8	RW	LUT_ANTENNA_ARRAY_2_ ANTENNA_10	Antenna 10 specification
7:4	RW	LUT_ANTENNA_ARRAY_2_ ANTENNA_9	Antenna 9 specification
3:0	RW	LUT_ANTENNA_ARRAY_2_ ANTENNA_8	Antenna 8 specification

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:28	LUT_ANTENNA_ ARRAY_2_ANTENNA_15	LUT_ANTENNA_ARRAY_2_ ANTENNA_15_DEFAULT		0x0*
27:24	LUT_ANTENNA_ ARRAY_2_ANTENNA_14	LUT_ANTENNA_ARRAY_2_ ANTENNA_14_DEFAULT		0x0*
23:20	LUT_ANTENNA_ ARRAY_2_ANTENNA_13	LUT_ANTENNA_ARRAY_2_ ANTENNA_13_DEFAULT		0x0*
19:16	LUT_ANTENNA_ ARRAY_2_ANTENNA_12	LUT_ANTENNA_ARRAY_2_ ANTENNA_12_DEFAULT		0x0*
15:12	LUT_ANTENNA_ ARRAY_2_ANTENNA_11	LUT_ANTENNA_ARRAY_2_ ANTENNA_11_DEFAULT		0x0*
11:8	LUT_ANTENNA_ ARRAY_2_ANTENNA_10	LUT_ANTENNA_ARRAY_2_ ANTENNA_10_DEFAULT		0x0*
7:4	LUT_ANTENNA_ ARRAY_2_ANTENNA_9	LUT_ANTENNA_ARRAY_2_ ANTENNA_9_DEFAULT		0x0*
3:0	LUT_ANTENNA_ ARRAY_2_ANTENNA_8	LUT_ANTENNA_ARRAY_2_ ANTENNA_8_DEFAULT		0x0*

5.9.0.83 RF_LUT_ANTENNA_ARRAY_3

Bit Field	Read/Write	Field Name	Description
31:28	RW	LUT_ANTENNA_ARRAY_3_ ANTENNA_23	Antenna 23 specification
27:24	RW	LUT_ANTENNA_ARRAY_3_ ANTENNA_22	Antenna 22 specification
23:20	RW	LUT_ANTENNA_ARRAY_3_ ANTENNA_21	Antenna 21 specification
19:16	RW	LUT_ANTENNA_ARRAY_3_ ANTENNA_20	Antenna 20 specification
15:12	RW	LUT_ANTENNA_ARRAY_3_ ANTENNA_19	Antenna 19 specification
11:8	RW	LUT_ANTENNA_ARRAY_3_ ANTENNA_18	Antenna 18 specification
7:4	RW	LUT_ANTENNA_ARRAY_3_ ANTENNA_17	Antenna 17 specification
3:0	RW	LUT_ANTENNA_ARRAY_3_ ANTENNA_16	Antenna 16 specification

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:28	LUT_ANTENNA_ ARRAY_3_ANTENNA_23	LUT_ANTENNA_ARRAY_3_ ANTENNA_23_DEFAULT		0x0*
27:24	LUT_ANTENNA_ ARRAY_3_ANTENNA_22	LUT_ANTENNA_ARRAY_3_ ANTENNA_22_DEFAULT		0x0*
23:20	LUT_ANTENNA_ ARRAY_3_ANTENNA_21	LUT_ANTENNA_ARRAY_3_ ANTENNA_21_DEFAULT		0x0*
19:16	LUT_ANTENNA_ ARRAY_3_ANTENNA_20	LUT_ANTENNA_ARRAY_3_ ANTENNA_20_DEFAULT		0x0*
15:12	LUT_ANTENNA_ ARRAY_3_ANTENNA_19	LUT_ANTENNA_ARRAY_3_ ANTENNA_19_DEFAULT		0x0*
11:8	LUT_ANTENNA_ ARRAY_3_ANTENNA_18	LUT_ANTENNA_ARRAY_3_ ANTENNA_18_DEFAULT		0x0*
7:4	LUT_ANTENNA_ ARRAY_3_ANTENNA_17	LUT_ANTENNA_ARRAY_3_ ANTENNA_17_DEFAULT		0x0*
3:0	LUT_ANTENNA_ ARRAY_3_ANTENNA_16	LUT_ANTENNA_ARRAY_3_ ANTENNA_16_DEFAULT		0x0*

5.9.0.84 RF_LUT_ANTENNA_ARRAY_4

Bit Field	Read/Write	Field Name	Description
31:28	RW	LUT_ANTENNA_ARRAY_4_ ANTENNA_31	Antenna 31 specification
27:24	RW	LUT_ANTENNA_ARRAY_4_ ANTENNA_30	Antenna 30 specification
23:20	RW	LUT_ANTENNA_ARRAY_4_ ANTENNA_29	Antenna 29 specification
19:16	RW	LUT_ANTENNA_ARRAY_4_ ANTENNA_28	Antenna 28 specification
15:12	RW	LUT_ANTENNA_ARRAY_4_ ANTENNA_27	Antenna 27 specification
11:8	RW	LUT_ANTENNA_ARRAY_4_ ANTENNA_26	Antenna 26 specification
7:4	RW	LUT_ANTENNA_ARRAY_4_ ANTENNA_25	Antenna 25 specification
3:0	RW	LUT_ANTENNA_ARRAY_4_ ANTENNA_24	Antenna 24 specification

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:28	LUT_ANTENNA_ ARRAY_4_ANTENNA_31	LUT_ANTENNA_ARRAY_4_ ANTENNA_31_DEFAULT		0x0*
27:24	LUT_ANTENNA_ ARRAY_4_ANTENNA_30	LUT_ANTENNA_ARRAY_4_ ANTENNA_30_DEFAULT		0x0*
23:20	LUT_ANTENNA_ ARRAY_4_ANTENNA_29	LUT_ANTENNA_ARRAY_4_ ANTENNA_29_DEFAULT		0x0*
19:16	LUT_ANTENNA_ ARRAY_4_ANTENNA_28	LUT_ANTENNA_ARRAY_4_ ANTENNA_28_DEFAULT		0x0*
15:12	LUT_ANTENNA_ ARRAY_4_ANTENNA_27	LUT_ANTENNA_ARRAY_4_ ANTENNA_27_DEFAULT		0x0*
11:8	LUT_ANTENNA_ ARRAY_4_ANTENNA_26	LUT_ANTENNA_ARRAY_4_ ANTENNA_26_DEFAULT		0x0*
7:4	LUT_ANTENNA_ ARRAY_4_ANTENNA_25	LUT_ANTENNA_ARRAY_4_ ANTENNA_25_DEFAULT		0x0*
3:0	LUT_ANTENNA_ ARRAY_4_ANTENNA_24	LUT_ANTENNA_ARRAY_4_ ANTENNA_24_DEFAULT		0x0*

5.9.0.85 RF_REG50

Bit Field	Read/Write	Field Name	Description
27	R	FEATURES_HAS_BLE_AES	Bluetooth AES block availability
26	R	FEATURES_HAS_BLE_DF_AOA_AOD	Bluetooth Direction Finding AoA/AoD feature availability
25	R	FEATURES_HAS_BLE_LONG_RANGE	Bluetooth long range feature availability
24	R	FEATURES_FEATURES_AVAILABLE	Features availability
23:16	W	BLR_PKT_LEN_BLR_PKT_LEN	Packet length of the BLR packet
15:8	W	PROT_TIMER_PT_CMD	Protocol timer command
0	W	COMMANDS_START_SUBBAND	Subband selection algorithm

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
27	FEATURES_HAS_BLE_AES	FEATURES_HAS_NOT_BLE_AES	AES block is not available	0x0*
		FEATURES_HAS_BLE_AES	AES block is available	0x1
26	FEATURES_HAS_BLE_DF_AOA_AOD	FEATURES_HAS_NOT_BLE_DF_AOA_AOD	AoA/AoD is not available	0x0
		FEATURES_HAS_BLE_DF_AOA_AOD	AoA/AoD is available	0x1*
25	FEATURES_HAS_BLE_LONG_RANGE	FEATURES_HAS_NOT_BLE_LONG_RANGE	Long range is not available	0x0
		FEATURES_HAS_BLE_LONG_RANGE	Long range is available	0x1*
24	FEATURES_FEATURES_AVAILABLE	FEATURES_FEATURES_NOT_AVAILABLE	Features are not available	0x0
		FEATURES_FEATURES_AVAILABLE	Features are available	0x1*
23:16	BLR_PKT_LEN_BLR_PKT_LEN	BLR_PKT_LEN_BLR_PKT_LEN_DEFAULT		0x0
15:8	PROT_TIMER_PT_CMD	PROT_TIMER_PT_CMD_DEFAULT		0x0
0	COMMANDS_START_SUBBAND	COMMANDS_START_SUBBAND_IDLE	No action	0x0
		COMMANDS_START_SUBBAND_START	Start the subband selection algorithm	0x1

5.9.0.86 RF_REG51

Bit Field	Read/Write	Field Name	Description
31:24	RW	FSM_MODE_RM_TX	Remapped register of FSM_MODE
23:16	RW	PA_PWR_RM	Remapped register of PA_PWR
15:8	RW	CHANNEL_RM_TX	Remapped register of CHANNEL
7:0	RW	RATE_TX	Remapped register of BANK

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:24	FSM_MODE_RM_TX	FSM_MODE_RM_TX_DEFAULT	The remapped registers are accessible over SPI only	0x0*
23:16	PA_PWR_RM	PA_PWR_RM_DEFAULT	The remapped registers are accessible over SPI only	0x0*
15:8	CHANNEL_RM_TX	CHANNEL_RM_TX_DEFAULT	The remapped registers are accessible over SPI only	0x0*
7:0	RATE_TX	RATE_TX_DEFAULT	The remapped registers are accessible over SPI only	0x0*

5.9.0.87 RF_REG52

Bit Field	Read/Write	Field Name	Description
31:24	RW	ACCESS_ADDRESS	Remapped register of PATTERN
23:16	RW	FSM_MODE_RM_RX	Remapped register of FSM_MODE
15:8	RW	CHANNEL_RM_RX	Remapped register of CHANNEL
7:0	RW	RATE_RX	Remapped register of BANK

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:24	ACCESS_ADDRESS	ACCESS_ADDRESS_DEFAULT	The remapped registers are accessible over SPI only	0x0*
23:16	FSM_MODE_RM_RX	FSM_MODE_RM_RX_DEFAULT	The remapped registers are accessible over SPI only	0x0*
15:8	CHANNEL_RM_RX	CHANNEL_RM_RX_DEFAULT	The remapped registers are accessible over SPI only	0x0*
7:0	RATE_RX	RATE_RX_DEFAULT	The remapped registers are accessible over SPI only	0x0*

5.9.0.88 RF_REG53

Bit Field	Read/Write	Field Name	Description
23:16	R	RSSI_MAX_RM	Remapped register of RSSI_MAX
15:8	R	RSSI_MIN_RM	Remapped register of RSSI_MIN
7:0	R	RSSI_AVG_RM	Remapped register of RSSI_AVG

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
23:16	RSSI_MAX_RM	RSSI_MAX_RM_DEFAULT	The remapped registers are accessible over SPI only	0x0*
15:8	RSSI_MIN_RM	RSSI_MIN_RM_DEFAULT	The remapped registers are accessible over SPI only	0x0*
7:0	RSSI_AVG_RM	RSSI_AVG_RM_DEFAULT	The remapped registers are accessible over SPI only	0x0*

5.9.0.89 RF_REG54

Bit Field	Read/Write	Field Name	Description
7:0	W	BLR_PACKET_LEN	Remapped register of BLR_PACKET_LEN

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
7:0	BLR_PACKET_LEN	BLR_PACKET_LEN_DEFAULT	The remapped registers are accessible over SPI only	0x0

5.9.0.90 RF_REG55

Bit Field	Read/Write	Field Name	Description
7:0	R	ITRX_FEATURES	Remapped register of ITRX_FEATURES

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
7:0	ITRX_FEATURES	ITRX_FEATURES_DEFAULT	The remapped registers are accessible over SPI only	0x0*

5.9.0.91 RF_REG56

Bit Field	Read/Write	Field Name	Description
31:24	R	CHIP_ID_CHIP_ID	Version of the chip
23:16	R	MD5_REGS_MD5_REGS	MD5 calculated on the register map file
15:8	W	SCAN_2_SCAN_2_PASSWORD	SCAN 2 key
7:0	W	SCAN_1_SCAN_1_PASSWORD	SCAN 1 key

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:24	CHIP_ID_CHIP_ID	CHIP_ID_CHIP_ID_DEFAULT		0x30*
23:16	MD5_REGS_MD5_ REGS	MD5_REGS_MD5_REGS_DEFAULT		0x0*
15:8	SCAN_2_SCAN_2_ PASSWORD	SCAN_2_SCAN_2_PASSWORD_ DEFAULT		0x0
7:0	SCAN_1_SCAN_1_ PASSWORD	SCAN_1_SCAN_1_PASSWORD_ DEFAULT		0x0

CHAPTER 6

Bluetooth Low Energy Baseband Controller

RSL15 has dedicated hardware designed to handled Bluetooth Low Energy activities and events. From one side, the hardware communicates with the RF front-end control interface through an internal SPI interface (for RX or TX programming and radio FSM control purposes), along with internal signals connected between Bluetooth Low Energy hardware and the RF front-end for the TX/RX data/clock and IQ data. From the other side, the hardware communicates to the Bluetooth Low Energy stack software. This communication is executed in firmware through shared memory, Bluetooth Low Energy interrupts, and Bluetooth Low Energy hardware registers.

6.1 Overview

In general, RSL15 handles the Bluetooth Low Energy protocol through a Bluetooth Low Energy stack, which is in charge of the upper protocol layers and Bluetooth Low Energy hardware (HW). User applications are not responsible for Bluetooth Low Energy hardware. Therefore, this topic presents a high-level overview of the design and how it works.

NOTE: In this documentation, *BB* refers to Bluetooth Low Energy hardware. While there is a type of baseband processing in the RFFE, take care not to confuse it with the BB controller terminology used here. Throughout this documentation, you can assume that *BB controller* or *baseband controller* always means the Bluetooth Low Energy hardware, unless it is clearly stated that the RFFE baseband is being referenced.

Since the Bluetooth Low Energy HW (BB controller or BB) acts like an individual hardware accelerator, the RSL15 SoC contains an interface block, the baseband controller interface (BBIF). This block is designed to allow the Arm Cortex-M33 processor to configure the BB controller or to obtain the required information and status from the BB block.

In the "Bluetooth Low Energy Baseband Controller Block Diagram" figure (Figure 19), the Bluetooth Low Energy HW block diagram and its interface to the other blocks of RSL15 are illustrated.

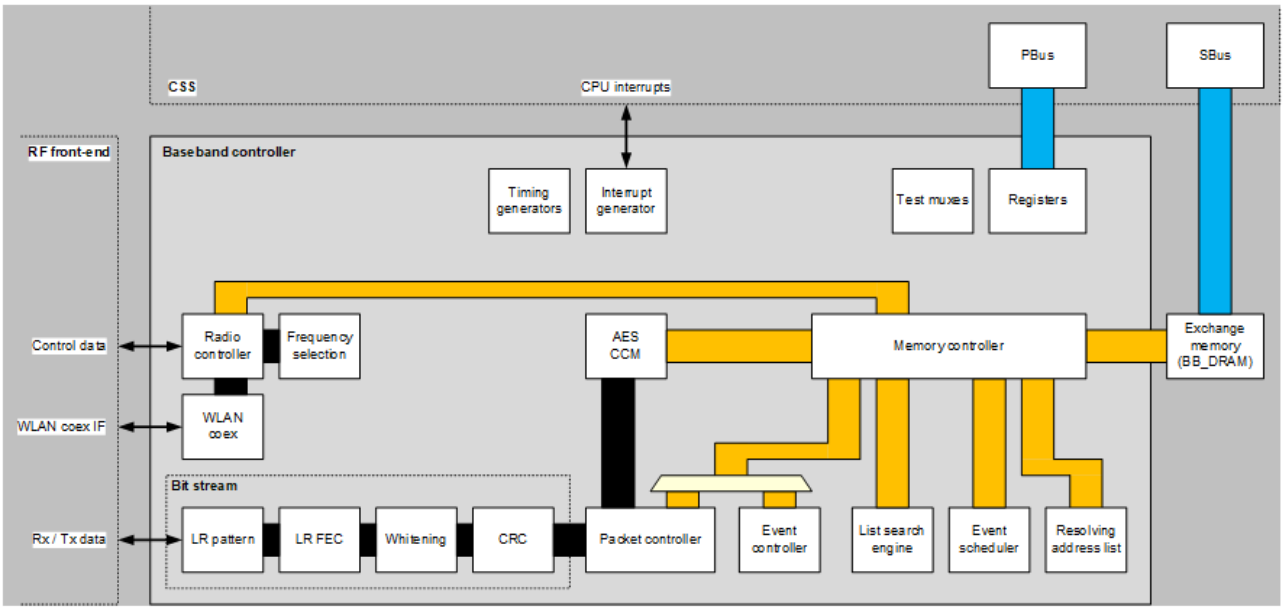


Figure 19. Bluetooth Low Energy Baseband Controller Block Diagram

The implementation supports the Bluetooth Low Energy features as follows:

- Bluetooth 5.2 Specifications compliant
- All packet types (Broadcasting / Advertising / Data / Control / Long Range)
- Advertising Extension
- Encryption/Decryption (AES-CCM)
- Bit stream processing (CRC, Whitening)
- Frequency hopping calculation (scheme #1 and scheme #2)
- FDMA/TDMA/events formatting and synchronization
- All device modes support (Broadcaster, Central, Observer, Peripheral)

The software stack schedules, configures, and manages the Bluetooth Low Energy events, while the BB controller processes the actual events, controls the RFFE, and handles real time bit stream processing.

Bluetooth Low Energy HW is connected to the exchange memory via the SBus. The RSL15 processor (Arm Cortex-M33 core) has access to this memory as well. To receive or send data, the BB_DRAM shared memory is used to share and communicate data between the Arm Cortex-M33 processor and the BB controller. The baseband controller communicates with:

1. The CPU through:
    a. The baseband controller interrupts
    b. An internal bus and the DMA that access the internal configuration registers
    c. The SBus and the DMA that access the exchange memory (BB_DRAM0 & BB_DRAM1) parallel to the baseband controller. This memory access is managed by an external arbiter.
2. The SBus and the DMA that access the exchange memory (BB_DRAM0 & BB_DRAM1) parallel to the baseband controller. This memory access is managed by an external arbiter.
3. The RFFE through the control and data signals listed in the "Interface Signals Between Baseband Controller and RFFE" table (Table 6). The control signals are transmitted over an internal SPI. The baseband controller is configured as the master, and the RF front-end as the slave. The RFFE modem sends/receives digital data to/from through the RX/TX data bus, as shown in the "Bluetooth Low Energy Baseband Controller Block Diagram" figure (Figure 19). The clock and reset signals are provided by the BBIF.

Table 6. Interface Signals Between Baseband Controller and RFFE

Signal	Baseband Controller	RFFE	Description
SPI_MISO	IN	OUT	SPI data slave to master
SPI_MOSI	OUT	IN	SPI data master to slave
SPI_CLK	OUT	IN	SPI data clock
SPI_CSN	OUT	IN	SPI slave device select
RX_DATA	IN	OUT	Rx data
RX_CLK	IN	OUT	Rx data clock
SYNC_P	IN	OUT	Access address detection
TX_DATA	OUT	IN	Tx data
TX_DATA_VALID	OUT	IN	Tx data_qualifier
CTE_MODE	OUT	IN	Enable CTE mode

Table 6. Interface Signals Between Baseband Controller and RFFE (Continued)

CTE_SAMPLE_P	OUT	IN	CTE sample pulse
IQ_DATA_P	IN	OUT	IQ data qualifier
IQ_DATA	IN	OUT	IQ data

When the RF front-end is isolated or powered down, the SPI MISO as well as RX_DATA and RX_CLK signals are forced to zero. When the baseband controller is isolated or powered down, the signals going to the RF front-end are forced to zero (except SPI CSN, which is forced to one). Disabling the RF front-end or the baseband controller stops and potentially corrupts any ongoing SPI transaction. For testing purpose, all the above signals can be accessed through the GPIOs, as illustrated in the "Interface Between the Baseband Controller and the RF Front-End" figure (Figure 20)

NOTE: The information in the "Interface Between the Baseband Controller and the RF Front-End" figure (Figure 20) is for debugging purposes only.

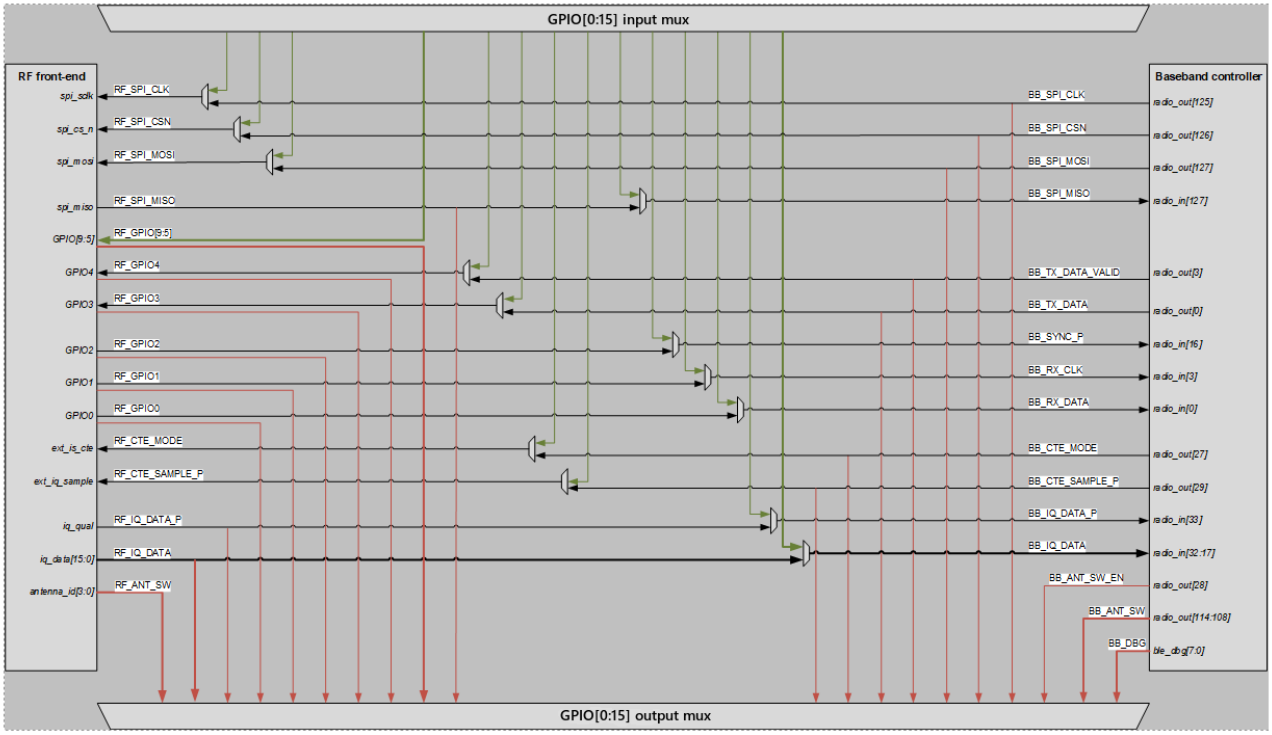


Figure 20. Interface Between the Baseband Controller and the RF Front-End

By default, the system is configured such that the BB controller is connected to the RFFE, and the Bluetooth Low Energy stack software configures the RFFE in Test Mode (see Chapter 5 "RF Front-End" on page 140) to be able to communicate with the BB controller. When the `BLE_Initialize()` function is called, the Bluetooth Low Energy stack resets the BB controller, configures the static registers of the RFFE, and prepares the BB controller registers. Finally, it releases the BB from reset and enables it.

From a network protocol layer point of view, the BB controller is in charge of the link layer that is being controlled, and is managed by the Bluetooth Low Energy software stack, which means that the link layer is implemented in the BB controller and the Bluetooth Low Energy stack together.

The BB controller is clocked from a scaled-down version of the system clock, known as the master clock. This baseband controller implements a software configurable internal clock divider for generating a 1 MHz reference clock from the master clock. Both are prescaled from the system clock. A description of how to configure the prescale settings is found in Chapter 6.3 "Baseband Controller Interface" on page 311.

6.2 System Resources

6.2.1 Exchange Memory

Memories BB_DRAM0 and BB_DRAM1 are allocated for use by the BB controller with the S-Bus, which allows it to share data with the Arm Cortex-M33 processor; this memory area is called the exchange memory. This section gives an overview of how the exchange memory works with the RSL15 Bluetooth Low Energy software and hardware.

The exchange memory is used for several purposes. The first 256-byte area of BB_DRAM0 is dedicated to event scheduling, to a maximum of 16 events, and is called the exchange table. The exchange table has 16 entries, which correspond to either successive advertising, scanning, or connection event anchor points. While the software prepares the future entries (i.e., anchor points), the hardware uses one entry to transmit/receive packets. Each exchange table entry points to a unique control structure that determines the event type to be performed. Each control structure points to a TX descriptor chained list, while a register determines the RX Descriptor location. Descriptors are chained lists in which the information/status of the current processed RX/TX packets are contained. Descriptors also contain the pointer to the next descriptor of the chained list. Each descriptor has an associated data pointer, giving the memory location of an associated data buffer. Each data buffer has a maximum length of 251 bytes.

Exchange table entries are fetched on a Bluetooth Low Energy Core software request, triggering a fetch of the control structure and descriptors at an optimized and defined period of time before the event starts.

The software prepares the control structures in advance, and the pointer in the exchange table is only set up when this operation is completed. The Bluetooth Low Energy software stack indicates to the baseband controller that a given entry has to be processed, using the `BB_ACTSCHCNTL` register. The BB controller accesses the relevant fields in the selected exchange table entry (i.e defined timestamp, priority levels, control structure pointer, etc.). The software can prepare the next entry and safely access the pointers. These mechanisms are handled by the Bluetooth Low Energy software stack. The "Exchange Table, Control Structure, Descriptors and Data Buffers" figure (Figure 21) provides an overview:

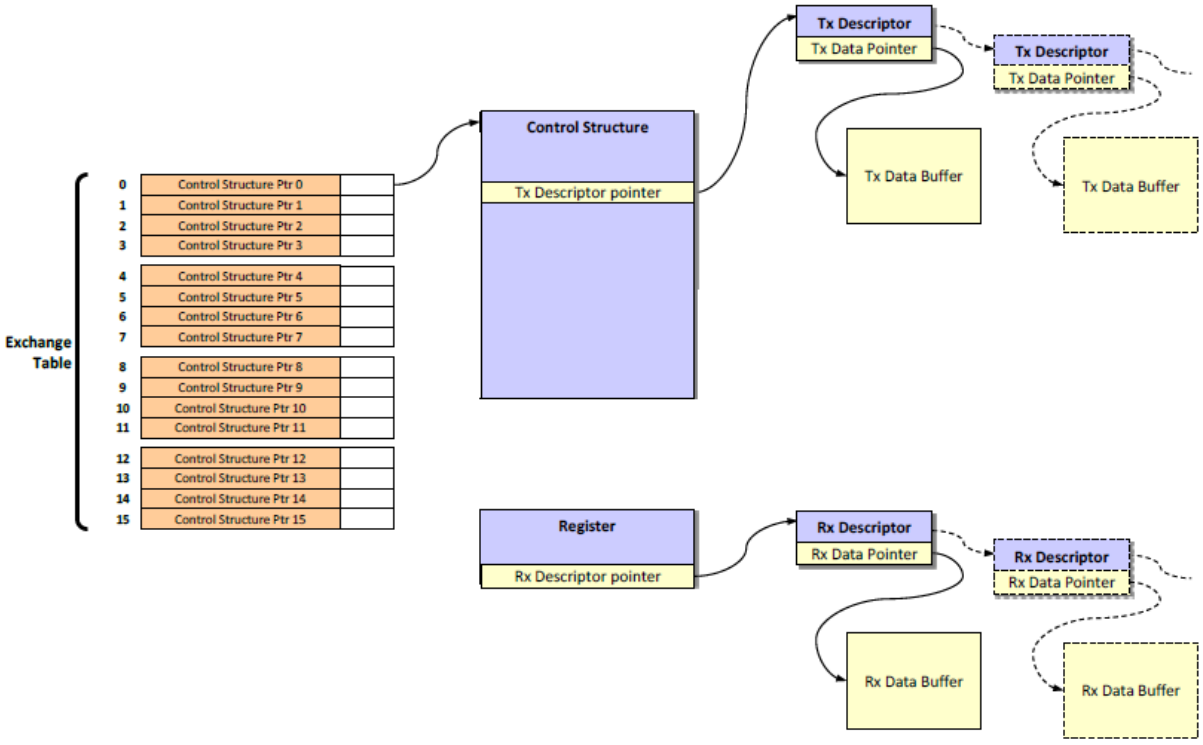


Figure 21. Exchange Table, Control Structure, Descriptors and Data Buffers

Events (advertising/connection/scanning) are programmed by the means of the control structure. Depending on the settings of the control structure format, the events can be composed differently. The Bluetooth Low Energy software stack is responsible for programming the advertising events and connection events. The BB controller activity scheduler generates an interrupt on each end of an event, as well as when an event is not processed: each exchange table is read and taken into account only after the current event in process is closed. The "Connection Event Programming and Exchange Table Pre-Fetch / Master Device Example" figure (Figure 22) and the "Connection Event Programming and Exchange Table Pre-Fetch / Slave Device Example" figure (Figure 23) show the exchange table pre-fetch for a master and a slave device, and how the BB controller behaves during each connection event:

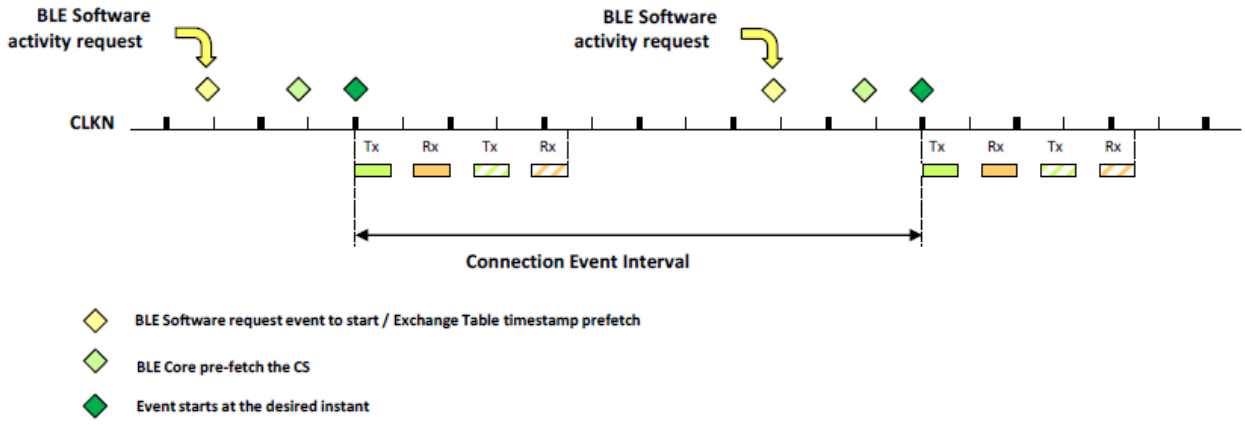


Figure 22. Connection Event Programming and Exchange Table Pre-Fetch / Master Device Example

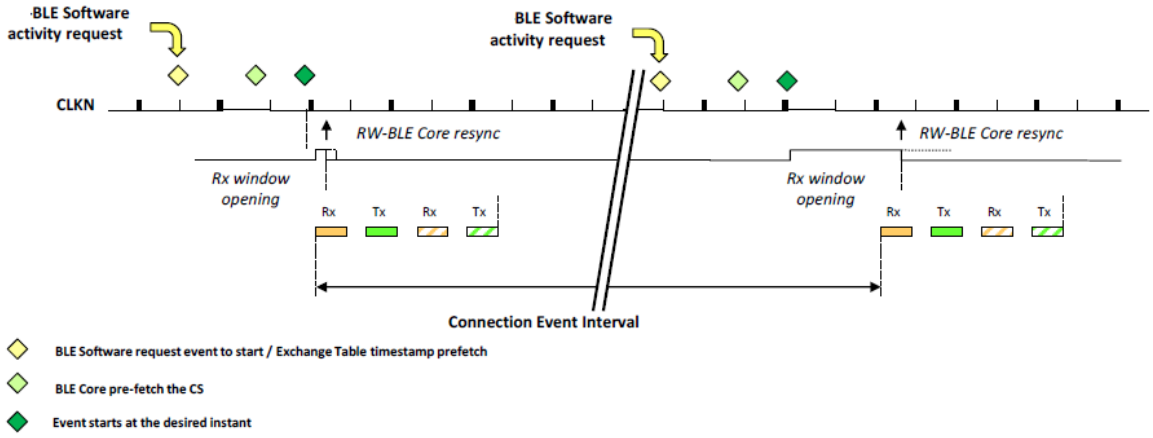


Figure 23. Connection Event Programming and Exchange Table Pre-Fetch / Slave Device Example

The other data is kept in exchange memory, including:

- A frequency table for VCO programming of the RFFE for each Bluetooth Low Energy channel
- A Bluetooth Low Energy white list
- A Bluetooth Low Energy resolving address list
- A periodic advertisement list
- An antenna pattern ID for AoA/AoD

6.2.2 Filtering Lists and Device Filtering

Device addresses and list organization are managed by the Bluetooth Low Energy software stack. A device for which IRKs have not been exchanged stands either in the white list, or in the periodic advertiser list: in this case, the device identity is used for device filtering (if used). As soon as IRKs are exchanged, a corresponding entry in the resolving address list is created. Each entry in the resolving address list indicates whether the device is in the white list or the periodic advertiser list. The "Device Filtering Lists Organization" figure (Figure 24) shows an example of the device filtering lists organization within the BB controller:

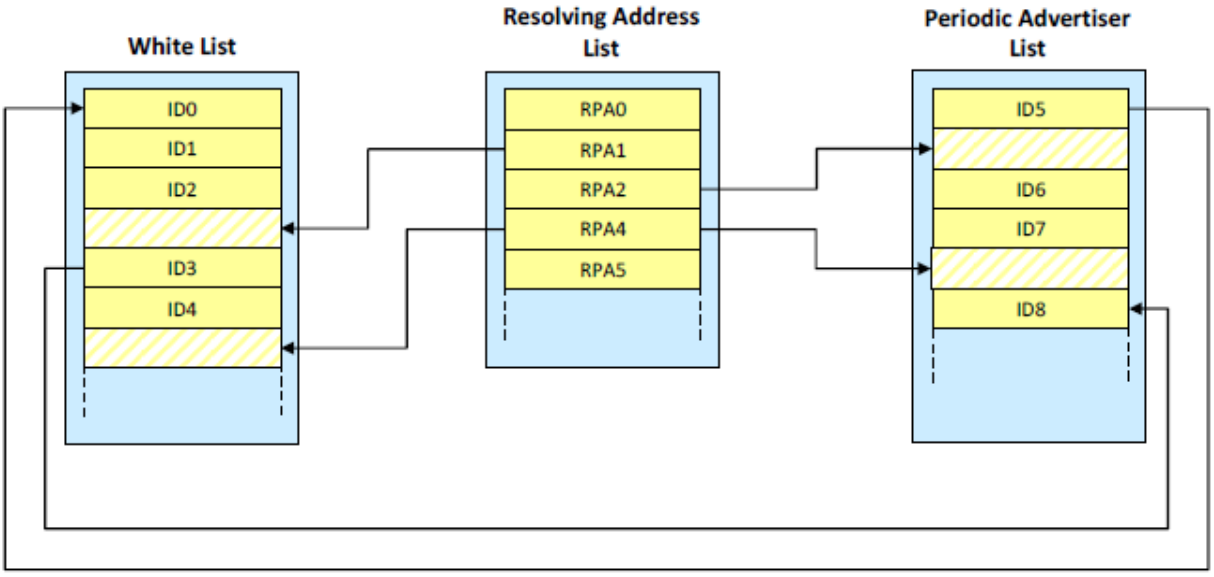


Figure 24. Device Filtering Lists Organization

Device filtering is based on the peer device's address. It is used to minimize the number of devices the system responds to during device discovery/connection. Filtering policies in between advertising state, scanning state, and initiating state are independent. Device filtering policy is determined by a parameter set by the Bluetooth Low Energy stack software in the control structure of the `CS-FILTER_POLICY` field, and applies differently according to the `CS-FORMAT` value. Additionally, the RPA resolution can be enabled when the device uses Enhanced Privacy Mode. If an RPA is not resolved, depending on device filtering policy, the device might not respond to the received packet.

6.2.3 AES-128 On Software Request

The BB controller uses an AES-128 hardware accelerator for Bluetooth Low Energy encryption and decryption of data, and for resolving the address list engine. This HW block is available to application software. When the Bluetooth Low Energy stack is not used, this HW block can be used safely by software; but if the Bluetooth Low Energy stack is used, we recommend that application software uses GAP API (refer to the GAP documentation in the *RSL15 Firmware Reference*), to avoid causing any issue for the Bluetooth Low Energy data being encrypted or decrypted by the BB controller.

AES-128 can be used by the software that stores the block to cipher, in the exchange memory (in the BB_DRAM0 and BB_DRAM1 address range) at the address specified in the `BB_AESPTR` register. The block size to cipher is always considered equal to 16 bytes. The software defines the input key by setting its value in the `BB_AESKEY*_*` registers. Once the setting is done, the software begins to use AES-128 by setting the cipher mode in `the BB_AESCNTL_AES_MODE` bit of the `BB_AESCNTL` register, and writing a 1 to the `BB_AESCNTL_AES_START` bit of the same register. On normal termination, the software receives a `BLE_CRYPT_IRQ`. When finished, the software can find the ciphered data at the `AESPTR`+16 address, as shown in the "Memory Mapping for AES-128 Software Use" figure (Figure 25) (considering byte address memory).

When an encrypted link layer event has to be processed, it is possible that the event controller and packet controller require AES-CCM to run while a software request is ongoing. In this case, the current ciphering process is stopped, as real time AES-CCM processing has higher priority. When the real time encryption process ends, the AES-128 FSM restarts the ciphering process until normal termination, and then the `BB_AESCNTL_AES_START` bit of the `BB_`

AESCNTL register is reset. The BB_AESPTR_AESPTR address defined in the BB_AESPTR register needs to point to the exchange memory at a location not used by the baseband controller. The software can use the below code to make sure the address pointer is not used by HW:

```
BB_AESPTR = (EM_ENC_IN_OFFSET >> 2);
```

NOTE: AES deciphering is not supported.

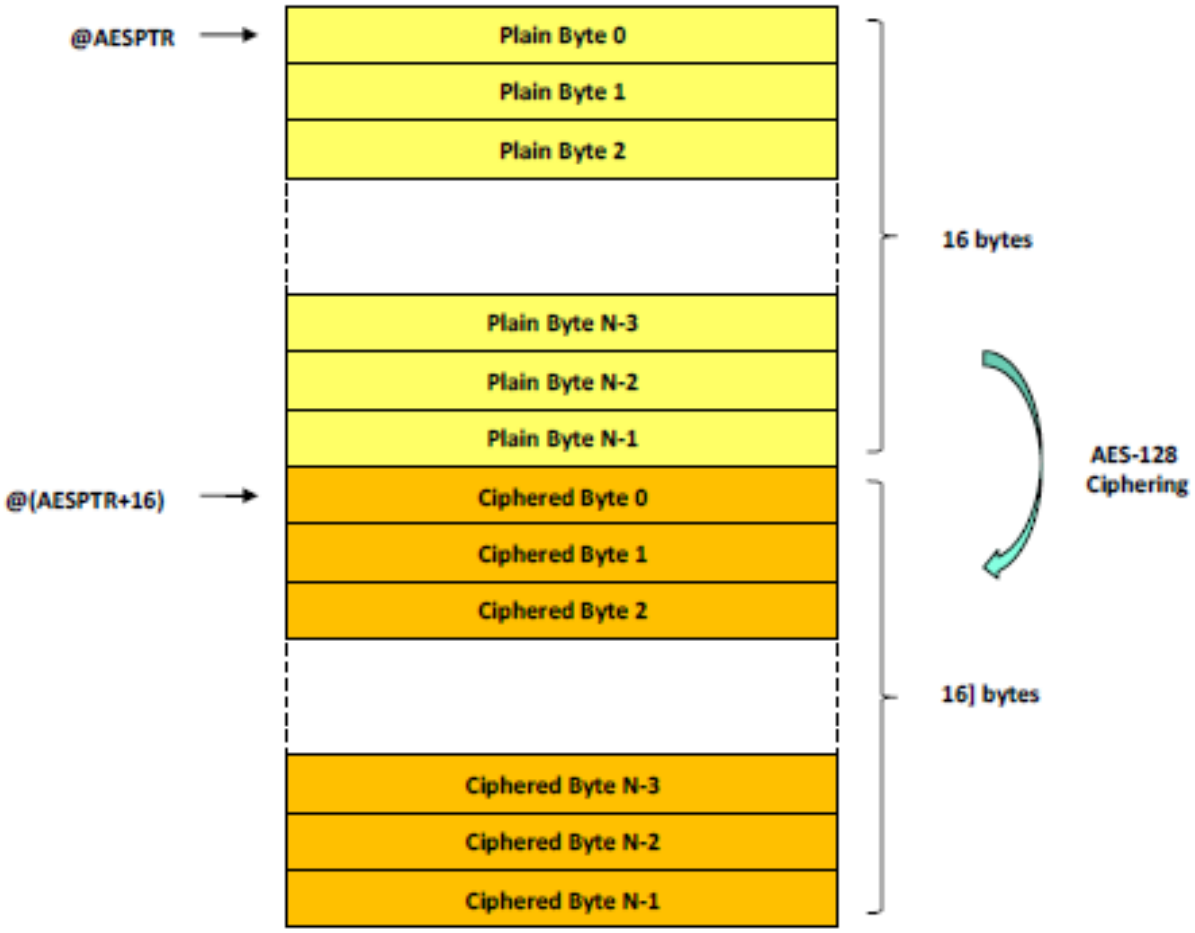


Figure 25. Memory Mapping for AES-128 Software Use

6.2.4 RF Test Modes

RF Test modes are dedicated to both Bluetooth RF qualification and regulatory body support. There are two main test modes supported by the BB controller:

- TX test mode, which is Bluetooth RF qualification oriented for TX modes.
- RX test mode, which is Bluetooth RF qualification oriented for RX modes.

The access address is fixed to 0x71764129 for RF test packets.

In the Bluetooth Low Energy specification, these tests are run in so-called Direct Test Mode (DTM). When an HCI or GAP DTM test mode command is received, the Bluetooth Low Energy software stack programs the BB controller, which then takes care of programming and sending DTM packets or configuring the RFFE for RX DTM Mode accordingly. Once the DTM `end` command is received, the Bluetooth Low Energy stack disables and removes the DTM event, and reads the number of correctly-received packets from BB controller register `BB_RXPKTCNT`, if the test mode is DTM. The number of transmitted packets can be read from register `BB_TXPKTCNT` if the text mode is DTM TX.

6.2.5 Angle of Arrival (AoA) and Angle of Departure (AoD) Support

AoA/AoD can be performed through ACL connection or periodic advertising. It operates over LE 1M and LE 2M only. CTE information is located either in the packet header (ACL or Direct test Mode cases) or in the extended packet header (periodic advertising case). Both carry the same data, which is related to the type of operation (i.e., AoA/AoD) and CTE field duration. Depending on the use case, specific counting mechanisms are required for aligning baseband timings to radio/modem timings, for antenna switching instant and pattern control (TX or RX), and I&Q sampling instant (RX only).

To support AoA/AoD, the transmitter needs to add a constant tone extension (CTE) following the CRC part of a Bluetooth Low Energy packet. The CTE field is divided into three periods:

1. 4 µs guard period, during which:
    a. Reference antenna (i.e., the first antenna in the list) must be used.
    b. Switching is not allowed for either transmitter or receiver.
    c. Sampling is not allowed for either transmitter or receiver.
2. 8 µs reference period, during which:
    a. Reference antenna (i.e., the first antenna in the list) must be used.
    b. Switching is not allowed for either transmitter or receiver.
    c. Sampling is required for the receiver.
3. the rest of the CTE period
    a. Antenna pattern is applied according to the provided antenna list.
    b. Switching is allowed for either transmitter or receiver, depending on the use case (AoA/AoD).
    c. Sampling is required for the receiver.

For further information, refer to the *Bluetooth Core Specification version 5.2*.

Antenna switching happens during AoD transmission or AoA reception. During AoD operations, the antenna switching interval (i.e., 1 µs or 2 µs) is determined by the value of `CTEType`, which is a field in the Bluetooth Low Energy packet header specified in the Bluetooth core specification.

The registers responsibe for various antenna timing controls are show in the "AoD Transmission Antenna Switching Time Controls" table (Table 7) and the "AoA Reception Antenna Switching Time Controls" table (Table 8).

Table 7. AoD Transmission Antenna Switching Time Controls

Duration	Control Register	Control Field from Register
1 Mbps PHY, 1 µs slot duration	BB_DFCNTL0_1US	TXSWSTINST0_1US
1 Mbps PHY, 2 µs slot duration	BB_DFCNTL0_2US	TXSWSTINST0_2US
2 Mbps PHY, 1 µs slot duration	BB_DFCNTL1_1US	TXSWSTINST1_1US
2 Mbps PHY, 2 µs slot duration	BB_DFCNTL1_2US	TXSWSTINST1_2US

Table 8. AoA Reception Antenna Switching Time Controls

Duration	Control Register	Control Field from Register
1 Mbps PHY, 1 µs slot duration	BB_DFCNTL0_1US	RXSWSTINST0_1US
1 Mbps PHY, 2 µs slot duration	BB_DFCNTL0_2US	RXSWSTINST0_2US
2 Mbps PHY, 1 µs slot duration	BB_DFCNTL1_1US	RXSWSTINST1_1US
2 Mbps PHY, 2 µs slot duration	BB_DFCNTL1_2US	RXSWSTINST1_2US

These register fields are in microsecond units. By increasing them, switching times are delayed, and vice versa.

RX I&Q sampling start instants are configured through the fields shown in the "RX I&Q Sampling Start Instant Configuration Controls" table (Table 9).

Table 9. RX I&Q Sampling Start Instant Configuration Controls

Duration	Control Register	Control Field from Register
1 Mbps PHY, 1 µs slot duration	BB_DFCNTL0_1US	RXSAMPSTINST0_1US
1 Mbps PHY, 2 µs slot duration	BB_DFCNTL0_2US	RXSAMPSTINST0_2US
2 Mbps PHY, 1 µs slot duration	BB_DFCNTL1_1US	RXSAMPSTINST1_1US
2 Mbps PHY, 2 µs slot duration	BB_DFCNTL1_2US	RXSAMPSTINST1_2US

The Bluetooth Low Energy stack configures sampling times and antenna switching times for an antenna switching HW, with typical 400 ns switching times from control signal to actual switching. if an external antenna switching HW has different timings, users might need to tune them accordingly. The "AoD CTE Transmission Timing Diagram" figure (Figure 26) and the "AoA CTE Reception Timing Diagram" figure (Figure 27) provide timing diagrams for actual HW signals during AoD TX and AoA RX:

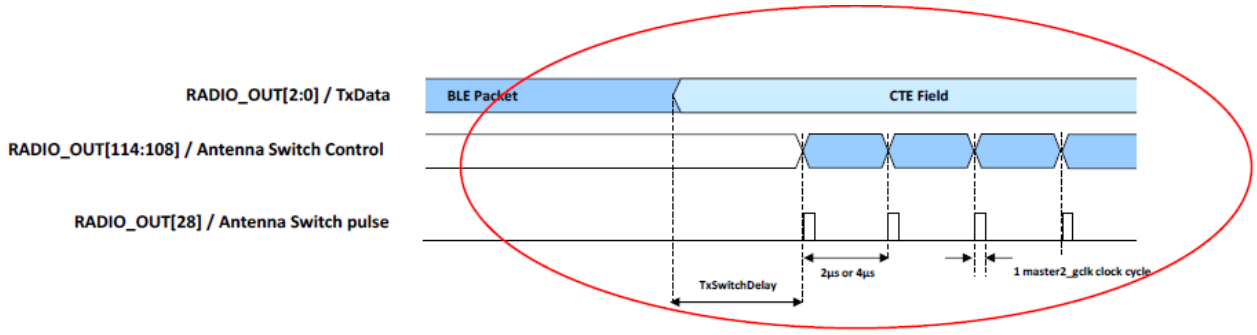


Figure 26. AoD CTE Transmission Timing Diagram

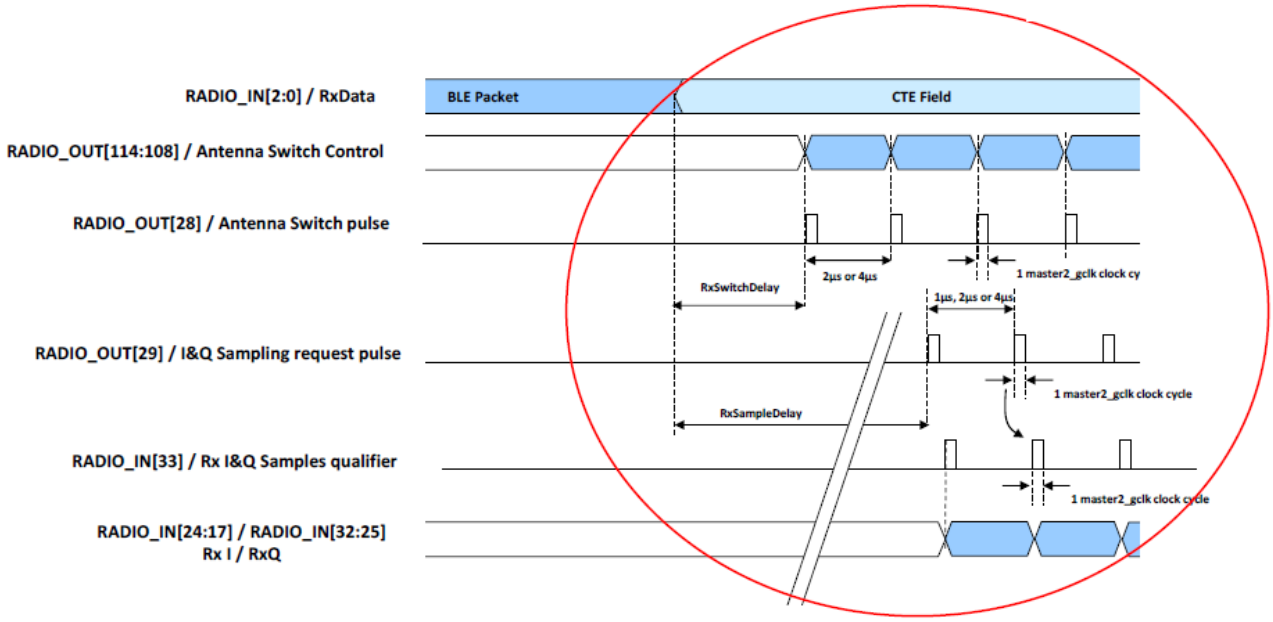


Figure 27. AoA CTE Reception Timing Diagram

6.3 Baseband Controller Interface

6.3.1 Baseband Controller Clock

The BB controller master clock is a scaled-down version of the system clock, configured through the `CLK_DIV_CFG0_BBCLK_PRESCALE` field of the `CLK_DIV_CFG0` register, and must fulfill these requirements:

- The ratio between the system clock and the BB controller master clock frequencies must be an integer.
- The BB controller master clock frequency must be an integer multiple of 1 MHz, from 6 MHz up to 16 MHz.

The internal baseband controller clock divider must be set according to the BB controller master clock frequency, to generate a 1 MHz reference clock. It can be configured in the `BBIF_CTRL_CLK_SEL` field of the `BBIF_CTRL` register with an integer value from 6 up to 24.

The `CLK_ENABLE` bit controls the external gating of the different baseband controller clocks. If the baseband controller is not used, we recommend that you disable the clocks and power down the baseband controller via the `SYSCTRL_RF_POWER_CFG` register. We also recommend that you power up the baseband controller using the following sequence:

```
/* Enable BB power switches */
SYSCTRL->RF_POWER_CFG |= BB_POWER_STARTUP;
SYSCTRL->RF_POWER_CFG |= BB_POWER_ENABLE;

/* Remove BB isolation */
SYSCTRL->RF_ACCESS_CFG |= BB_ACCESS_ENABLE;

/* Enable BB clocks */
BBIF->CTRL = BB_CLK_ENABLE;
```

NOTE: By default, the Bluetooth Low Energy initialization function (`BLE_Initialize`) assumes that the application has enabled the BB controller master clocks and provides a master clock of 8 MHz. Although configuring 16 MHz works, full functionality is not guaranteed, and this setting is not recommended.

6.3.2 WLAN Coexistence

RSL15 can be collocated with another 2.4 GHz radio technology such as another Bluetooth device or WiFi chip; this is referred to as WLAN coexistence. For example, they might share the same antenna, but the electromagnetic interference might impact the performance and/or functionality of both systems. The coexistence functionality can be used when, for example, an external amplifier is required, where it needs to be activated before a TX activity. For this purpose the BB controller can provide some signals to an external device indicating its radio (Bluetooth Low Energy) activity, and it can accept the other device's indication as an input signal that is called a WLAN signal. The BB controller includes the following provisions for Coexistence with WLAN devices:

- Ability to report real time Bluetooth Low Energy TX and RX activity to a collocated WLAN device
- Ability to abort Bluetooth Low Energy TX and/or RX activity when real time WLAN RX activity is reported
- Ability to abort Bluetooth Low Energy TX and/or RX activity when real time WLAN TX activity is reported
- A Bluetooth Low Energy real time synchronization signal

As output signals, the following signals can be routed through any configured GPIO to report Bluetooth Low Energy activities through configuring the `GPIO_CFG*` registers:

- WLAN coex sync signal that is generated synchronously with internal 625 µs timing reference
- WLAN coex in-process signal that is raised when any TX or RX is ongoing
- WLAN coex TX signal that is generated when a TX activity is happening
- WLAN coex RX signal that is generated during any RX activity
- WLAN coex PTI signal (4 bits) that indicates the priority value of the ongoing activity

The Bluetooth Low Energy activity signals (ble_rx and ble_tx) can be generated before the actual time when are scheduled for, by a maximum of 16 µs in advance. This is done through the `BB_COEXIFCNTL2_RX_ANT_DELAY` and `BB_COEXIFCNTL2_TX_ANT_DELAY` fields of the `BB_COEXIFCNTL2` register. By default, this time is set to zero.

A Bluetooth Low Energy activity (RF activity) can be aborted if the relevant external WLAN signal is set (providing it is routed through a GPIO), or when the software sets the `BBIF_COEX_CTRL_TX` or `BBIF_COEX_CTRL_RX` field of the `BBIF_COEX_CTRL` register. The abort mechanism can be enabled or disabled through the `BB_COEXIFCNTL0_WLANTXMSK` and `BB_COEXIFCNTL0_WLANRXMSK` fields of the `BB_COEXIFCNTL0` register.

As input, WLAN TX and WLAN RX external activities can be routed from any GPIO to baseband controller input signals wlan_tx and wlan_rx. Do this by configuring the desired GPIO as input and selecting accordingly via the `GPIO_SRC_BB_COEX` register. (See Chapter 10 "General Purpose Input/Output" on page 511for more information.)

The "WLAN Coexistence Timing when WLC<TX/RX>PRIOMODE = “00”" figure (Figure 28) and the "WLAN Coexistence Timing when WLC<TX/RX>PRIOMODE = “01”" figure (Figure 29) show two WLAN coexistence interface timing diagrams, including TX/RX power-up delay for two modes, when the `BB_COEXIFCNTL0_WLCRXPRIOMODE` and `BB_COEXIFCNTL0_WLCTXPRIOMODE` fields in the `BB_COEXIFCNTL0` register are set to 0 or 1 to include or exclude TX and RX power-up delays (see Section 1.0.1 “BB_COEXIFCNTL0” on page 1 for field descriptions):

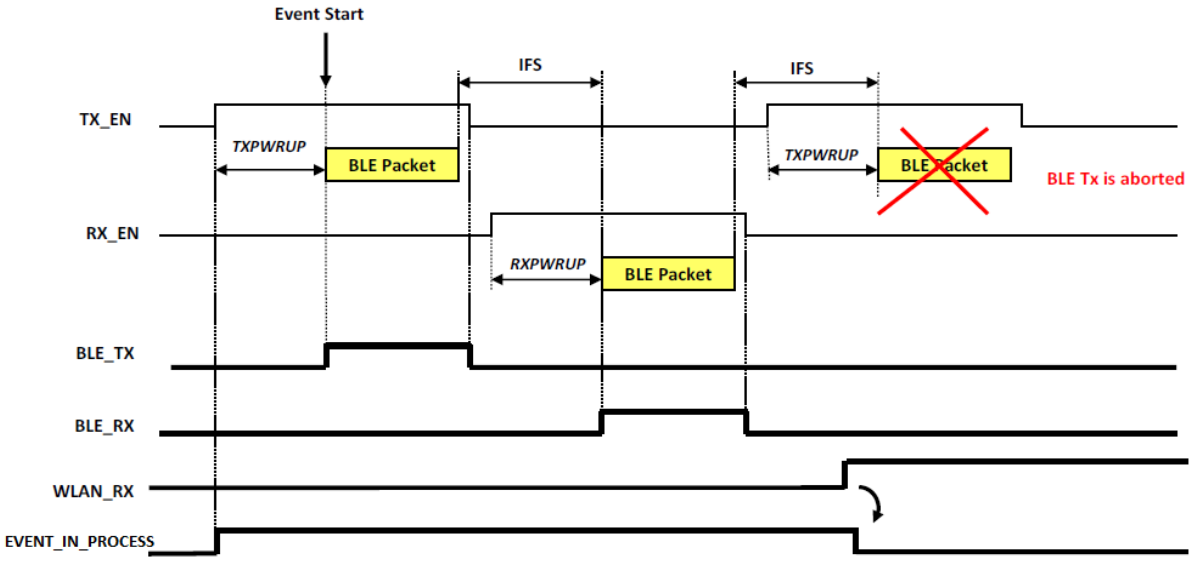


Figure 28. WLAN Coexistence Timing when WLC<TX/RX>PRIOMODE = "00"

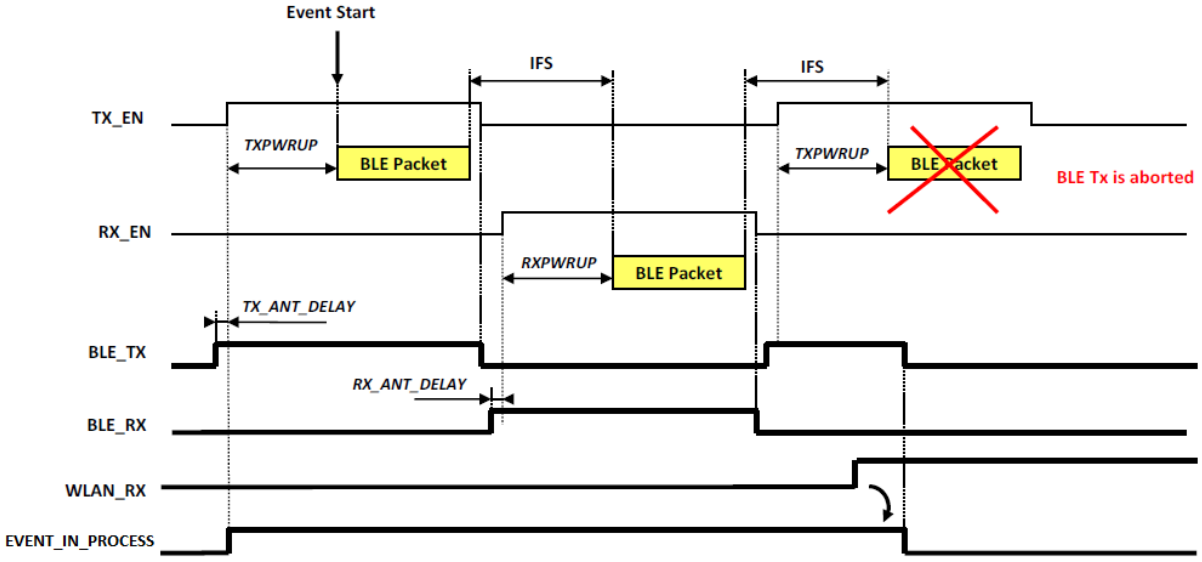


Figure 29. WLAN Coexistence Timing when WLC<TX/RX>PRIOMODE = "01"

The status of Bluetooth Low Energy activities can be read at any time through register `BBIF_STATUS`. The `BBIF_STATUS_CLK_STATUS` field in this register is low when it is in Active Clock Mode, and is high when it switches to the Low Power Clock Mode that is called Bluetooth Low Energy Deep Sleep Mode. The `BBIF_STATUS_LINK_FORMAT` field in this register indicates the type of activity that is ongoing The `BBIF_STATUS_LINK_LABEL` field shows the Bluetooth Low Energy activity index, which the application is informed of through GAP APIs when an activity is created. The `BBIF_STATUS_LINK_FORMAT` values are as listed below (in the binary format found in the field):

- 00010: Master Connect
- 00011: Slave Connect
- 00100: Low Duty Cycle Advertiser
- 00101: High Duty Cycle Advertiser
- 00110: Extended Advertiser
- 01000: Passive Scanner
- 01001: Active Scanner
- 01010: Extended Passive Scanner
- 01011: Extended Active Scanner
- 01110: Initiator
- 01111: Extended Initiator
- 11100: TX Test Mode
- 11101: RX Test Mode
- 11110: TX/RX Test Mode

The status of Bluetooth Low Energy coexistence signals can be read through register BBIF_COEX_STATUS.

In addition, BBIF allows coexistence signals to generate a desired interrupt when any TX or RX activity starts or finishes. The desired interrupt is enabled by setting the fields of register BBIF_COEX_INT_CFG.

Interrupts can be configured for either rising edge or falling edge, or any edge of the BBIF_COEX_STATUS_BLE_TX and BBIF_COEX_STATUS_BLE_RX signals. The signal bit BBIF_COEX_STATUS_BLE_IN_PROCESS in the BBIF_COEX_STATUS register is set when any of radio bits BBIF_COEX_STATUS_BLE_RX or BBIF_COEX_STATUS_BLE_TX are set. The signal bit goes back to low when the radio activities finish.

The BBIF_COEX_STATUS_EVENT_IN_PROCESS signal is raised when the baseband controller schedules an event, not when radio activity (including TX, RX, and IFS) is ongoing, and it is different with BBIF_COEX_STATUS_BLE_IN_PROCESS. The BBIF_COEX_STATUS_EVENT_IN_PROCESS signal is shown in the "WLAN Coexistence Timing when WLC<TX/RX>PRIOMODE = “00”" figure (Figure 28) and the "WLAN Coexistence Timing when WLC<TX/RX>PRIOMODE = “01”" figure (Figure 29).

6.3.3 BB Controller Interrupts

The baseband controller interrupt generator creates the following interrupts:

- BLE_ERROR_IRQ: Error interrupt, generated when undesired behavior or bad programming occurs in the Bluetooth Low Energy Core. Error status is reported in the BB_ERRORTYPESTAT register.
- BLE_HSLOT_IRQ: 312.5 µs half slot interrupt, available in active mode. This interrupt can be partially masked using sub-rating patterns.
- BLE_SLP_IRQ: End of Sleep Mode interrupt
- BLE_CRYPT_IRQ: Encryption engine interrupt, generated when SW-driven AES-128 ciphering/deciphering process is finished
- BLE_FINETGT_IRQ: Fine Target Timer interrupt generated when Fine Target timer expires. Timer resolution is 312.5 µs.
- BLE_TIMESTAP_TGT1 and BLE_TIMESTAP_TGT2: Time Stamp Target interrupts generated when reaching a desired CLKN and fine counter value. Instant precision of 0.5 µs.
- BLE_SW_IRQ: SW triggered interrupt generated on SW request through register access (the BB_BLECNTL_SWINT_REQ of the BB_BLECNTL register)

- BLE_FIFO_IRQ: Activity FIFO interrupt, generated each time any of the following internal events is generated:
  - Start of events internal interrupt. Generated on any event starting instant.
  - End of events internal interrupt. Generated on normal termination, or on anticipated pre-fetch mechanism request.
  - Skipping of events internal interrupt. Generated when an event is not processed for priority management reasons.
  - End of transmitted and acknowledged packet, or when number of transmitted packets equals a threshold set by the Bluetooth Low Energy stack
  - End of received packets, or when number of received packets equals a threshold set by the Bluetooth Low Energy stack

Interrupts can be masked using the BB_INTCNTL* registers. Each interrupt status is provided in the BB_INTSTAT* registers. Interrupts are acknowledged by writing to the corresponding BB_INTACK* registers. Each interrupt has its own output to the Arm Cortex-M33 processor interrupt controller.

The BB_ACTFIFOSTAT register provides internal details about real-time Bluetooth Low Energy event operations. These details are stored in the FIFO IRQ, including separated fields for skipped events and the current event exchange table entry index.

The FIFO has an interrupt depth of 16, and generates an interrupt each time it is loaded by any internal interrupt while empty. FIFO interrupt status is kept high until all the entries are read by the Bluetooth Low Energy software. Accessing the next element of the FIFO is performed by acknowledging the FIFO interrupt. The basic procedure to read the FIFO IRQ is the following:

1. Bluetooth Low Energy software waits for a BLE_FIFO_IRQ to be generated.
2. BLE_FIFO_IRQ is generated, and Bluetooth Low Energy software enters its FIFO interrupt management routine.
3. Bluetooth Low Energy software reads the value in the BB_ACTFIFOSTAT register, and processes the reported pending interrupt(s).
4. Bluetooth Low Energy software acknowledges the FIFO interrupt, asking the FIFO interrupt to provide next elements (if any).
5. Bluetooth Low Energy software reads the BB_INTSTAT1_FIFOINTSTAT field of the BB_INTSTAT1 register to determine whether FIFO is empty.
6. If the FIFO is not empty, loop to step 3.
7. If the FIFO is empty, loop to step 1.

When the Bluetooth Low Energy software stack is used, it handles all Bluetooth Low Energy interrupts, and interrupt software routines are not exposed to the application.

6.3.4 Baseband Counters and Timers

The BB controller can be put in two modes:

- Active Mode
- Low Power Mode (Bluetooth Low Energy Sleep Mode)

6.3.4.1 Active Mode

In Active Mode, which is the normal mode in which Bluetooth Low Energy events are processed, there are some counters and timers that are used by the Bluetooth Low Energy software stack. Counters are also available to the applications, though timers are not. If an application does not use the Bluetooth Low Energy stack, the application can use these counters as general purpose timers. There are three of these counters, as follows:

- CLKN: A 28-bit counter with a 312.5 µs precision, which wraps after approximately 23.3 hours
- FINECNT: A 10-bit fine counter with 0.5 us precision, which wraps every 625 increments to zero, meaning that it counts up to 624 µs
- BB low power clock timer: A 32-bit wakeup counter that counts the number of low power clk clock cycles (used only in Low Power Mode)

The CLKN and FINECNT counters are clocked synchronously.

The values of the BB_SLOTCLK and FINETIMECNT counters can be read at any time by the software or the Bluetooth Low Energy stack, using the following mechanism:

- Set the field `BB_SLOTCLK_SAMP` field of the `BB_SLOTCLK` register to get the current samples of counters.
- Wait until this field is reset, then read the saved sample of the counters from the `BB_CLKN_SCLK` field of the `BB_CLKN` register (28 bits) and the `BB_FINETIMECNT_FINECNT` field of the `BB_FINETIMECNT` register (10 bits).

We recommend that you disable interrupts before reading the values of counters, and enable the interrupts again once the counters have been read. These counters can be used in combination by software at any time, as a general real-time timer.

In the Bluetooth Low Energy stack, these counters are used by hardware to capture the times at which any activity or event has occurred in shared memory, so that later the Bluetooth Low Energy stack can read them and find out the real-time value associated with any event or packet.

There are three timers that can generate interrupts when they are configured to be used by software for a desired period of time. When the Bluetooth Low Energy stack is used, these timers are not available to applications, as the Bluetooth Low Energy stack uses them for its own internal protocol timers and kernel timer. They are:

- Fine Timer Target: A 28-bit timer with 312.5 µs precision
- Timestatmp Target 1: with 0.5 µs precision
- Timestamp Target 2: with 0.5 µs precision

These timers can be configured by setting the desired target values and enabling the appropriate interrupt bit in the `BB_INTCNTL0` register.

Refer to Section 6.6 “Baseband Controller Registers” on page 331 for descriptions of these registers/fields:

- The `BB_FINETIMTGT_BB_FINETARGET` field of the `BB_FINETIMTGT` register
- The `BB_HMICROSECTGT1_HMICROSECTGT1` field of the `BB_HMICROSECTGT1` register
- The `BB_CLKNTGT1_CLKNTGT1` field of the `BB_CLKNTGT1` register
- The `BB_HMICROSECTGT2_HMICROSECTGT2` field of the `BB_HMICROSECTGT2` register
- The `BB_CLKNTGT2_CLKNTGT2` field of the `BB_CLKNTGT2` register

These timers use the counters and target value registers to generate interrupts when the counters reach the configured target values.

The Bluetooth Low Energy software stack configures Timestamp Target2 with resolution 1 ms for kernel timer purposes when required, which means that the kernel timer has a resolution of 1 ms.

6.3.4.2 Low Power Mode

In RSL15, when the Bluetooth Low Energy controller is put in Low Power Mode (Deep Sleep Mode), the low power timer counts the number of low power clock cycles when all clocks are disabled and/or the BB controller is powered off. This wakes up the BB controller at the right time, so that it can continue processing the scheduled activities programmed by the Bluetooth Low Energy software stack. The low power timer is sourced from an RTC clock that can be any one of the following options:

- Crystal 32768 Hz (XTAL 32K)
- RC 32 kHz oscillator
- One of GPIOs 0 to 3

By default, the timings for sleep duration are calculated assuming a 32768 low power clock frequency. However, any desired frequency (for example, between 10 and 100 kHz) can be provided externally through GPIOs. Therefore, the Bluetooth Low Energy stack needs to be informed of the frequency for its own calculations. This takes place through the API `LPCLK_PeriodValue_Set(average_period)` where an application has to call it, at least in `PARAM_ID_LPCLK_NO_XTAL32K` in the function `uint8_t Device_BLE_Param_Get(uint8_t param_id, uint8_t *lengthPtr, uint8_t *buf)`. The value of the `average_period` argument needs to be provided in the application. For example, the code below shows how a 40 kHz clock can be used as the source of the low power clock:

```
...

#define LPCLK_PERIOD_VALUE              (float)(1000000.0 / 40000)
case PARAM_ID_LPCLK_NO_XTAL32K:
buf[0] = true;
LPCLK_PeriodValue_Set(LPCLK_PERIOD_VALUE);
break;

...
```

The accuracy of the low power clock must be better than 500 ppm to be able to keep the Bluetooth Low Energy connection stable, to send periodic advertisements, or to synchronize with a peer device's periodic advertisements. However, a mechanism has been implemented that allows acceptance of any desired clock accuracy through `PARAM_ID_LPCLK_DRIFT` in the `Device_BLE_Param_Get()` function. When low power clock accuracy varies (for example, because of fast temperature changes for the 32 kHz crystal oscillator), this mechanism allows the application to provide greater than 500 ppm accuracy, while still maintaining the Bluetooth Low Energy link with a central device. Accepting clock accuracies higher than 500 ppm is allowed only for devices in the peripheral role, according to the Bluetooth Low Energy Core Specification.

Refer to the *ble_peripheral_server_sleep* sample application to see how an application can configure the clock accuracy.

To keep the baseband low power timer active when the device is in Sleep Power Mode (with VDDC retention regulator disabled), the following procedure needs to be applied:

1. Enable the `ACS_VDDRET_CTRL_VDDTRET_EN` bit in the `ACS_VDDRET_CTRL` register to keep the baseband low power timer powered.
2. Configure the baseband controller register `BB_DEEPSLWKUP` to determine the time to be spent (in low power clock cycles) before waking up the device.
3. In the baseband controller register `BB_ENBPRESET`, configure these fields:
    a. `BB_ENBPRESET_TWOSC`, defining the time (in low power clock cycles) allowed for stabilization of the high frequency oscillator. The `BB_ENBPRESET_TWOSC` field controls the signal osc_en, which is

internally used to wake up the ACS and bring back the system from Sleep Mode to Run Mode.

b. BB_ENBPRESET_TWRM, defining the time (in low power clock cycles) allowed for the radio module to leave Low Power Mode. The BB_ENBPRESET_TWRM field controls the signal radio_en, which is internally used to manage the isolation of the baseband low power timer when the device is powered down. Only the baseband low power timer input signals are isolated (except the external wakeup request available through the BBIF_CTRL_WAKEUP_REQ field of the BBIF_CTRL register). The output signals remain available.

4. Make sure that the following conditions are fulfilled:
   a. The sleep time defined in the register BB_DEEPSLWKUP is larger than the value of the BB_ENBPRESET_TWOSC field in the BB_ENBPRESET register.
   b. The BB_ENBPRESET_TWOSC value is large enough that the system has time to go back to Run Mode and be ready to catch the baseband wakeup interrupt BLE_SLP_IRQ.
   c. The BB_ENBPRESET_TWOSC value is larger than the value of the BB_ENBPRESET_TWRM field in the same register (at least two low power clock cycles), so that the system has time to go back to Run Mode before the baseband low power timer isolation is removed.
5. Request the baseband low power timer to go into Deep Sleep Mode by setting these three bits in the BB_DEEPSLCNTL register:
   - BB_DEEPSLCNTL_DEEP_SLEEP_ON
   - BB_DEEPSLCNTL_RADIO_SLEEP_EN
   - BB_DEEPSLCNTL_OSC_SLEEP_EN
6. Wait until the status bits BBIF_STATUS_OSC_EN and BBIF_STATUS_RADIO_EN of the BBIF_STATUS register are reset, indicating that the baseband low power timer is in Deep Sleep Mode and properly isolated.
7. Set the system to Sleep Power Mode.

When coming back from Sleep Mode to Run Mode, we recommend that you use the following procedure:

1. The OSC_EN signal is set by the baseband low power timer, and automatically wakes up the ACS, which brings back the system from Sleep Mode to Run Mode.
2. The RADIO_EN signal is set by the baseband low power timer and the isolation is automatically removed.
3. Power up the baseband controller by setting consecutively the SYSCTRL_RF_POWER_BB_STARTUP bit and, at least 2 µs later, the SYSCTRL_RF_POWER_BB_ENABLE bit, both from the SYSCTRL_RF_POWER register.
4. Remove the baseband controller isolation by setting the SYSCTRL_RF_ACCESS_BB_ACCESS bit of the SYSCTRL_RF_ACCESS register.
5. Enable the baseband controller clocks by setting the BBIF_CTRL_CLK_ENABLE bit of the BBIF_CTRL register.
6. Wait for the baseband wakeup interrupt BLE_SLP_IRQ, which is generated once the baseband low power timer sleep timer has expired.

In addition to the baseband low power timer, it is possible to wake up the baseband controller by setting the BBIF_CTRL_WAKEUP_REQ bit of the BBIF_CTRL register. Since no external signal is routed to the baseband controller from a GPIO, it is the responsibility of the application (software) to control this bit, for example, when RSL15 is wakened up by an external wakeup source.

The "Active Mode to Low Power Mode Timing Diagram" figure (Figure 30), the "Deep Sleep Mode – Standard Termination by Baseband Low Power Timer" figure (Figure 31), and the "Deep Sleep Mode – Aborted by Wakeup Request" figure (Figure 32) show the timing when the baseband controller is in Deep Sleep Mode (low power clock mode) and when it is awakened by the baseband low power timer or by an external wakeup request:

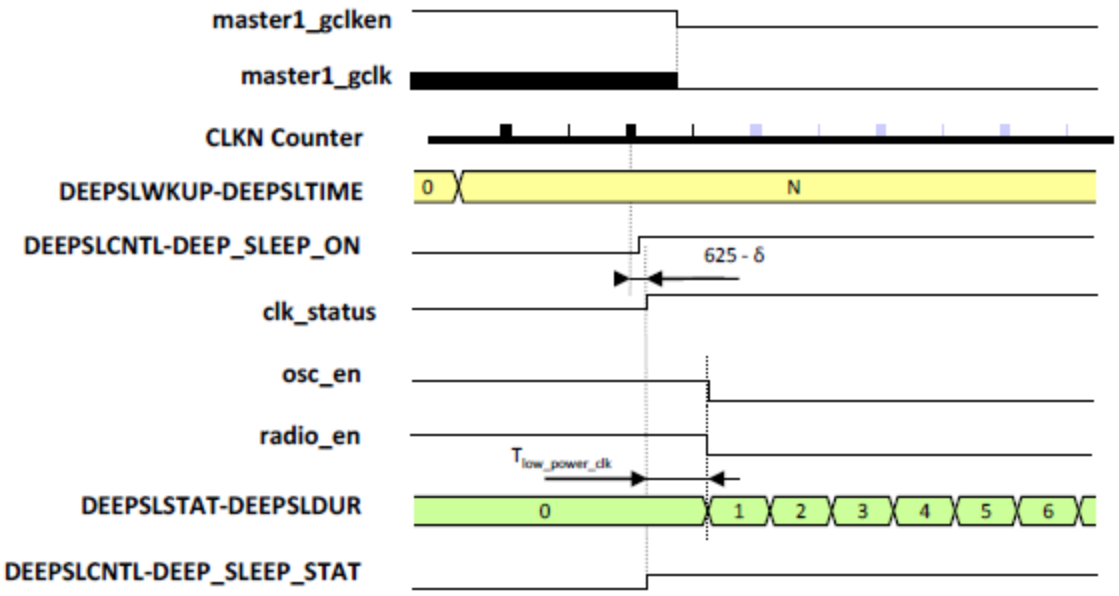


Figure 30. Active Mode to Low Power Mode Timing Diagram

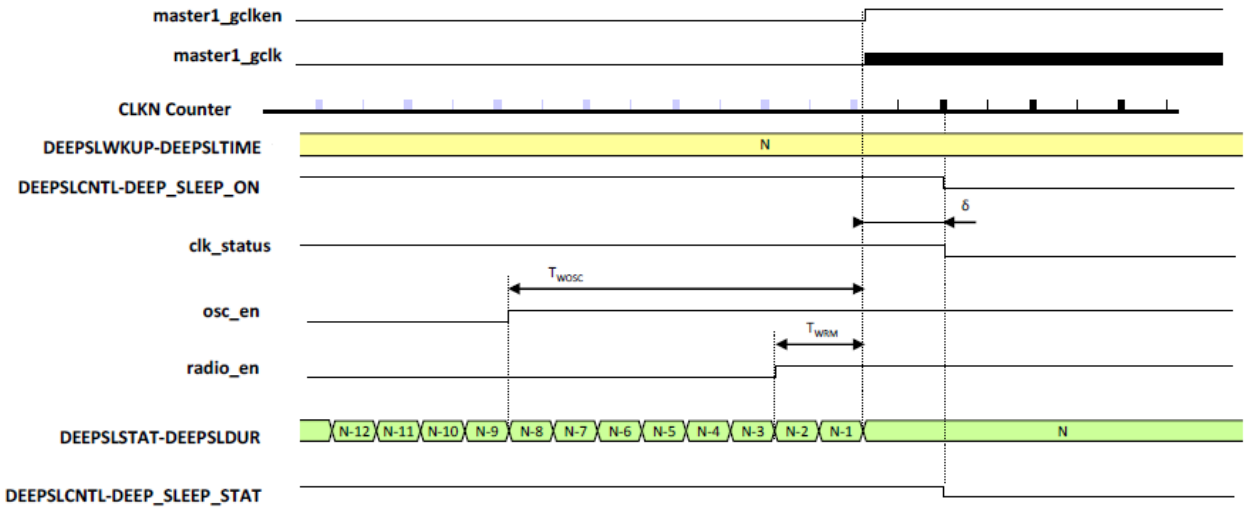


Figure 31. Deep Sleep Mode – Standard Termination by Baseband Low Power Timer

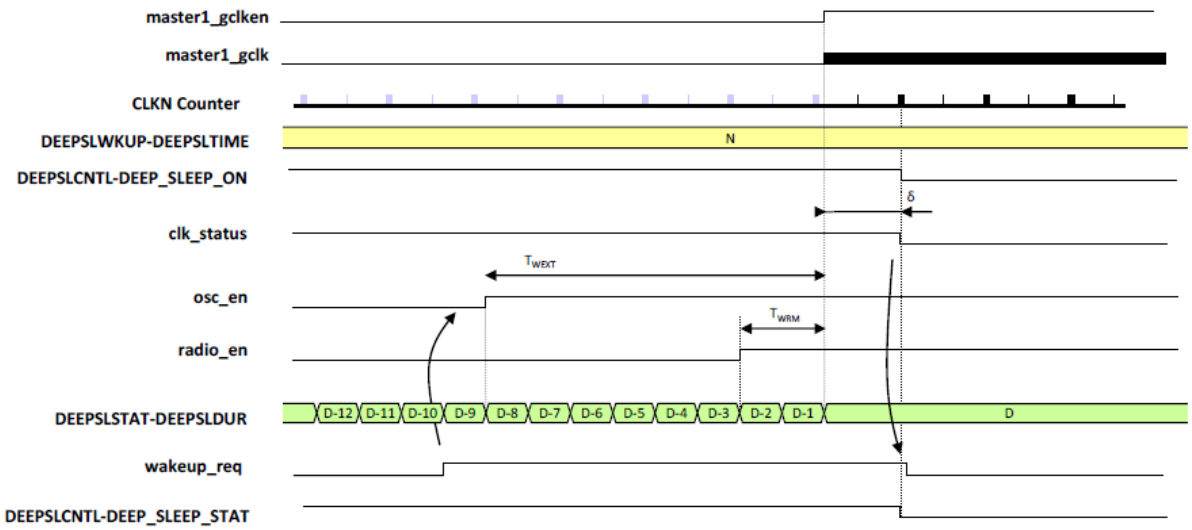


Figure 32. Deep Sleep Mode – Aborted by Wakeup Request

An external wakeup request can be filtered when the BB_DEEPSLCNTL_EXTWKUPDSB field in the EXTWKUPDSB register is set. The wakeup request must be maintained high for at least two low power clock cycles to be taken into account correctly. Although the BB_ENBPRESET_TWOSC, BB_ENBPRESET_TWRM, and BB_ENBPRESET_TWEXT fields in the BB_ENBPRESET register can be set by the application, we suggest that the BB_ENBPRESET_TWOSC parameters only be adjusted when the Bluetooth Low Energy stack is used. After wakeup, the value of BB_ENBPRESET_TWOSC is the time that the system needs to power on the BB controller (if it is off), enable RFCLK (if it is disabled), and restore the BB controller registers and RFFE register contents (if BB and RFFE have been off during sleep). Before going to BB deep sleep, if the baseband and RFFE are powered off, the contents of their registers need to be saved in DRAM, and maintained by enabling VDDM for the appropriate DRAM instance(s). The application needs to make sure that after wakeup, the BB controller and the RFFE can resume their own activities—for example, to maintain a Bluetooth Low Energy connection.

At the end of each low power time, the time duration is written to the BB_DEEPSLSTAT_DEEPSLDUR field of the BB_DEEPSLSTAT register. Then the software must apply CLKN counter and Fine counter correction to restore the 312.5 µs CLKN reference alignment. This correction mechanism can be achieved using the BB_CLKNCNTCORR and BB_FINECNTCORR registers. This mechanism is mandatory, since the CLKN counter is running onto the master clock that is gated during Deep Sleep Mode, freezing both counters, and letting the system run onto the low power clock. The correction is achieved by updating both the 10-bit Fine counter values and the 28-bit CLKN counter value, i.e., correcting respectively the fractional part and the integer part of the phase error of the 312.5 µs reference timing event internal counter.

After termination of the Deep Sleep period, the 312.5 µs reference timing event generator has to be corrected based on the actual time spent in Low Power Mode *mSleep* (*@T-osc*) reported in the BB_DEEPSLSTAT_DEEPSLDUR field of the BB_DEEPSLSTAT register.

$$mSleep = T - osc * DEEPSLDUR$$

The integer part of the 312.5 µs reference timing, *K*, covers an entire CLKN Counter range (i.e. between 0 and 228-1), and has to be written in the CLKNCNTCORR register. *K* corresponds to the CLKN Counter correction value to apply after Deep Sleep duration, assuming that a 32 kHz clock has been used. So this formula hereafter applies.

$$K = floor\left(\frac{mSleep}{312.5\mu s}\right)$$

The fractional part of the 312.5 µs reference counter, *R*, has to be written in the FINECNTCORR register. *R* covers an entire 312.5 µs Fine Counter range (i.e., between 0 and 624 with 0.5 µs precision). As the 10-bit Fine Counter is a downward counter, this *R* formula hereafter applies.

$$R = 2 * (312.5 - integer\,(mSleep - K * 312.5))$$

where the integer function provides the integer part of a number.

In any case of Deep Sleep termination, the Bluetooth Low Energy software must compensate both the CLKN counter and the FINECNT counter using (respectively) the CLKNCNTCORR and FINECNTCORR registers. Those values are calculated by the software, and applied by the BB controller as described here:

- On Deep Sleep termination, the software reads the DEEPSLSTAT value.
- The software determines the CLKNCNTCORR and FINECNTCORR values and stores them in the appropriate registers.
- The software signals the correction to be applied, writing a 1 to the BB_DEEPSLCNTL_DEEP_SLEEP_CORR_EN field of the BB_DEEPSLCNTL register.

CLKN Counter and Fine Counter correction is applied when Fine Counter equals zero. CLKN Counter is corrected by applying the following formula:

$$\textbf{new}\ CLKNCNT = \textbf{current}\ CLKNCNT + 1 + CLKNCNTCORR$$

Fine Counter correction is done by directly loading FINECNTCORR when Fine Counter equals zero. The "CLKN Counter Correction Mechanism" figure (Figure 33) shows the CLKN counter correction mechanism:

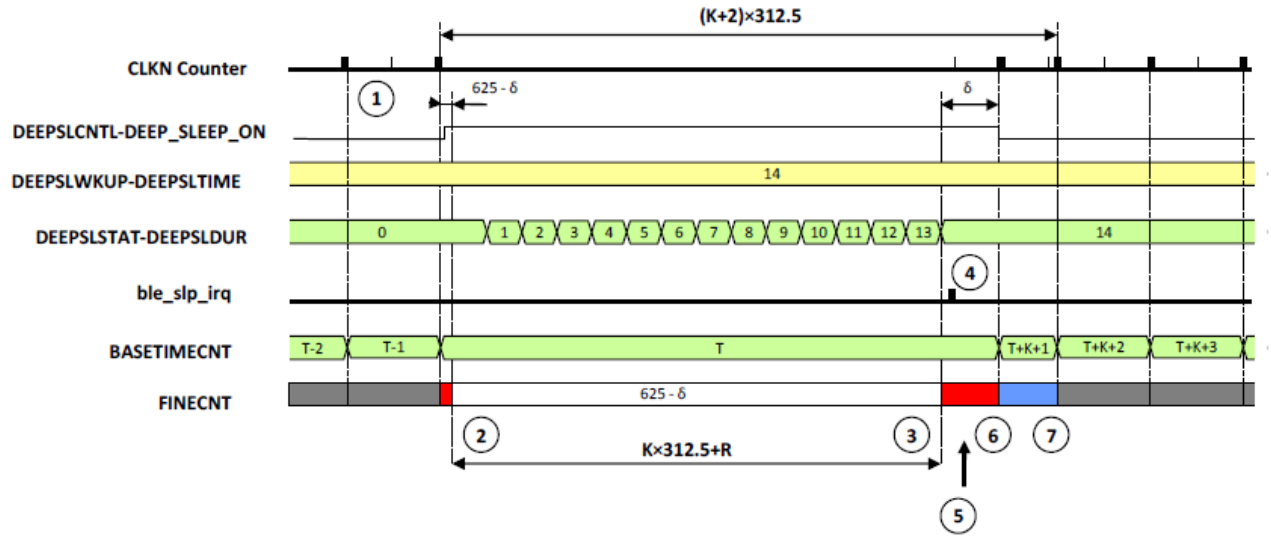


Figure 33. CLKN Counter Correction Mechanism

The following sequence of events occur for counter correction:

1. The software stack requires the system to go into Deep Sleep.
2. The software stack goes into Deep Sleep Mode, and freezes CLKN Counter to *T*, and Fine Counter to 625-δ.
3. When the number of low power clock cycles equals the value of `BB_DEEPSLSTAT_DEEPSLDUR`, or after a wakeup request, the Bluetooth Low Energy Core wakes up, and Fine Counter resumes.
4. BLE_SLP_IRQ is generated once the master clock is enabled, and the Bluetooth Low Energy software reads the value of the `BB_DEEPSLWKUP_DEEPSLTIME` field in the `BB_DEEPSLWKUP` register and calculates the *K* and *R* values.
5. The software stack programs CLKNCNTCORR = *K* and FINECNTCORR = *R*, and the requirements for compensation.
6. Once Fine Counter equals zero, CLKN Counter and Fine Counter are updated.
7. The 312.5 µs half slot timing reference is corrected and realigned.

6.3.5 Bluetooth Low Energy Stack Deep Sleep Mode

The functionality provided by the baseband controller is used by the Bluetooth Low Energy software stack to put itself into Sleep Mode, and allows the stack to resume its normal functionality after wakeup. This section describes how the Bluetooth Low Energy stack works for switching between Deep Sleep Mode and Active Mode, and how it interacts with a user application.

When sending the Bluetooth Low Energy stack into Sleep Mode, a user application needs to call the function (API) `BLE_Baseband_Sleep(&ble_sleep_api_param)`. This API checks whether it is the right time for the Bluetooth Low Energy stack to go to sleep or not; if the return value is `RWIP_DEEP_SLEEP`, then the BB controller is already in Sleep Mode, and the BB timer has been started by the low power clock. Hence, the application does not need to configure the BB controller registers and baseband low power clock. However, the rest of system must be put into the desired power mode, such as Sleep Mode or Standby Mode.

The user application is responsible for configuring the ACS, VDDC and VDDM retention regulators, deciding whether to power off BB, RFFE, and RFCLK or not, any other step that is specified in Section 8.6 “Power Modes” on page 431. From the BB controller’s point of view, it is in low power clock mode. Therefore, for cases when the `BLE_Baseband_Sleep()` function returns `RWIP_DEEP_SLEEP`, the `BB_Sleep()` function has been developed and provided in the RSL15 Bluetooth Low Energy sample applications for Low Power Modes as an example showing how put RSL15 into a Low Power Mode. In the `BB_Sleep()` function, the BB controller and RFFE registers are saved in DRAM to be restored later at wakeup time (if the BB controller and the RFFE are powered off).

The structure of *ble_sleep_api_param* returns the actual sleep duration that `BLE_Baseband_Sleep()` has programmed in register `BB-DEEPSLWKUP`, in the `calculated_sleep_duration` field of the low power clock unit (for example 1/32768 s).

These other fields are input for this function:

- `app_sleep_request` set to one means it goes to sleep; if the parameter equals zero, it returns without checking or going to sleep.
- `max_sleep_duration` is the maximum sleep duration that the application would like to be in sleep for, in 312.5 µs.
- `min_sleep_duration` is the minimum sleep duration that the application asks for from the stack, in µs.

The sleep duration is calculated based on the next immediate Bluetooth Low Energy event or the earliest kernel timer expiry time (whichever happens sooner), minus a processing delay of approximately 500 µs. The BB low power timer expires at the amount of time equal to the `BB_ENBPRESET_TWOSC` value sooner than the value that is programmed in the BB timer. The "BLE_Baseband_Sleep() Function Logical Flowchart for Checking and Going to Deep Sleep" figure (Figure 34) shows how the Bluetooth Low Energy stack checks if it is the right time to go to sleep, and then calculates the sleep duration. At the end of function `BLE_Baseband_Sleep()`, a flag is reset to indicate that the BB controller is in deep sleep, to be used by the application; this can be checked by calling API `BLE_Baseband_Is_Awake()`. If the return value of this API is false, `BLE_Kernel_Process()` and `BLE_Baseband_Sleep()` do not need to be called again.

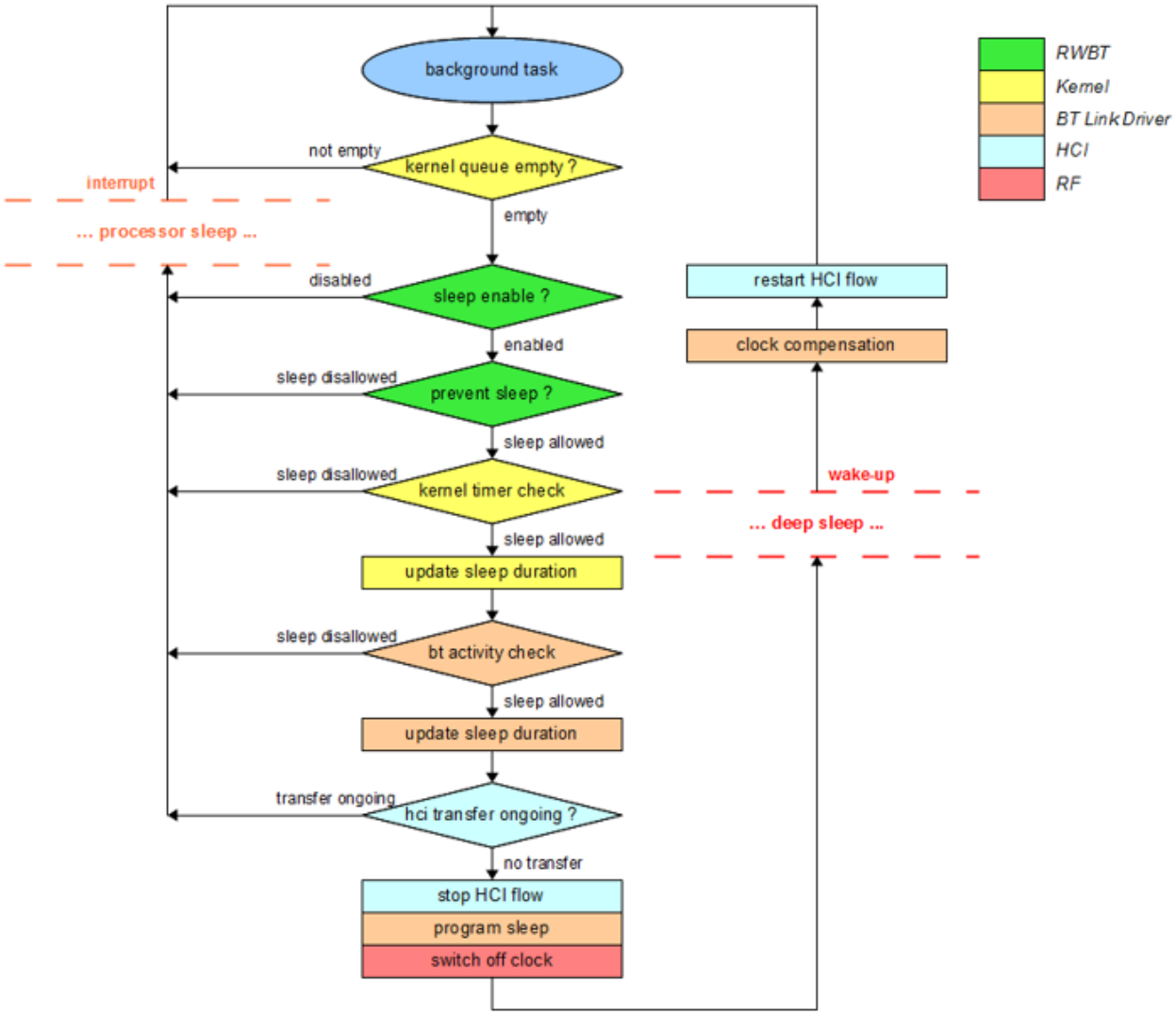


Figure 34. BLE_Baseband_Sleep() Function Logical Flowchart for Checking and Going to Deep Sleep

When the BB timer expires, generating a wakeup signal for the ACS block at a time equal to the value of `BB_ENBPRESET_TWOSC` sooner than the programmed sleep duration, BLE_SLP_IRQ is generated. In Bluetooth Low Energy SLP ISR, a mechanism called clock compensation is enabled and waits for a BLE_HSLOT_IRQ. At BLE_SLP_IRQ, the Bluetooth Low Energy stack corrects the value of the Bluetooth Low Energy clock according to the sleep duration, and waits for a BLE_HSLOT_IRQ when the clock correction is applied by the Bluetooth Low Energy HW. It sets back the `BLE_Baseband_Is_Awake` flag to enable the Bluetooth Low Energy application to continue its normal Bluetooth Low Energy functionality in active mode. All clock correction and compensations are handled by the Bluetooth Low Energy stack; user applications only need to make sure that at wakeup they prepare the required RSL15

HW blocks in terms of the RFCLK, the RFFE and BB power, and enabling the BB clock If the wakeup source is not the BB timer, then the application needs to wake up the BB controller by setting the BBIF_CTRL_WAKEUP_REQ bit in the BBIF_CTRL register.

If the Bluetooth Low Energy stack is not used, the clock correction must be handled by the user application.

6.3.5.1 Bluetooth Low Energy Stack Deep Sleep Mode Registers

For tables containing the details of these registers, see Section 6.4.0.1 “BBIF_CTRL” on page 326.

6.3.6 Baseband Controller Registers and Non-CTE Antenna Definition

Certain register fields are used to define the antenna used during the non-CTE part of the a packet.

On each TX start instant, if the TXPRIMIDCNTLEN field in the BB_DFANTRCNTL register is set, then the antenna to be used is defined by the TXPRIMANTID field in the same register.

On each RX start instant, if the RXPRIMIDCNTLEN field in the BB_DFANTRCNTL register is set, then the antenna to be used is defined by the RXPRIMANTID field in the same register.

These fields and settings in the BB_DFANTRCNTL register are used as shown in the "BB_DFANTRCNTL Fields Defining Antenna in Non-CTE Cases" table (Table 10).

Table 10. BB_DFANTRCNTL Fields Defining Antenna in Non-CTE Cases

Field Name	Description
TXPRIMANTID[6:0]	Primary antenna ID to be used on each TX start instant
TXPRIMIDCNTLEN	Transmit Primary antenna ID enable control
RXPRIMANTID[6:0]	Primary antenna ID to be used on each RX start instant
RXPRIMIDCNTLEN	Reception Primary antenna ID enable control

The current stack configuration it is set up with the following field values in the BB_DFANTRCNTL register:

- TXPRIMIDCNTLEN: 1
- TXPRIMANTID: 0
- RXPRIMIDCNTLEN: 1
- RXPRIMANTID: 0

6.4 Baseband Controller Interface Registers

Register Name	Register Description	Address
BBIF_CTRL	Baseband controller control register	0x40001800
BBIF_STATUS	Baseband controller status register	0x40001804
BBIF_COEX_CTRL	Coexistence control register	0x40001808
BBIF_COEX_STATUS	Coexistence status register	0x4000180C
BBIF_COEX_INT_CFG	Coexistence interrupts configuration register	0x40001810
BBIF_COEX_INT_STATUS	Coexistence interrupt status register	0x40001814
BBIF_ID_NUM	Baseband controller interface ID number	0x400018FC

6.4.0.1 BBIF_CTRL

Bit Field	Read/Write	Field Name	Description
9:4	RW	CLK_SEL	Configure the internal baseband controller clock divider in order to provide a 1MHz reference clock
1	RW	WAKEUP_REQ	External wake up request used to sort-out sleep modes
0	RW	CLK_ENABLE	Enable the baseband controller clocks generation

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
9:4	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
1	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
0	CLK_ENABLE	BB_CLK_DISABLE	Baseband controller clocks are gated	0x0*
		BB_CLK_ENABLE	Baseband controller clocks are generated	0x1

6.4.0.2 BBIF_STATUS

Bit Field	Read/Write	Field Name	Description
16:12	R	LINK_FORMAT	BLE link format
8:4	R	LINK_LABEL	BLE link label
2	R	CLK_STATUS	Clock status defining the current active clock in use
1	R	OSC_EN	Oscillator front-end enabling
0	R	RADIO_EN	RF front-end enabling

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
2	CLK_STATUS	MASTER_CLK	Baseband controller running on master1/2_clk	0x0*
		LOW_POWER_CLK	Baseband controller running on low_power_clk	0x1
1	OSC_EN	OSC_DISABLED	Oscillator can be safely disabled	0x0
		OSC_ENABLED	Oscillator must be enabled	0x1*
0	RADIO_EN	RF_DISABLED	RF front-end can be safely disabled	0x0
		RF_ENABLED	RF front-end must be enabled	0x1*

6.4.0.3 BBIF_COEX_CTRL

Bit Field	Read/Write	Field Name	Description
1	RW	TX	WLAN collocated device transmit active
0	RW	RX	WLAN collocated device reception active

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
1	TX	COEX_TX_IDLE	RF front-end has no non-BLE TX activity	0x0*
		COEX_TX_BUSY	RF front-end performs a non-BLE TX activity	0x1
0	RX	COEX_RX_IDLE	RF front-end has no non-BLE RX activity	0x0*
		COEX_RX_BUSY	RF front-end performs a non-BLE RX activity	0x1

6.4.0.4 BBIF_COEX_STATUS

Bit Field	Read/Write	Field Name	Description
11:8	R	BLE_PTI	BLE packet traffic information
4	R	BLE_SYNC	BLE 625us timing pulse reference
3	R	EVENT_IN_PROCESS	BLE event status
2	R	BLE_IN_PROCESS	BLE in process indicator
1	R	BLE_TX	BLE transmit indicator
0	R	BLE_RX	BLE reception indicator

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
11:8	BLE_PTI	BLE_PTI_PRIORITY_0	BLE_PTI lowest priority	0x0*
		BLE_PTI_PRIORITY_15	BLE_PTI highest priority	0xF
4	BLE_SYNC	BLE_SYNC_IDLE	RW-BLE sync idle	0x0*
		BLE_SYNC_BUSY	RW-BLE sync active	0x1
3	EVENT_IN_PROCESS	EVENT_IDLE	No BLE event is processed	0x0*
		EVENT_IN_PROCESS	BLE event is processed	0x1
2	BLE_IN_PROCESS	BLE_IDLE	RW-BLE processes no event	0x0*
		BLE_IN_PROCESS	RW-BLE processes an event	0x1
1	BLE_TX	BLE_TX_IDLE	RW-BLE core performs no Tx activity	0x0*
		BLE_TX_BUSY	RW-BLE core performs Tx activity	0x1
0	BLE_RX	BLE_RX_IDLE	RW-BLE core performs no Rx activity	0x0*
		BLE_RX_BUSY	RW-BLE core performs Rx activity	0x1

6.4.0.5 BBIF_COEX_INT_CFG

Bit Field	Read/Write	Field Name	Description
7:6	RW	EVENT_IN_PROCESS	EVENT_IN_PROCESS event interrupt configuration
5:4	RW	BLE_IN_PROCESS	BLE_IN_PROCESS event interrupt configuration
3:2	RW	BLE_TX	BLE_TX event interrupt configuration
1:0	RW	BLE_RX	BLE_RX event interrupt configuration

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
7:6	EVENT_IN_PROCESS	EVENT_IN_PROCESS_NONE	Interrupt not triggered	0x0*
		EVENT_IN_PROCESS_RISING_EDGE	Interrupt triggered on rising edge	0x1
		EVENT_IN_PROCESS_FALLING_EDGE	Interrupt triggered on falling edge	0x2
		EVENT_IN_PROCESS_TRANSITION	Interrupt triggered on any edge	0x3
5:4	BLE_IN_PROCESS	BLE_IN_PROCESS_NONE	Interrupt not triggered	0x0*
		BLE_IN_PROCESS_RISING_EDGE	Interrupt triggered on rising edge	0x1
		BLE_IN_PROCESS_FALLING_EDGE	Interrupt triggered on falling edge	0x2

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		BLE_IN_PROCESS_TRANSITION	Interrupt triggered on any edge	0x3
3:2	BLE_TX	BLE_TX_NONE	Interrupt not triggered	0x0*
		BLE_TX_RISING_EDGE	Interrupt triggered on rising edge	0x1
		BLE_TX_FALLING_EDGE	Interrupt triggered on falling edge	0x2
		BLE_TX_TRANSITION	Interrupt triggered on any edge	0x3
1:0	BLE_RX	BLE_RX_NONE	Interrupt not triggered	0x0*
		BLE_RX_RISING_EDGE	Interrupt triggered on rising edge	0x1
		BLE_RX_FALLING_EDGE	Interrupt triggered on falling edge	0x2
		BLE_RX_TRANSITION	Interrupt triggered on any edge	0x3

6.4.0.6 BBIF_COEX_INT_STATUS

Bit Field	Read/Write	Field Name	Description
11	R	EVENT_IN_PROCESS	EVENT_IN_PROCESS interrupt status flag
10	R	BLE_IN_PROCESS	BLE_IN_PROCESS interrupt status flag
9	R	BLE_TX	BLE_TX interrupt status flag
8	R	BLE_RX	BLE_RX interrupt status flag
3	W	EVENT_IN_PROCESS_CLEAR	Clear EVENT_IN_PROCESS status flag
2	W	BLE_IN_PROCESS_CLEAR	Clear BLE_IN_PROCESS status flag
1	W	BLE_TX_CLEAR	Clear BLE_TX status flag
0	W	BLE_RX_CLEAR	Clear BLE_RX status flag

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
11	EVENT_IN_PROCESS	EVENT_IN_PROCESS_NO_INT	No EVENT_IN_PROCESS_EVENT interrupt has been generated	0x0*
		EVENT_IN_PROCESS_INT	A EVENT_IN_PROCESS_EVENT interrupt has been generated	0x1
10	BLE_IN_PROCESS	BLE_IN_PROCESS_NO_INT	No BLE_IN_PROCESS_EVENT interrupt has been generated	0x0*
		BLE_IN_PROCESS_INT	A BLE_IN_PROCESS_EVENT interrupt has been generated	0x1
9	BLE_TX	BLE_TX_NO_INT	No BLE_TX_EVENT interrupt has been generated	0x0*
		BLE_TX_INT	A BLE_TX_EVENT interrupt has been	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
			generated	
8	BLE_RX	BLE_RX_NO_INT	No BLE_RX_EVENT interrupt has been generated	0x0*
		BLE_RX_INT	A BLE_RX_EVENT interrupt has been generated	0x1
3	EVENT_IN_PROCESS_ CLEAR	EVENT_IN_PROCESS_CLEAR		0x1
2	BLE_IN_PROCESS_ CLEAR	BLE_IN_PROCESS_CLEAR		0x1
1	BLE_TX_CLEAR	BLE_TX_CLEAR		0x1
0	BLE_RX_CLEAR	BLE_RX_CLEAR		0x1

6.4.0.7 BBIF_ID_NUM

Bit Field	Read/Write	Field Name	Description
15:8	R	BBIF_MAJOR_REVISION	Baseband controller interface Major Revision number
7:0	R	BBIF_MINOR_REVISION	Baseband controller interface Minor Revision number

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
15:8	BBIF_MAJOR_ REVISION	BBIF_MAJOR_REVISION	Baseband controller interface revision 1.2	0x1*
7:0	BBIF_MINOR_REVISION	BBIF_MINOR_REVISION	Baseband controller interface revision 1.2	0x2*

6.5 Baseband Timer Registers

Register Name	Register Description	Address
ACS_BB_TIMER_CTRL	Baseband timer and standby clock control register	0x40001B54

6.5.0.1 ACS_BB_TIMER_CTRL

Bit Field	Read/Write	Field Name	Description
0	RW	BB_TIMER_NRESET	nReset signal for the baseband timer

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
0	BB_TIMER_NRESET	BB_TIMER_RESET	Baseband timer is in reset state	0x0*
		BB_TIMER_NRESET	Baseband timer reset is released	0x1

6.6 Baseband Controller Registers

Register Name	Register Description	Address
BB_RWBBCNTL	Baseband control register	0x40001900
BB_VERSION	BLE revision register	0x40001904
BB_RWBLEBCONF	Baseband configuration register (compilation options dependant)	0x40001908
BB_INTCNTL0	Interrupts control register 0	0x4000190C
BB_INTSTAT0	Interrupts status register 0	0x40001910
BB_INTACK0	Interrupts raw status register 0	0x40001914
BB_INTCNTL1	Interrupts control register 1	0x40001918
BB_INTSTAT1	Interrupts status register 1	0x4000191C
BB_INTACK1	Interrupts acknowledgement register 1	0x40001920
BB_ACTFIFOSTAT	Actif FIFO status register	0x40001924
BB_CURRENTRXDESCPTR	Rx descriptor pointer register	0x40001928
BB_ETPR	Rx descriptor pointer register	0x4000192C
BB_DEEPSLCNTL	Deep sleep control register	0x40001930
BB_DEEPSLWKUP	Deep sleep wakeup register	0x40001934
BB_DEEPSLSTAT	Deep sleep status register	0x40001938
BB_ENBPRESET	Stabilization times	0x4000193C
BB_FINECNTCORR	Fine timer correction register	0x40001940
BB_CLKNCNTCORR	Slot clock correction register	0x40001944
BB_DIAGCNTL	Diagnostic ports control register	0x40001950
BB_DIAGSTAT	Diagnostic ports status register	0x40001954
BB_DEBUGADDMAX	Diagnostic ports upper limit	0x40001958
BB_DEBUGADDMIN	Diagnostic ports lower limit	0x4000195C
BB_ERRORTYPESTAT	Diagnostic ports errors register	0x40001960
BB_SWPROFILING	Software profiling register	0x40001964
BB_RADIOCNTL0	Principal control register for the radio interface	0x40001970
BB_RADIOCNTL1	Second control register for the radio interface	0x40001974
BB_RADIOCNTL2	Third control register for the radio interface	0x40001978
BB_RADIOCNTL3	Fourth control register for the radio interface	0x4000197C
BB_RADIOPWRUPDN0	Principal control register for the radio interface power up/down delays	0x40001980

Register Name	Register Description	Address
BB_RADIOPWRUPDN1	Second control register for the radio interface power up/down delays	0x40001984
BB_RADIOPWRUPDN2	Third control register for the radio interface power up/down delays (only when LR is instantiated)	0x40001988
BB_RADIOPWRUPDN3	Fourth control register for the radio interface power up/down delays (only when LR is instantiated)	0x4000198C
BB_RADIOTXRXTIM0	Principal control register for the radio interface timing compensation delays	0x40001990
BB_RADIOTXRXTIM1	Second control register for the radio interface timing compensation delays	0x40001994
BB_RADIOTXRXTIM2	Third control register for the radio interface timing compensation delays (only when LR is instantiated)	0x40001998
BB_RADIOTXRXTIM3	Fourth control register for the radio interface timing compensation delays (only when LR is instantiated)	0x4000199C
BB_SPIPTRCNTL0	First control register for the radio interface SPI pointers	0x400019A0
BB_SPIPTRCNTL1	Second control register for the radio interface SPI pointers	0x400019A4
BB_SPIPTRCNTL2	Third control register for the radio interface SPI pointers	0x400019A8
BB_SPIPTRCNTL3	Fourth control register for the radio interface SPI pointers	0x400019AC
BB_AESCNTL	AES-128 ciphering control register	0x400019B0
BB_AESKEY31_0	AES encryption 128-bit key register (bits 31:0)	0x400019B4
BB_AESKEY63_32	AES encryption 128-bit key register (bits 63:32)	0x400019B8
BB_AESKEY95_64	AES encryption 128-bit key register (bits 95:64)	0x400019BC
BB_AESKEY127_96	AES encryption 128-bit key register (bits 127:96)	0x400019C0
BB_AESPTR	AES memory zone pointer	0x400019C4
BB_TXMICVAL	AES-CCM plain MIC value register in Tx	0x400019C8
BB_RXMICVAL	AES-CCM plain MIC value register in Rx	0x400019CC
BB_RFTESTCNTL	RF testing and regulatory body support register	0x400019D0
BB_RFTESTTXSTAT	Number of transmitted packet during test modes	0x400019D4
BB_RFTESTRXSTAT	Number of correctly received packet during test modes	0x400019D8
BB_TIMGENCNTL	Timing generator control register	0x400019E0
BB_FINETIMTGT	Fine timer control register	0x400019E4
BB_CLKNTGT1	CLKN target value 1	0x400019E8
BB_HMICROSECTGT1	Half microsecond target value 1	0x400019EC

Register Name	Register Description	Address
BB_CLKNTGT2	CLKN target value 2	0x400019F0
BB_HMICROSECTGT2	Half microsecond target value 2	0x400019F4
BB_SLOTCLK	Value of the 312.5us CLKN counter	0x400019F8
BB_FINETIMECNT	Value of the current half us fine time reference counter	0x400019FC
BB_ACTSCHCNTL	Value of the 312.5us CLKN counter	0x40001A00
BB_STARTEVTCLKNTS	Value of the CLKN counter when ble_start_int is generated	0x40001A04
BB_STARTEVTFINECNTTS	Value of the fine counter when ble_start_int is generated	0x40001A08
BB_ENDEVTCLKNTS	Value of the CLKN counter when ble_end_int is generated	0x40001A0C
BB_ENDEVTFINECNTTS	Value of the fine counter when ble_end_int is generated	0x40001A10
BB_SKIPEVTCLKNTS	Value of the CLKN counter when ble_skip_int is generated	0x40001A14
BB_SKIPEVTFINECNTTS	Value of the fine counter when ble_skip_int is generated	0x40001A18
BB_ADVTIM	Delay information register handling advertising event timers	0x40001A20
BB_ACTSCANCNTL	Active scan mode control register	0x40001A24
BB_WPALCNTL	Devices in white list	0x40001A30
BB_WPALCURRENPTR	Current pointer in use for the White List	0x40001A34
BB_SEARCH_TIMEOUT	RAL and List Search engines timeout delay in us	0x40001A38
BB_COEXIFCNTL0	WLAN coexistence control register 0	0x40001A40
BB_COEXIFCNTL1	WLAN coexistence control register 1	0x40001A44
BB_COEXIFCNTL2	WLAN coexistence control register 2	0x40001A48
BB_BLEMPRIO0	Priority control register 0	0x40001A4C
BB_BLEMPRIO1	Priority control register 1	0x40001A50
BB_BLEMPRIO2	Priority control register 2	0x40001A54
BB_RALCNTL	Control of the Resolving Address List engine	0x40001A60
BB_RALCURRENTPTR	Current pointer of the RAL structure	0x40001A64
BB_RAL_LOCAL_RND	Register used by the local Resolving Address List engine	0x40001A68
BB_RAL_PEER_RND	Register used by the peer Resolving Address List engine	0x40001A6C
BB_DFCNTL0_1US	AoA/AOD control register 0 (1us)	0x40001A70
BB_DFCNTL0_2US	AoA/AOD control register 0 (2us)	0x40001A74
BB_DFCNTL1_1US	AoA/AOD control register 1 (1us)	0x40001A78
BB_DFCNTL1_2US	AoA/AOD control register 1 (2us)	0x40001A7C

Register Name	Register Description	Address
BB_DFCURRENTPTR	Rx CTE descriptor current pointer	0x40001A80
BB_DFANTCNTL	AoA/AOD antenna control register	0x40001A84
BB_DFIFCNTL	AoA/AOD interface control register	0x40001A88

6.6.0.1 BB_RWBBCNTL

Bit Field	Read/Write	Field Name	Description
31	RW	MASTER_SOFT_RST	Reset the complete system except registers and timing generator
30	RW	MASTER_TGSOFT_RST	Reset the timing generator
29	RW	REG_SOFT_RST	Reset the complete register block
28	RW	RADIOCNTL_SOFT_RST	Reset the radio controller
27	RW	SWINT_REQ	Force the generation of ble_sw_irq
26	RW	RFTEST_ABORT	Abort the current RF testing defined as per CS-FORMAT
25	RW	ADVERT_ABORT	Abort the current scan window
24	RW	SCAN_ABORT	Abort the current advertising event
20	RW	MD_DSB	Allow a single Tx/Rx exchange whatever the MD bits are
19	RW	SN_DSB	Disable sequence number management
18	RW	NESN_DSB	Disable acknowledge scheme
17	RW	CRYPT_DSB	Disable encryption/decryption
16	RW	LRPMAP_DSB	LR pattern mapper/demapper enabled (has effect only if RW_BLE_LONG_RANGE_INST is defined)
15	RW	LRFEC_DSB	LR FEC encoder/decoder enabled (has effect only if RW_BLE_LONG_RANGE_INST is defined)
14	RW	WHIT_DSB	Disable whitening
13	RW	CRC_DSB	Disable CRC stripping
12	RW	HOP_REMAP_DSB	Disable frequency hopping remapping algorithm
11	RW	RXCTEERR_RETX_EN	Rx CTE error detection
10	RW	ANONYMOUS_ADVERT_FILT_EN	Anonymous extended advertising filtering enable control (operate in extended active scanner and extended passive scanner modes only, and when white list is used by device filtering policy)
9	RW	ADVERTFILT_EN	Advertising channels error filtering enable control
8	RW	RWBLE_EN	Enable RW-BLE core exchange table pre-fetch mechanism
3:0	RW	RXWINSZDEF	Default Rx window size in us (used when device is master connected or performs its second receipt)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31	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
30	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
29	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
28	RADIOCNTL_SOFT_RST	RADIOCNTL_SOFT_RST_0	No action happens if it is written with 0	0x0*
		RADIOCNTL_SOFT_RST_1	Reset the radio controller, when written with a 1	0x1
27	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
26	RFTEST_ABORT	RFTEST_ABORT_0	No action happens if it is written with 0	0x0*
		RFTEST_ABORT_1	Abort the current RF testing	0x1
25	ADVERT_ABORT	ADVERT_ABORT_0	No action happens if it is written with 0	0x0*
		ADVERT_ABORT_1	Abort the current scan window	0x1
24	SCAN_ABORT	SCAN_ABORT_0	No action happens if it is written with 0	0x0*
		SCAN_ABORT_1	Abort the current advertising event	0x1
20	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
19	SN_DSB	SN_DSB_0	Normal operation of sequence number	0x0*
		SN_DSB_1	Sequence number management disabled	0x1
18	NESN_DSB	NESN_DSB_0	Normal operation of acknowledge	0x0*
		NESN_DSB_1	Acknowledge scheme disabled	0x1
17	CRYPT_DSB	CRYPT_DSB_0	Normal operation (encryption / decryption enabled)	0x0*
		CRYPT_DSB_1	Encryption / decryption disabled	0x1
16	LRPMAP_DSB	LRPMAP_DSB_0	Normal operation. LR Pattern Mapper/Demapper enabled	0x0*
		LRPMAP_DSB_1	LR Pattern Mapper/Demapper disabled.	0x1
15	LRFEC_DSB	LRFEC_DSB_0	Normal operation. LR FEC	0x0*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
			encoder/decoder enabled	
		LRFEC_DSB_1	LR FEC encoder/decoder disabled.	0x1
14	WHIT_DSB	WHIT_DSB_0	Normal operation (whitening enabled)	0x0*
		WHIT_DSB_1	Whitening disabled	0x1
13	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
12	HOP_REMAP_DSB	HOP_REMAP_DSB_0	Normal operation (requency hopping remapping algorithm enabled)	0x0*
		HOP_REMAP_DSB_1	Frequency hopping remapping algorithm disabled	0x1
11	RXCTEERR_RETX_EN	RX_CTE_ERROR_DETECTION_0	RW-BLE Core does not reports anonymous advertiser to RW-BLE Software	0x0*
		RX_CTE_ERROR_DETECTION_1	RW-BLE Core reports anonymous advertisers to RW-BLE Software	0x1
10	ANONYMOUS_ADVERT_FILT_EN	ANONYMOUS_ADVERT_FILT_EN_0	RW-BLE Core does not reports anonymous advertiser to RW-BLE Software	0x0*
		ANONYMOUS_ADVERT_FILT_EN_1	RW-BLE Core reports anonymous advertisers to RW-BLE Software	0x1
9	ADVERTFILT_EN	ADVERTFILT_EN_0	RW-BLE core reports all errors to RW-BLE software	0x0*
		ADVERTFILT_EN_1	RW-BLE core reports only correctly received packet, without error to RW-BLE software	0x1
8	RWBLE_EN	RWBLE_EN_0	Disable RW-BLE core exchange table pre-fetch mechanism	0x0*
		RWBLE_EN_1	Enable RW-BLE core exchange table pre-fetch mechanism	0x1
3:0	RXWINSZDEF	RXWINSZDEF_0	Default Rx half Window size in us for uncoded PHY, and in multiple of 2us for coded PHY	0x0*

6.6.0.2 BB_VERSION

Bit Field	Read/Write	Field Name	Description
31:24	R	TYP	RW-BLE core type
23:16	R	REL	RW-BLE core version - major release number
15:8	R	UPG	RW-BLE core upgrade - upgrade number
7:0	R	BUILD	RW-BLE core build - build number

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:24	TYP	TYP_A	BLE v5.1 Madrid release	0xA*
23:16	REL	REL_0		0x0*
15:8	UPG	UPG_11		0x11*
7:0	BUILD	BUILD_0		0x0*

6.6.0.3 BB_RWBLEBCONF

Bit Field	Read/Write	Field Name	Description
31	R	DMMODE	RW-BLE core dual mode
29	R	WLANCOEX	WLAN coexistence mechanism
28	R	CORRELATOR	Correlator present
27	R	USERXLR	Long range Rx present
26	R	USETXLR	Long range Tx present
24	R	USEISO	Support of isochronous channels
22:16	R	RFIF	Support of the RF front-end
15	R	USEDBG	Diagnostic port
14	R	DECIPHER	AES deciphering present
13:8	R	CLK_SEL	Operating frequency (in MHz)
7	R	INTMODE	Interruption mode
6	R	BUSTYPE	Processor bus type
4:0	R	ADD_WIDTH	Value of the RW_BLE_ADDRESS_WIDTH parameter concerted into binary

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31	DMMODE	DMMODE_0	RW-BLE core is used as a standalone BLE device	0x0*
		DMMODE_1	RW-BLE core is used in a dual mode device	0x1
29	WLANCOEX	COEX_0	WLAN coexistence mechanism not present	0x0
		COEX_1	WLAN coexistence mechanism present	0x1*
28	CORRELATOR	CORRELATOR_0	Correlator logic is not present	0x0*
		CORRELATOR_1	Correlator logic is present	0x1
27	USERXLR	USERXLR_0	Long Range receive bit-stream logic is not present	0x0*
		USERXLR_1	Long Range receive bit-stream logic is present	0x1
26	USETXLR	USETXLR_0	Long Range transmit bit-stream logic is not present	0x0
		USETXLR_1	Long Range transmit bit-stream logic is present	0x1*
24	USEISO	USEISO_0	Isochronous channel feature not supported	0x0*
		USEISO_1	Isochronous channel feature supported	0x1
22:16	RFIF	RFIF_RIPPLE	Ripple and Thesis RF	0x1
		RFIF_EXT	External radio controller support	0x2
		RFIFCALYPSO	Calypso radio	0x4
		RFIF_ICYTRX_V2	IcyTRx version 2 radio	0x8*
		RFIF_BTIPT	BTIPT radio	0x10
		RFIF_AURA_SEMI	Aura semi 50xx radio	0x20
		RFIF_RESERVED	Reserved	0x40
15	USEDBG	USEDBG_0	WLAN coexistence mechanism present	0x0
		USEDBG_1	Diagnostic port instantiated	0x1*
14	DECIPHER	DECIPHER_0	AES deciphering not present	0x0*
		DECIPHER_1	AES deciphering present	0x1
13:8	CLK_SEL	CLK_SEL_8	Default value is 8MHz	0x8*
7	INTMODE	INTMODE_0	Interrupts are edge level generated, i.e. pulse	0x0*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		INTMODE_1	Interrupts are trigger level generated, i.e. stays active at 1 till acknowledgement	0x1
6	BUSTYPE	BUSTYPE_0	Processor bus type: AHB bus	0x0
		BUSTYPE_1	Processor bus type: XBAR bus	0x1*
4:0	ADD_WIDTH	ADD_WIDTH_14	EM size is 16kB	0xE*

6.6.0.4 BB_INTCNTL0

Bit Field	Read/Write	Field Name	Description
16	RW	ERRORINTMSK	Error interrupt mask
6	RW	ISORXINTMSK	Isochronous channel Rx interrupt mask
5	RW	ISOTXINTMSK	Isochronous channel Tx interrupt mask
4	RW	RXINTMSK	Rx interrupt mask
3	RW	TXINTMSK	Tx interrupt mask
2	RW	SKIPEVTINTMSK	Skipped event interrupt mask
1	RW	ENDEVTINTMSK	End of event interrupt mask
0	RW	STARTEVTINTMSK	Start of event interrupt mask

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
16	ERRORINTMSK	ERRORINTMSK_0	Interrupt not generated	0x0*
		ERRORINTMSK_1	Interrupt generated	0x1
6	ISORXINTMSK	ISORXINTMSK_0	Interrupt not reported in FIFO IRQ	0x0*
		ISORXINTMSK_1	Interrupt reported in FIFO IRQ	0x1
5	ISOTXINTMSK	ISOTXINTMSK_0	Interrupt not reported in FIFO IRQ	0x0*
		ISOTXINTMSK_1	Interrupt reported in FIFO IRQ	0x1
4	RXINTMSK	RXINTMSK_0	Interrupt not reported in FIFO IRQ	0x0*
		RXINTMSK_1	Interrupt reported in FIFO IRQ	0x1
3	TXINTMSK	TXINTMSK_0	Interrupt not reported in FIFO IRQ	0x0*
		TXINTMSK_1	Interrupt reported in FIFO IRQ	0x1
2	SKIPEVTINTMSK	SKIPEVTINTMSK_0	Interrupt not reported in FIFO IRQ	0x0*
		SKIPEVTINTMSK1	Interrupt reported in FIFO IRQ	0x1
1	ENDEVTINTMSK	ENDEVTINTMSK_0	Interrupt not reported in FIFO IRQ	0x0

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		ENDEVTINTMSK_1	Interrupt reported in FIFO IRQ	0x1*
0	STARTEVTINTMSK	STARTEVTINTMSK_0	Interrupt not reported in FIFO IRQ	0x0
		STARTEVTINTMSK_1	Interrupt reported in FIFO IRQ	0x1*

6.6.0.5 BB_INTSTAT0

Bit Field	Read/Write	Field Name	Description
16	R	ERRORINTSTAT	Error interrupt status

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
16	ERRORINTSTAT	ERRORINTSTAT_0	No error interrupt	0x0*
		ERRORINTSTAT_1	An error interrupt is pending	0x1

6.6.0.6 BB_INTACK0

Bit Field	Read/Write	Field Name	Description
16	RW	ERRORINTACK	Error interrupt acknowledgement

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
16	ERRORINTACK	ERRORINTACK_0	No error interrupt	0x0*
		ERRORINTACK_1	An error interrupt is pending	0x1

6.6.0.7 BB_INTCNTL1

Bit Field	Read/Write	Field Name	Description
30:28	RW	CLKNINTSRMSK	CLKN/half-slot interrupt sub rating mask (valid range is [0:4])
27:24	RW	CLKNINTSRVAL	CLKN/half-slot interrupt sub rating value
15	RW	FIFOINTMSK	FIFO interrupt mask
6	RW	TIMESTAMPTGT2INTMSK	Time stamp target timer 2 interrupt mask
5	RW	TIMESTAMPTGT1INTMSK	Time stamp target timer 1 interrupt mask
4	RW	FINETGTIMINTMSK	Fine target timer interrupt mask
3	RW	SWINTMSK	SW triggered interrupt mask
2	RW	CRYPTINTMSK	Encryption engine interrupt mask
1	RW	SLPINTMSK	Sleep mode interrupt mask
0	RW	CLKNINTMSK	CLKN/half slot interrupt mask

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
30:28	CLKNINTSRMSK	CLKNINTSRMSK_0		0x0*
27:24	CLKNINTSRVAL	CLKNINTSRVAL_0		0x0*
15	FIFOINTMSK	FIFOINTMSK_0	Interrupt not generated	0x0
		FIFOINTMSK_1	Interrupt generated	0x1*
6	TIMESTAMPTGT2INTMSK	TIMESTAMPTGT2INTMSK_0	Interrupt not generated	0x0*
		TIMESTAMPTGT2INTMSK_1	Interrupt generated	0x1
5	TIMESTAMPTGT1INTMSK	TIMESTAMPTGT1INTMSK_0	Interrupt not generated	0x0*
		TIMESTAMPTGT1INTMSK_1	Interrupt generated	0x1
4	FINETGTIMINTMSK	FINETGTIMINTMSK_0	Interrupt not generated	0x0*
		FINETGTIMINTMSK_1	Interrupt generated	0x1
3	SWINTMSK	SWINTMSK_0	Interrupt not generated	0x0*
		SWINTMSK_1	Interrupt generated	0x1
2	CRYPTINTMSK	CRYPTINTMSK_0	Interrupt not generated	0x0*
		CRYPTINTMSK_1	Interrupt generated	0x1
1	SLPINTMSK	SLPINTMSK_0	Interrupt not generated	0x0
		SLPINTMSK_1	Interrupt generated	0x1*
0	CLKNINTMSK	CLKNINTMSK_0	Interrupt not generated	0x0
		CLKNINTMSK_1	Interrupt generated	0x1*

6.6.0.8 BB_INTSTAT1

Bit Field	Read/Write	Field Name	Description
15	R	FIFOINTSTAT	FIFO interrupt status
6	R	TIMESTAMPTGT2INTSTAT	Time stamp target timer 2 interrupt status
5	R	TIMESTAMPTGT1INTSTAT	Time stamp target timer 1 interrupt status
4	R	FINETGTIMINTSTAT	Fine target timer interrupt status
3	R	SWINTSTAT	SW triggered interrupt status
2	R	CRYPTINTSTAT	Encryption engine interrupt status
1	R	SLPINTSTAT	Sleep mode interrupt status
0	R	CLKNINTSTAT	CLKN/half slot interrupt status

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
15	FIFOINTSTAT	FIFOINTSTAT_0	No FIFO interrupt	0x0*
		FIFOINTSTAT_1	A FIFO interrupt is pending	0x1
6	TIMESTAMPTGT2INTSTAT	TIMESTAMPTGT2INTSTAT_0	No fine target timer 2 interrupt	0x0*
		TIMESTAMPTGT2INTSTAT_1	A fine target timer 2 interrupt is pending	0x1
5	TIMESTAMPTGT1INTSTAT	TIMESTAMPTGT1INTSTAT_0	No fine target timer 1 interrupt	0x0*
		TIMESTAMPTGT1INTSTAT_1	A fine target timer 1 interrupt is pending	0x1
4	FINETGTIMINTSTAT	FINETGTIMINTSTAT_0	No fine target timer interrupt	0x0*
		FINETGTIMINTSTAT_1	A fine target timer interrupt is pending	0x1
3	SWINTSTAT	SWINTSTAT_0	No SW triggered interrupt	0x0*
		SWINTSTAT_1	A SW triggered interrupt is pending	0x1
2	CRYPTINTSTAT	CRYPTINTSTAT_0	No encryption / decryption interrupt	0x0*
		CRYPTINTSTAT_1	An encryption / decryption interrupt is pending	0x1
1	SLPINTSTAT	SLPINTSTAT_0	No end of sleep mode interrupt	0x0*
		SLPINTSTAT_1	An end of sleep mode interrupt is pending	0x1
0	CLKNINTSTAT	CLKNINTSTAT_0	No half slot interrupt	0x0*
		CLKNINTSTAT_1	A half slot interrupt is pending	0x1

6.6.0.9 BB_INTACK1

Bit Field	Read/Write	Field Name	Description
15	RW	FIFOINTACK	FIFO interrupt acknowledgement
6	RW	TIMESTAMPTGT2INTACK	Time stamp target timer 2 interrupt acknowledgement
5	RW	TIMESTAMPTGT1INTACK	Time stamp target timer 1 interrupt acknowledgement
4	RW	FINETGTIMINTACK	Fine target timer interrupt acknowledgement
3	RW	SWINTACK	SW triggered interrupt acknowledgement
2	RW	CRYPTINTACK	Encryption engine interrupt acknowledgement
1	RW	SLPINTACK	Sleep mode interrupt acknowledgement
0	RW	CLKNINTACK	CLKN/half slot interrupt acknowledgement

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
15	FIFOINTACK	FIFOINTACK_0		0x0*
		FIFOINTACK_1	Acknowledge the FIFO interrupt	0x1
6	TIMESTAMPTGT2INTACK	TIMESTAMPTGT2INTACK_0		0x0*
		TIMESTAMPTGT2INTACK_1	Acknowledge the time stamp target timer 2 interrupt	0x1
5	TIMESTAMPTGT1INTACK	TIMESTAMPTGT1INTACK_0		0x0*
		TIMESTAMPTGT1INTACK_1	Acknowledge the time stamp target timer 1 interrupt	0x1
4	FINETGTIMINTACK	FINETGTIMINTACK_0		0x0*
		FINETGTIMINTACK_1	Acknowledge the fine timer interrupt	0x1
3	SWINTACK	SWINTACK_0		0x0*
		SWINTACK_1	Acknowledge the SW triggered timer interrupt	0x1
2	CRYPTINTACK	CRYPTINTACK_0		0x0*
		CRYPTINTACK_1	Acknowledge the encryption timer interrupt	0x1
1	SLPINTACK	SLPINTACK_0		0x0*
		SLPINTACK_1	Acknowledge the end of sleep interrupt	0x1
0	CLKNINTACK	CLKNINTACK_0		0x0*
		CLKNINTACK_1	Acknowledge the half slot interrupt	0x1

6.6.0.10 BB_ACTFIFOSTAT

Bit Field	Read/Write	Field Name	Description
31:28	R	SKIP_ET_IDX	Exchange table entry index of the reported skipped event (valid when SKIPACTINTSTAT is set)
27:24	R	CURRENT_ET_IDX	Exchange table entry index of the reported current event (valid for any set reported interrupt except SKIPACTINTSTAT)
15	R	ACTFLAG	Forced to 1 in BLE
6	R	ISORXINTSTAT	Isochronous channel Rx interrupt status
5	R	ISOTXINTSTAT	Isochronous channel Tx interrupt status
4	R	RXINTSTAT	Rx interrupt status
3	R	TXINTSTAT	Tx interrupt status

Bit Field	Read/Write	Field Name	Description
2	R	SKIPACTINTSTAT	Skipped event interrupt status
1	R	ENDACTINTSTAT	End of event interrupt status
0	R	STARTACTINTSTAT	Start of event interrupt status

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:28	SKIP_ET_IDX	SKIP_ET_IDX_0		0x0*
27:24	CURRENT_ET_IDX	CURRENT_ET_IDX_0		0x0*
15	ACTFLAG	ACTFLAG_0		0x0*
		ACTFLAG_1		0x1
6	ISORXINTSTAT	ISORXINTSTAT_0	No isochronous channel Rx interrupt	0x0*
		ISORXINTSTAT_1	An isochronous channel Rx interrupt is pending	0x1
5	ISOTXINTSTAT	ISOTXINTSTAT_0	No isochronous channel Tx interrupt.	0x0*
		ISOTXINTSTAT_1	An isochronous channel Tx interrupt is pending	0x1
4	RXINTSTAT	RXINTSTAT_0	No Rx interrupt.	0x0*
		RXINTSTAT_1	An Rx interrupt is pending	0x1
3	TXINTSTAT	TXINTSTAT_0	No Tx interrupt.	0x0*
		TXINTSTAT_1	A Tx interrupt is pending	0x1
2	SKIPACTINTSTAT	SKIPACTINTSTAT_0	No skipped event interrupt	0x0*
		SKIPACTINTSTAT_1	A skipped event interrupt is pending	0x1
1	ENDACTINTSTAT	ENDACTINTSTAT_0	No end of event interrupt	0x0*
		ENDACTINTSTAT_1	An end of event interrupt is pending	0x1
0	STARTACTINTSTAT	STARTACTINTSTAT_0	No start of event interrupt	0x0*
		STARTACTINTSTAT_1	A start of event interrupt is pending	0x1

6.6.0.11 BB_CURRENTRXDESCPTR

Bit Field	Read/Write	Field Name	Description
13:0	RW	CURRENTRXDESCPTR	Rx descriptor pointer that determines the starting point of the receive buffer chained list

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
13:0	CURRENTRXDESCPTR	CURRENTRXDESCPTR_0		0x0*

6.6.0.12 BB_ETPR

Bit Field	Read/Write	Field Name	Description
13:0	RW	ETPTR	Exchange table pointer that determines the starting point of the exchange table

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
13:0	ETPTR	ETPTR_0		0x0*

6.6.0.13 BB_DEEPSLCNTL

Bit Field	Read/Write	Field Name	Description
31	RW	EXTWKUPDSB	External wake-up disable
15	R	DEEP_SLEEP_STAT	Indicator of current deep sleep clock mux status
3	RW	DEEP_SLEEP_CORR_EN	Half slot counter integer and fractional part correction (apply when system has been woken-up from deep sleep mode)
2	RW	DEEP_SLEEP_ON	RW-BLE core power mode control
1	RW	RADIO_SLEEP_EN	Control the radio module
0	RW	OSC_SLEEP_EN	Control the RF high frequency crystal oscillator

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31	EXTWKUPDSB	EXTWKUPDSB_0	RW-BLE core can be woken by external wake-up	0x0*
		EXTWKUPDSB_1	RW-BLE core cannot be woken up by external wake-up	0x1
15	DEEP_SLEEP_STAT	DEEP_SLEEP_STAT_0	RW-BLE core is not yet in deep sleep mode	0x0*
		DEEP_SLEEP_STAT_1	RW-BLE core is in deep sleep mode (only low_power_clk is running)	0x1
3	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
2	DEEP_SLEEP_ON	DEEP_SLEEP_ON_0	RW-BLE core in normal active mode	0x0*
		DEEP_SLEEP_ON_1	Request RW-BLE core to switch in deep sleep mode	0x1
1	RADIO_SLEEP_EN	RADIO_SLEEP_EN_0	Radio stands in normal active mode	0x0*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		RADIO_SLEEP_EN_1	Allow to disable radio	0x1
0	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

6.6.0.14 BB_DEEPSLWKUP

Bit Field	Read/Write	Field Name	Description
31:0	RW	DEEPSLTIME	Determine the time in low_power_clk clock cycles to spend in deep sleep mode before waking-up the device

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:0	DEEPSLTIME	DEEPSLTIME_0		0x0*

6.6.0.15 BB_DEEPSLSTAT

Bit Field	Read/Write	Field Name	Description
31:0	R	DEEPSLDUR	Actual duration of the last deep sleep phase measured in low_power_clk clock cycle

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:0	DEEPSLDUR	DEEPSLDUR_0	DEEPSLDUR is set to zero at the beginning of the deep sleep phase, and is incremented at each low_power_clk clock cycle until the end of the deep sleep phase.	0x0*

6.6.0.16 BB_ENBPRESET

Bit Field	Read/Write	Field Name	Description
31:21	RW	TWEXT	Time in low power oscillator cycles allowed for stabilization of the high frequency oscillator following an external wake-up request (signal wakeup_req)
20:10	RW	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])
9:0	RW	TWRM	Time in low power oscillator cycles allowed for the radio module to leave low-power mode

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:21	TWEXT	TWEXT_0		0x0*
20:10	TWOSC	TWOSC_0		0x0*
9:0	TWRM	TWRM_0		0x0*

6.6.0.17 BB_FINECNTCORR

Bit Field	Read/Write	Field Name	Description
9:0	RW	FINECNTCORR	Phase correction value for the 312.5us reference counter (i.e. fine counter) in half us

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
9:0	FINECNTCORR	FINECNTCORR_0		0x0*

6.6.0.18 BB_CLKNCNTCORR

Bit Field	Read/Write	Field Name	Description
31	RW	ABS_DELTA	Determine whether CLKNCNTCORR is an absolute correction or a signed "delta" increment correction
27:0	RW	CLKNCNTCORR	CLKN counter correction value

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31	ABS_DELTA	ABS_DELTA_0	Absolute correction	0x0*
		ABS_DELTA_1	Signed "delta" increment correction.	0x1
27:0	CLKNCNTCORR	CLKNCNTCORR_0	CLKN Counter correction value	0x0*

6.6.0.19 BB_DIAGCNTL

Bit Field	Read/Write	Field Name	Description
31	RW	DIAG3_EN	Enable diagnostic port 3 output
30:24	RW	DIAG3	Selection of the outputs that must be driven to the diagnostic port 3
23	RW	DIAG2_EN	Enable diagnostic port 2 output
22:16	RW	DIAG2	Selection of the outputs that must be driven to the diagnostic port 2
15	RW	DIAG1_EN	Enable diagnostic port 1 output
14:8	RW	DIAG1	Selection of the outputs that must be driven to the diagnostic port 1
7	RW	DIAG0_EN	Enable diagnostic port 0 output
6:0	RW	DIAG0	Selection of the outputs that must be driven to the diagnostic port 1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31	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
30:24	DIAG3	DIAG3_0		0x0*
23	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
22:16	DIAG2	DIAG2_0		0x0*
15	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
14:8	DIAG1	DIAG1_0		0x0*
7	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
6:0	DIAG0	DIAG0_0		0x0*

6.6.0.20 BB_DIAGSTAT

Bit Field	Read/Write	Field Name	Description
31:24	R	DIAG3STAT	Directly connected to ble_dbg3[7:0] output (debug use only)
23:16	R	DIAG2STAT	Directly connected to ble_dbg2[7:0] output (debug use only)
15:8	R	DIAG1STAT	Directly connected to ble_dbg1[7:0] output (debug use only)
7:0	R	DIAG0STAT	Directly connected to ble_dbg0[7:0] output (debug use only)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:24	DIAG3STAT	DIAG3STAT_0		0x0*
23:16	DIAG2STAT	DIAG2STAT_0		0x0*
15:8	DIAG1STAT	DIAG1STAT_0		0x0*
7:0	DIAG0STAT	DIAG0STAT_0		0x0*

6.6.0.21 BB_DEBUGADDMAX

Bit Field	Read/Write	Field Name	Description
31:16	RW	REG_ADDMAX	Upper limit for the register zone indicated by the reg_inzone flag
15:0	RW	EM_ADDMAX	Upper limit for the exchange memory zone indicated by the em_ inzone flag

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:16	REG_ADDMAX	REG_ADDMAX_0		0x0*
15:0	EM_ADDMAX	EM_ADDMAX_0		0x0*

6.6.0.22 BB_DEBUGADDMIN

Bit Field	Read/Write	Field Name	Description
31:16	RW	REG_ADDMIN	Lower limit for the register zone indicated by the reg_inzone flag
15:0	RW	EM_ADDMIN	Lower limit for the exchange memory zone indicated by the em_ inzone flag

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:16	REG_ADDMIN	REG_ADDMIN_0		0x0*
15:0	EM_ADDMIN	EM_ADDMIN_0		0x0*

6.6.0.23 BB_ERRORTYPESTAT

Bit Field	Read/Write	Field Name	Description
22	R	DFCNTL_EMACC_ERROR	Indicate that direction finding controller has an EM Access error (happen when exchange memory accesses are not served in time and data are corrupted)
21	R	FIFOINTOVF	Indicate that the FIFO IRQ is overflowed
20	R	PHY_ERROR	Indicate that the programmed CS-AUX/TX/RX-RATE fields are not matching the RADIOCNTL2-PHYMSK fields indicating which PHY the radio is currently supporting
19	R	TXAEHEADER_PTR_ERROR	Indicate Tx pointer to the extended advertising packet that has to be sent is null, while the extended header length is not null and the packet to be sent is an extended advertising packet
18	R	TMAFS_ERROR	Indicate T_MAFS is smaller than 300us in between a transmitted advertising packet containing an AuxPtr field and its chained packet (this comes from bad settings of Aux_Offset and/or Offset_ Unit values)
17	R	RAL_UNDERRUN	Indicate resolving address list engine under run issue (happen

Bit Field	Read/Write	Field Name	Description
			when RAL List parsing not finished on time)
16	R	RAL_ERROR	Indicate resolving address list engine faced a bad setting
15	R	RXDATA_PTR_ERROR	Indicate whether Rx data buffer pointer value programmed is null (major failure)
14	R	TXDATA_PTR_ERROR	Indicate 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)
13	R	RXDESC_EMPTY_ERROR	Indicate whether Rx descriptor pointer value programmed in register is null (major failure)
12	R	TXDESC_EMPTY_ERROR	Indicate whether Tx descriptor pointer value programmed in control structure is null during advertising/scanning/initiating events (major failure)
11	R	CSFORMAT_ERROR	Indicate whether CS-FORMAT has been programmed with an invalid value (major failure)
10	R	LLCHMAP_ERROR	Indicate link layer channel map error (happen when actual number of CS-LLCHMAP bit set to one is different from CS-NBCHGOOD at the beginning of frequency hopping process)
9	R	ADV_UNDERRUN	Indicate advertising interval under run
8	R	IFS_UNDERRUN	Indicate inter frame space under run (occur 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)
7	R	LIST_ERROR	Indicate a software programming issue (white list or periodic advertiser list or ADI list search request with empty list, or null base pointer)
6	R	EVT_CNTL_APFM_ERROR	Indicate anticipated pre-fetch mechanism error (happen when 2 consecutive events are programmed, and when the first event is not completely finished while second pre-fetch instant is reached)
5	R	ACT_SCHDL_APFM_ERROR	Indicate anticipated pre-fetch mechanism error (happen when 2 consecutive events are programmed, and when the first event is not completely finished while second pre-fetch instant is reached)
4	R	ACT_SCHDL_ENTRY_ERROR	Indicate activity scheduler faced Invalid timing programing on two consecutive ET entries
3	R	RADIO_EMACC_ERROR	Indicate radio controller exchange memory access error (happen when exchange memory accesses are not served in time and data are corrupted)
2	R	PKTCNTL_EMACC_ERROR	Indicate packet controller exchange memory access error (happen when exchange memory accesses are not served in time and

Bit Field	Read/Write	Field Name	Description
			Tx/Rx data are corrupted)
1	R	RXCRYPT_ERROR	Indicate real time decryption error (happen when AES-CCM decryption is too slow compared to packet controller requests)
0	R	TXCRYPT_ERROR	Indicate real time encryption error (happen when AES-CCM encryption is too slow compared to packet controller requests)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
22	DFCNTL_EMACC_ ERROR	DFCNTL_EMACC_ERROR_0	No error	0x0*
		DFCNTL_EMACC_ERROR_1	Error occurred	0x1
21	FIFOINTOVF	FIFOINTOVF_0	No error	0x0*
		FIFOINTOVF_1	Error occurred	0x1
20	PHY_ERROR	PHY_ERROR_0	No error	0x0*
		PHY_ERROR_1	Error occurred	0x1
19	TXAEHEADER_PTR_ ERROR	TXAEHEADER_PTR_ERROR_0	No error	0x0*
		TXAEHEADER_PTR_ERROR_1	Error occurred	0x1
18	TMAFS_ERROR	TMAFS_ERROR_0	No error	0x0*
		TMAFS_ERROR_1	Error occurred	0x1
17	RAL_UNDERRUN	RAL_UNDERRUN_0	No error	0x0*
		RAL_UNDERRUN_1	Error occurred	0x1
16	RAL_ERROR	RAL_ERROR_0	No error	0x0*
		RAL_ERROR_1	Error occurred	0x1
15	RXDATA_PTR_ERROR	RXDATA_PTR_ERROR_0	No error	0x0*
		RXDATA_PTR_ERROR_1	Error occurred	0x1
14	TXDATA_PTR_ERROR	TXDATA_PTR_ERROR_0	No error	0x0*
		TXDATA_PTR_ERROR_1	Error occurred	0x1
13	RXDESC_EMPTY_ ERROR	RXDESC_EMPTY_ERROR_0	No error	0x0*
		RXDESC_EMPTY_ERROR_1	Error occurred	0x1
12	TXDESC_EMPTY_ ERROR	TXDESC_EMPTY_ERROR_0	No error	0x0*
		TXDESC_EMPTY_ERROR_1	Error occurred	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
11	CSFORMAT_ERROR	CSFORMAT_ERROR_0	No error	0x0*
		CSFORMAT_ERROR_1	Error occurred	0x1
10	LLCHMAP_ERROR	LLCHMAP_ERROR_0	No error	0x0*
		LLCHMAP_ERROR_1	Error occurred	0x1
9	ADV_UNDERRUN	ADV_UNDERRUN_0	No error	0x0*
		ADV_UNDERRUN_1	Error occurred	0x1
8	IFS_UNDERRUN	IFS_UNDERRUN_0	No error	0x0*
		IFS_UNDERRUN_1	Error occurred	0x1
7	LIST_ERROR	LIST_ERROR_0	No error	0x0*
		LIST_ERROR_1	Error occurred	0x1
6	EVT_CNTL_APFM_ ERROR	EVT_CNTL_APFM_ERROR_0	No error	0x0*
		EVT_CNTL_APFM_ERROR_1	Error occurred	0x1
5	ACT_SCHDL_APFM_ ERROR	ACT_SCHDL_APFM_ERROR_0	No error	0x0*
		ACT_SCHDL_APFM_ERROR_1	Error occurred	0x1
4	ACT_SCHDL_ENTRY_ ERROR	ACT_SCHDL_ENTRY_ERROR_0	No error	0x0*
		ACT_SCHDL_ENTRY_ERROR_1	Error occurred	0x1
3	RADIO_EMACC_ERROR	RADIO_EMACC_ERROR_0	No error	0x0*
		RADIO_EMACC_ERROR_1	Error occurred	0x1
2	PKTCNTL_EMACC_ ERROR	PKTCNTL_EMACC_ERROR_0	No error	0x0*
		PKTCNTL_EMACC_ERROR_1	Error occurred	0x1
1	RXCRYPT_ERROR	RXCRYPT_ERROR_0	No error	0x0*
		RXCRYPT_ERROR_1	Error occurred	0x1
0	TXCRYPT_ERROR	TXCRYPT_ERROR_0	No error	0x0*
		TXCRYPT_ERROR_1	Error occurred	0x1

6.6.0.24 BB_SWPROFILING

Bit Field	Read/Write	Field Name	Description
31:0	RW	SWPROF	Software profiling register (used by RW-BLE software for profiling purpose)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:0	SWPROF	SWPROF_0		0x0*

6.6.0.25 BB_RADIOCNTL0

Bit Field	Read/Write	Field Name	Description
29:16	RW	SPIPTR	Pointer to the buffer containing data to be transferred to or received from the SPI port
7	RW	SPICFG	SPI configuration/used for SW-driven access and SPI structure interpretation (interpretation is radio dependent)
5:4	RW	SPIFREQ	SPI clock frequency
1	R	SPICOMP	SPI transfer status
0	RW	SPIGO	Start SPI transfer when writing a 1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
29:16	SPIPTR	SPIPTR_0	SPI pointer	0x0*
7	SPICFG	SPICFG_0		0x0*
		SPICFG_1		0x1
5:4	SPIFREQ	SPIFREQ_0	SPI clock is master1_gclk / 2	0x0*
		SPIFREQ_1	NA for IcyTRX	0x1
		SPIFREQ_2	NA for IcyTRX	0x2
		SPIFREQ_3	NA for IcyTRX	0x3
1	SPICOMP	SPICOMP_0	Indicates SPI transfer in progress	0x0
		SPICOMP_1	Indicates SPI transfer is completed. RW-BLE core is ready to start a new transfer	0x1*
0	SPIGO	SPIGO_0		0x0*
		SPIGO_1	Triggers the SPI transfer	0x1

6.6.0.26 BB_RADIOCNTL1

Bit Field	Read/Write	Field Name	Description
31	RW	FORCEAGC_EN	Control AGC force mode based onto FORCEAGC_LENGTH value
30	RW	FORCEIQ	Control modulation mode in between FM and I and Q
29	RW	RXDNSL	Do not send length (over SPI) during Rx operation
28	RW	TXDNSL	Do not send length (over SPI) during Tx operation

Bit Field	Read/Write	Field Name	Description
27:16	RW	FORCEAGC_LENGTH	Control ATLAS/Ripple AGC force mode based on radioCNTL2-FORCEAGC_LENGTH value
15	RW	SYNC_PULSE_MODE	Define whether the SYNC_P pulse is generated as pulse or level
14	RW	SYNC_PULSE_SRC	Define whether access address synchronization detection is generated internally or comes from the radio
13	RW	DPCORR_EN	Enable the use of delayed DC compensated data path in radio correlator block
12	RW	JEF_SELECT	Select jitter elimination FIFO
9:4	RW	XRFSEL	Extended radio selection field
3:0	RW	SUBVERSION	CSEM RF sub-version selection

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31	FORCEAGC_EN	FORCEAGC_EN_0	Disable	0x0*
		FORCEAGC_EN_1	Enable	0x1
30	FORCEIQ	FORCEBLEIQ_0	FM modulation mode	0x0*
		FORCEBLEIQ_1	I and Q modulation mode	0x1
29	RXDNSL	RXDNSL_0	Normal operations	0x0*
		RXDNSL_1	Prevent from sending valid reception length indication to the RF over SPI	0x1
28	TXDNSL	TXDNSL_0	Normal operations	0x0*
		TXDNSL_1	Prevent from sending valid transmit length indication to the RF over SPI	0x1
27:16	FORCEAGC_LENGTH	FORCEAGC_LENGTH_0		0x0*
15	SYNC_PULSE_MODE	SYNC_PULSE_MODE_0	SYNC_P generated as pulse	0x0*
		SYNC_PULSE_MODE_1	SYNC_P generated as level	0x1
14	SYNC_PULSE_SRC	SYNC_PULSE_SRC_0	Internal detection	0x0*
		SYNC_PULSE_SRC_1	External detection	0x1
13	DPCORR_EN	DPCORR_EN_0	Disable	0x0*
		DPCORR_EN_1	Enable	0x1
12	JEF_SELECT	JEF_SELECT_0		0x0*
		JEF_SELECT_1		0x1
9:4	XRFSEL	NONE	No radio selected	0x0*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		RIPPLE	Ripple radio (BT4.0)	0x1
		EXTERNAL	External radio controller	0x2
		ICYTRX_V2	ICYTRX radio (BLE)	0x4
		BTIPT	BTIPT Radio Controller	0x5
3:0	SUBVERSION	GCS2_LR	IcyTRx GCS2 withLong Range (BLE 5.0)	0x0*
		GCS3	IcyTRx GCS3 (BLE 5.1)	0x1

6.6.0.27 BB_RADIOCNTL2

Bit Field	Read/Write	Field Name	Description
31:30	RW	LRSYNCCOMPMODE	Long-range synchronization compensation operating mode
29	RW	RXCITERMBYPASS	Long-range CI bit[1] and TERM1 bypass mode (allow to receive only CI bit 0 and then wait for Rx payload directly)
28:24	RW	LRVTBFLUSH	Indicate long range Viterbi flush instant (this value corresponds to the Viterbi trace back depth used in the selected design, hence corresponding to the number of remaining samples flush out just before the end of the packet)
23:22	RW	PHYMSK	Indicate selected radio PHY support capabilities (in addition to 1Mbps that is mandatory)
21:20	RW	LRSYNCERR	Number of errors allowed during long range Rx stream detection (when performed internally)
18:16	RW	SYNCERR	Indicate the maximum number of errors allowed to recognize the synchronization word
13:0	RW	FREQTABLE_PTR	Frequency table pointer

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:30	LRSYNCCOMPMODE	LRSYNCCOMPMODE_3	Bit 0 controls 125kbps LR packet part using synchro tracking and Bit 1 controls 500kbps LR packet part using synchro tracking	0x3*
29	RXCITERMBYPASS	RXCITERMBYPASS_0		0x0*
		RXCITERMBYPASS_1		0x1
28:24	LRVTBFLUSH	LRVTBFLUSH_8		0x8*
23:22	PHYMSK	PHYMSK_0	Bit 0 indicates 2Mbps support when set. Bit 1 indicates Coded HPY support when	0x0*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
			set	
21:20	LRSYNCERR	LRSYNCERR_0		0x0*
18:16	SYNCERR	SYNCERR_0	Has to be set to 0x0 for normal operations	0x0*
13:0	FREQTABLE_PTR	FREQTABLE_PTR_64		0x40*

6.6.0.28 BB_RADIOCNTL3

Bit Field	Read/Write	Field Name	Description
31:30	RW	RXRATE3CFG	Rate out programmable value in Rx when CS-RXRATE is set to 0x3 (i.e 500kbps Long range)
29:28	RW	RXRATE2CFG	Rate out programmable value in Rx when CS-RXRATE is set to 0x2 (i.e 125kbps Long range)
27:26	RW	RXRATE1CFG	Rate out programmable value in Rx when CS-RXRATE is set to 0x1 (i.e 2Mbps)
25:24	RW	RXRATE0CFG	Rate out programmable value in Rx when CS-RXRATE is set to 0x0 (i.e 1Mbps)
22:20	RW	GETRSSIDELAY	Delay to read RSSI after an RSSI read request
18	RW	RXSYNC_ROUTING	Access address detection information routing
17:16	RW	RXVALID_BEH	Define radio_in[3] expected behavior
15:14	RW	TXRATE3CFG	Rate out programmable value in Tx when CS-TX/AUX-RATE is set to 0x3 (i.e 500kbps Long range)
13:12	RW	TXRATE2CFG	Rate out programmable value in Tx when CS-TX/AUX-RATE is set to 0x2 (i.e 125kbps Long range)
11:10	RW	TXRATE1CFG	Rate out programmable value in Tx when CS-TX/AUX-RATE is set to 0x1 (i.e 2Mbps)
9:8	RW	TXRATE0CFG	Rate out programmable value in Tx when CS-TX/AUX-RATE is set to 0x0 (i.e 1Mbps)
1:0	RW	TXVALID_BEH	Define radio_out [3] expected behavior

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:30	RXRATE3CFG	RXRATE3CFG_3		0x3*
29:28	RXRATE2CFG	RXRATE3CFG_2		0x2*
27:26	RXRATE1CFG	RXRATE3CFG_1		0x1*
25:24	RXRATE0CFG	RXRATE3CFG_0		0x0*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
22:20	GETRSSIDELAY	GETRSSIDELAY_4		0x4*
18	RXSYNC_ROUTING	RXSYNC_ROUTING_0	Uses radio_in[16] (i.e sync_p)	0x0*
		RXSYNC_ROUTING_1	Uses first radio_in[3] edge (Rx Valid)	0x1
17:16	RXVALID_BEH	RXVALID_BEH_0	Rx data aligned on radio_in[3] rising edges	0x0*
		RXVALID_BEH_1	Rx data aligned on radio_in[3] falling edges	0x1
		RXVALID_BEH_2	Rx data aligned on radio_in[3] edges (toggle mode)	0x2
		RXVALID_BEH_3	NA	0x3
15:14	TXRATE3CFG	TXRATE3CFG_3		0x3*
13:12	TXRATE2CFG	TXRATE3CFG_2		0x2*
11:10	TXRATE1CFG	TXRATE3CFG_1		0x1*
9:8	TXRATE0CFG	TXRATE3CFG_0		0x0*
1:0	TXVALID_BEH	TXVALID_BEH_0	Tx data aligned on radio_out[3] rising edges	0x0*
		TXVALID_BEH_1	Tx data aligned on radio_out[3] falling edges	0x1
		TXVALID_BEH_2	Tx data aligned on radio_out[3] edges (toggle mode)	0x2
		TXVALID_BEH_3	NA	0x3

6.6.0.29 BB_RADIOPWRUPDN0

Bit Field	Read/Write	Field Name	Description
31:24	RW	SYNC_POSITION0	Access address detection pulse/level position for uncoded PHY at 1Mbps
23:16	RW	RXPWRUP0	Radio Rx power up (in us) for uncoded PHY at 1Mbps
14:8	RW	TXPWRDN0	Radio Tx power down (in us) for uncoded PHY at 1Mbps
7:0	RW	TXPWRUP0	Radio Tx power up (in us) for uncoded PHY at 1Mbps

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:24	SYNC_POSITION0	SYNC_POSITION0_0		0x0*
23:16	RXPWRUP0	RXPWRUP0_0		0x0*
14:8	TXPWRDN0	TXPWRDN0_0		0x0*
7:0	TXPWRUP0	TXPWRUP0_0		0x0*

6.6.0.30 BB_RADIOPWRUPDN1

Bit Field	Read/Write	Field Name	Description
31:24	RW	SYNC_POSITION1	Access address detection pulse/level position for uncoded PHY at 2Mbps
23:16	RW	RXPWRUP1	Radio Rx power up (in us) for uncoded PHY at 2Mbps
14:8	RW	TXPWRDN1	Radio Tx power down (in us) for uncoded PHY at 2Mbps
7:0	RW	TXPWRUP1	Radio Tx power up (in us) for uncoded PHY at 2Mbps

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:24	SYNC_POSITION1	SYNC_POSITION1_0		0x0*
23:16	RXPWRUP1	RXPWRUP1_0		0x0*
14:8	TXPWRDN1	TXPWRDN1_0		0x0*
7:0	TXPWRUP1	TXPWRUP1_0		0x0*

6.6.0.31 BB_RADIOPWRUPDN2

Bit Field	Read/Write	Field Name	Description
31:24	RW	SYNC_POSITION2	Access address detection pulse/level position for coded PHY at 125kbps
23:16	RW	RXPWRUP2	Radio Rx power up (in us) for coded PHY at 125kbps
14:8	RW	TXPWRDN2	Radio Tx power down (in us) for coded PHY at 125kbps
7:0	RW	TXPWRUP2	Radio Tx power up (in us) for coded PHY at 125kbps

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:24	SYNC_POSITION2	SYNC_POSITION2_0		0x0*
23:16	RXPWRUP2	RXPWRUP2_0		0x0*
14:8	TXPWRDN2	TXPWRDN2_0		0x0*
7:0	TXPWRUP2	TXPWRUP2_0		0x0*

6.6.0.32 BB_RADIOPWRUPDN3

Bit Field	Read/Write	Field Name	Description
14:8	RW	TXPWRDN3	Radio Tx power down (in us) for coded PHY at 500kbps
7:0	RW	TXPWRUP3	Radio Tx power up for (in us) PHY at 500kbps

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
14:8	TXPWRDN3	TXPWRDN3_0		0x0*
7:0	TXPWRUP3	TXPWRUP3_0		0x0*

6.6.0.33 BB_RADIOTXRXTIM0

Bit Field	Read/Write	Field Name	Description
22:16	RW	RFRXTMDA0	RF Rx test mode delay adjustment for uncoded PHY at 1Mbps
14:8	RW	RXPATHDLY0	Rx path delay (in us) for uncoded PHY at 1Mbps
6:0	RW	TXPATHDLY0	Rx path delay (in us) for uncoded PHY at 1Mbps

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
22:16	RFRXTMDA0	RFRXTMDA0_0		0x0*
14:8	RXPATHDLY0	RXPATHDLY0_0		0x0*
6:0	TXPATHDLY0	TXPATHDLY0_0		0x0*

6.6.0.34 BB_RADIOTXRXTIM1

Bit Field	Read/Write	Field Name	Description
22:16	RW	RFRXTMDA1	RF Rx test mode delay adjustment for uncoded PHY at 2Mbps
14:8	RW	RXPATHDLY1	Rx path delay (in us) for uncoded PHY at 2Mbps
6:0	RW	TXPATHDLY1	Rx path delay (in us) for uncoded PHY at 2Mbps

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
22:16	RFRXTMDA1	RFRXTMDA1_0		0x0*
14:8	RXPATHDLY1	RXPATHDLY1_0		0x0*
6:0	TXPATHDLY1	TXPATHDLY1_0		0x0*

6.6.0.35 BB_RADIOTXRXTIM2

Bit Field	Read/Write	Field Name	Description
31:24	RW	RXFLUSHPATHDLY2	Rx path delay (in us) for coded PHY at 125kbps
23:16	RW	RFRXTMDA2	RF Rx test mode delay adjustment for uncoded PHY at 125kbps
15:8	RW	RXPATHDLY2	Rx path delay (in us) for uncoded PHY at 125kbps
6:0	RW	TXPATHDLY2	Rx path delay (in us) for uncoded PHY at 125kbps

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:24	RXFLUSHPATHDLY2	RXFLUSHPATHDLY2_0		0x0*
23:16	RFRXTMDA2	RFRXTMDA2_0		0x0*
15:8	RXPATHDLY2	RXPATHDLY2_0		0x0*
6:0	TXPATHDLY2	TXPATHDLY2_0		0x0*

6.6.0.36 BB_RADIOTXRXTIM3

Bit Field	Read/Write	Field Name	Description
31:24	RW	RXFLUSHPATHDLY3	Rx path delay (in us) for coded PHY at 500kbps
22:16	RW	RFRXTMDA3	RF Rx test mode delay adjustment for coded PHY at 500kbps
6:0	RW	TXPATHDLY3	Rx path delay (in us) for coded PHY at 500kbps

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:24	RXFLUSHPATHDLY3	RXFLUSHPATHDLY3_0		0x0*
22:16	RFRXTMDA3	RFRXTMDA3_0		0x0*
6:0	TXPATHDLY3	TXPATHDLY3_0		0x0*

6.6.0.37 BB_SPIPTRCNTL0

Bit Field	Read/Write	Field Name	Description
29:16	RW	TXOFFPTR	Pointer to the TxOFF sequence address section
13:0	RW	TXONPTR	Pointer to the TxON sequence address section

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
29:16	TXOFFPTR	TXOFFPTR_0		0x0*
13:0	TXONPTR	TXONPTR_0		0x0*

6.6.0.38 BB_SPIPTRCNTL1

Bit Field	Read/Write	Field Name	Description
29:16	RW	RXOFFPTR	Pointer to the RxOFF sequence address section
13:0	RW	RXONPTR	Pointer to the RxON sequence address section

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
29:16	RXOFFPTR	RXOFFPTR_0		0x0*
13:0	RXONPTR	RXONPTR_0		0x0*

6.6.0.39 BB_SPIPTRCNTL2

Bit Field	Read/Write	Field Name	Description
29:16	RW	RXLENGTHPTR	Pointer to the received length write sequence address section
13:0	RW	RSSIPTR	Pointer to the RSSI read sequence address section

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
29:16	RXLENGTHPTR	RXLENGTHPTR_0		0x0*
13:0	RSSIPTR	RSSIPTR_0		0x0*

6.6.0.40 BB_SPIPTRCNTL3

Bit Field	Read/Write	Field Name	Description
29:16	RW	CTESAMPPTR	Pointer to the CTE sampling indication sequence address section
13:0	RW	RXPKTTYPPTR	Pointer to the received packet type indication sequence address section

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
29:16	CTESAMPPTR	CTESAMPPTR_0		0x0*
13:0	RXPKTTYPPTR	RXPKTTYPPTR_0		0x0*

6.6.0.41 BB_AESCNTL

Bit Field	Read/Write	Field Name	Description
1	RW	AES_MODE	Cipher mode control
0	RW	AES_START	Start AES-128 ciphering/deciphering process

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
1	AES_MODE	AES_MODE_0	Cipher mode	0x0*
		AES_MODE_1	Decipher mode	0x1
0	AES_START	AES_START_0		0x0*
		AES_START_1	Starts AES-128 ciphering/deciphering process (the bit is reset once the process is finished)	0x1

6.6.0.42 BB_AESKEY31_0

Bit Field	Read/Write	Field Name	Description
31:0	RW	AESKEY31_0	AES encryption 128-bit key (bits 31 down to 0)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:0	AESKEY31_0	AESKEY31_0_0		0x0*

6.6.0.43 BB_AESKEY63_32

Bit Field	Read/Write	Field Name	Description
31:0	RW	AESKEY63_32	AES encryption 128-bit key (bits 63 down to 32)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:0	AESKEY63_32	AESKEY63_32_0		0x0*

6.6.0.44 BB_AESKEY95_64

Bit Field	Read/Write	Field Name	Description
31:0	RW	AESKEY95_64	AES encryption 128-bit key (bits 95 down to 64)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:0	AESKEY95_64	AESKEY95_64_0		0x0*

6.6.0.45 BB_AESKEY127_96

Bit Field	Read/Write	Field Name	Description
31:0	RW	AESKEY127_96	AES encryption 128-bit key (bits 127 down to 96)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:0	AESKEY127_96	AESKEY127_96_0		0x0*

6.6.0.46 BB_AESPTR

Bit Field	Read/Write	Field Name	Description
13:0	RW	AESPTR	Pointer to the memory zone where the block to cipher/decipher using AES-128 is stored

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
13:0	AESPTR	AESPTR_0		0x0*

6.6.0.47 BB_TXMICVAL

Bit Field	Read/Write	Field Name	Description
31:0	R	TXMICVAL	AES-CCM plain MIC value (valid on when MIC has been calculated in Tx)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:0	TXMICVAL	TXMICVAL_0		0x0*

6.6.0.48 BB_RXMICVAL

Bit Field	Read/Write	Field Name	Description
31:0	R	RXMICVAL	AES-CCM plain MIC value (valid on once MIC has been extracted from Rx packet)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:0	RXMICVAL	RXMICVAL_0		0x0*

6.6.0.49 BB_RFTESTCNTL

Bit Field	Read/Write	Field Name	Description
31	RW	INFINITERX	Applicable in RF test mode only
27	RW	RXPKTCNTEN	Applicable in RF test mode only
25:24	RW	PERCOUNT_MODE	Applicable in RF direct Rx test mode only, and when RXPKTCNTEN equals to 1
15	RW	INFINITETX	Applicable in RF test mode only
14	RW	TXLENGTHSRC	Applicable only in Tx/Rx RF test mode
13	RW	PRBSTYPE	Applicable only in Tx/Rx RF test mode
12	RW	TXPLDSRC	Applicable only in Tx/Rx RF test mode
11	RW	TXPKTCNTEN	Applicable in RF test mode only
7:0	RW	TXLENGTH	Tx packet length in number of byte

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31	INFINITERX	INFINITERX_0	Normal mode of operation	0x0*
		INFINITERX_1	Infinite Rx window	0x1
27	RXPKTCNTEN	RXPKTCNTEN_0	Rx packet count disabled	0x0*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
25:24	PERCOUNT_MODE	PERCOUNT_MODE_0	Counts number of correctly received packets (no error)	0x0*
		PERCOUNT_MODE_1	Counts number of Access Address error detection only	0x1
		PERCOUNT_MODE_2	Counts number of CRC Error detection only (or any error leading to CRC error, such as length error or Rx time error)	0x2
		PERCOUNT_MODE_3	Counts Reception Error detected	0x3
15	INFINITETX	INFINITETX_0	Normal mode of operation	0x0*
		INFINITETX_1	Infinite Tx packet / Normal start of a packet but endless payload	0x1
14	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
13	PRBSTYPE	PRBSTYPE_0	Tx packet payload are PRBS9 type	0x0*
		PRBSTYPE_1	Tx packet payload are PRBS15 type	0x1
12	TXPLDSRC	TXPLDSRC_0	Tx packet payload source is the control structure	0x0*
		TXPLDSRC_1	Tx packet payload are PRBS generator	0x1
11	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
7:0	TXLENGTH	TXLENGTH_0		0x0*

6.6.0.50 BB_RFTESTTXSTAT

Bit Field	Read/Write	Field Name	Description
31:0	R	TXPKTCNT	Report number of transmitted packet during test modes

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:0	TXPKTCNT	TXPKTCNT_0		0x0*

6.6.0.51 BB_RFTESTRXSTAT

Bit Field	Read/Write	Field Name	Description
31:0	R	RXPKTCNT	Report number of correctly received packet during test modes

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:0	RXPKTCNT	RXPKTCNT_0		0x0*

6.6.0.52 BB_TIMGENCNTL

Bit Field	Read/Write	Field Name	Description
25:16	RW	PREFETCHABORT_TIME	Define the instant in us at which immediate abort is required after anticipated pre-fetch abort
8:0	RW	PREFETCH_TIME	Define exchange table pre-fetch instant in us

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
25:16	PREFETCHABORT_ TIME	PREFETCHABORT_TIME_254		0x1FE*
8:0	PREFETCH_TIME	PREFETCH_TIME_150		0x96*

6.6.0.53 BB_FINETIMTGT

Bit Field	Read/Write	Field Name	Description
27:0	RW	FINETARGET	Fine timer target value on which a ble_finetgtim_irq must be generated (this timer has a precision of 312.5us and interrupt is generated only when FINETARGET = CLKN)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
27:0	FINETARGET	FINETARGET_0		0x0*

6.6.0.54 BB_CLKNTGT1

Bit Field	Read/Write	Field Name	Description
27:0	RW	CLKNTGT1	This timer has a precision of 312.5us (interrupt is generated only when CLKNTGT = CLKN and HMICROSECTGT = FINECNT)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
27:0	CLKNTGT1	CLKNTGT1_0		0x0*

6.6.0.55 BB_HMICROSECTGT1

Bit Field	Read/Write	Field Name	Description
9:0	RW	HMICROSECTGT1	This timer has a precision of 0.5us (interrupt is generated only when CLKNTGT = CLKN and HMICROSECTGT = FINECNT)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
9:0	HMICROSECTGT1	HMICROSECTGT1_0		0x0*

6.6.0.56 BB_CLKNTGT2

Bit Field	Read/Write	Field Name	Description
27:0	RW	CLKNTGT2	This timer has a precision of 312.5us (interrupt is generated only when CLKNTGT = CLKN and HMICROSECTGT = FINECNT)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
27:0	CLKNTGT2	CLKNTGT2_0		0x0*

6.6.0.57 BB_HMICROSECTGT2

Bit Field	Read/Write	Field Name	Description
9:0	RW	HMICROSECTGT2	This timer has a precision of 0.5us (interrupt is generated only when CLKNTGT = CLKN and HMICROSECTGT = FINECNT)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
9:0	HMICROSECTGT2	HMICROSECTGT2_0		0x0*

6.6.0.58 BB_SLOTCLK

Bit Field	Read/Write	Field Name	Description
31	RW	SAMP	Sample the CLKN counter value in SCLK register field
30	RW	CLKN_UPD	Update CLKN counter
27:0	RW	SCLK	Value of the 312.5us CLKN counter

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31	SAMP	SAMP_0	No action happens if it is written with 0	0x0*
		SAMP_1	Sample the CLKN Counter value in SCLK register field	0x1
30	CLKN_UPD	CLKN_UPD_0	No action happens if it is written with 0	0x0*
		CLKN_UPD_1	Update CLKN counter	0x1
27:0	SCLK	SCLK_0		0x0*

6.6.0.59 BB_FINETIMECNT

Bit Field	Read/Write	Field Name	Description
9:0	R	FINECNT	This timer has a precision of 0.5us (interrupt is generated only when CLKNTGT = CLKN and HMICROSECTGT = FINECNT)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
9:0	FINECNT	FINECNT_0		0x0*

6.6.0.60 BB_ACTSCHCNTL

Bit Field	Read/Write	Field Name	Description
31	RW	START_ACT	Request the RW-BLE core to start an event
3:0	RW	ENTRY_IDX	Indicate the activity scheduler entry index which has to be used when START_ACT is set

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31	START_ACT	START_ACT_0	No action happens if it is written with 0	0x0*
		START_ACT_1	Request the RW-BLE Core to start an event	0x1
3:0	ENTRY_IDX	ENTRY_IDX_0		0x0*

6.6.0.61 BB_STARTEVTCLKNTS

Bit Field	Read/Write	Field Name	Description
27:0	R	STARTEVTCLKNTS	Value of the CLKN counter when ble_start_int is generated

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
27:0	STARTEVTCLKNTS	STARTEVTCLKNTS_0		0x0*

6.6.0.62 BB_STARTEVTFINECNTTS

Bit Field	Read/Write	Field Name	Description
9:0	R	STARTEVTFINECNTTS	Value of the fine counter when ble_start_int is generated

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
9:0	STARTEVTFINECNTTS	STARTEVTFINECNTTS_0		0x0*

6.6.0.63 BB_ENDEVTCLKNTS

Bit Field	Read/Write	Field Name	Description
27:0	R	ENDEVTCLKNTS	Value of the CLKN counter when ble_end_int is generated

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
27:0	ENDEVTCLKNTS	ENDEVTCLKNTS_0		0x0*

6.6.0.64 BB_ENDEVTFINECNTTS

Bit Field	Read/Write	Field Name	Description
9:0	R	ENDEVTFINECNTTS	Value of the fine counter when ble_end_int is generated

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
9:0	ENDEVTFINECNTTS	ENDEVTFINECNTTS_0		0x0*

6.6.0.65 BB_SKIPEVTCLKNTS

Bit Field	Read/Write	Field Name	Description
27:0	R	SKIPEVTCLKNTS	Value of the CLKN counter when ble_skip_int is generated

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
27:0	SKIPEVTCLKNTS	SKIPEVTCLKNTS_0		0x0*

6.6.0.66 BB_SKIPEVTFINECNTTS

Bit Field	Read/Write	Field Name	Description
9:0	R	SKIPEVTFINECNTTS	Value of the fine counter when ble_skip_int is generated

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
9:0	SKIPEVTFINECNTTS	SKIPEVTFINECNTTS_0		0x0*

6.6.0.67 BB_ADVTIM

Bit Field	Read/Write	Field Name	Description
31:24	RW	TX_AUXPTR_THR	Extended advertising AuxPtr threshold value in Tx (granularity 16us)
23:16	RW	RX_AUXPTR_THR	Extended advertising AuxPtr threshold value in Rx (granularity 16us)
13:0	RW	ADVINT	Advertising packet interval defining the time interval in between two ADV_xxx packet sent (value in us)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:24	TX_AUXPTR_THR	TX_AUXPTR_THR_0		0x0*
23:16	RX_AUXPTR_THR	RX_AUXPTR_THR_0		0x0*
13:0	ADVINT	ADVINT_0		0x0*

6.6.0.68 BB_ACTSCANCNTL

Bit Field	Read/Write	Field Name	Description
24:16	RW	BACKOFF	Active scan mode back-off counter initialization value
8:0	RW	UPPERLIMIT	Active scan mode upper limit counter value

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
24:16	BACKOFF	BACKOFF_1		0x1*
8:0	UPPERLIMIT	UPPERLIMIT_1		0x1*

6.6.0.69 BB_WPALCNTL

Bit Field	Read/Write	Field Name	Description
23:16	RW	WPALNBDEV	Number of devices in the white list
13:0	RW	WPALBASEPTR	Base address pointer of the white list

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
23:16	WPALNBDEV	WPALNBDEV_0		0x0*
13:0	WPALBASEPTR	WPALBASEPTR_0		0x0*

6.6.0.70 BB_WPALCURRENPTR

Bit Field	Read/Write	Field Name	Description
13:0	RW	WPALCURRENPTR	Current pointer in use for the white list

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
13:0	WPALCURRENPTR	WPALCURRENPTR_0		0x0*

6.6.0.71 BB_SEARCH_TIMEOUT

Bit Field	Read/Write	Field Name	Description
5:0	RW	SEARCH_TIMEOUT	RAL and list search engines timeout delay in us

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
5:0	SEARCH_TIMEOUT	SEARCH_TIMEOUT_16		0x10*

6.6.0.72 BB_COEXIFCNTL0

Bit Field	Read/Write	Field Name	Description
21:20	RW	MWSSCANFREQMSK	Determine how mws_scan_frequency impacts BLE Tx and Rx
19:18	RW	WLCRXPRIOMODE	Define BLE packet ble_rx mode behavior
17:16	RW	WLCTXPRIOMODE	Define BLE packet ble_tx mode behavior
15:14	RW	MWSTXFREQMSK	Determine how MWS Tx Frequency impacts BLE Tx and Rx
13:12	RW	MWSRXFREQMSK	Determine how MWS Rx frequency impacts BLE Tx and Rx
11:10	RW	MWSTXMSK	Determine how mws_tx impacts BLE Tx and Rx
9:8	RW	MWSRXMSK	Determine how mws_rx impacts BLE Tx and Rx
7:6	RW	WLANTXMSK	Determine how wlan_tx impacts BLE Tx and Rx
5:4	RW	WLANRXMSK	Determine how wlan_rx impacts BLE Tx and Rx
3	RW	MWSWCI_EN	Enable/disable control of the WCI MWS coexistence interface (valid in dual mode only)
2	RW	MWSCOEX_EN	Enable/disable control of the MWS Coexistence control (valid in dual mode only)
1	RW	SYNCGEN_EN	Determine whether ble_sync is generated or not
0	RW	WLANCOEX_EN	Enable/disable control of the MWS/WLAN coexistence control

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
21:20	MWSSCANFREQMSK	MWSSCANFREQMSK_0	mws_scan_frequency has no impact	0x0*
		MWSSCANFREQMSK_1	mws_scan_frequency can stop BLE Tx, no impact on BLE Rx	0x1
		MWSSCANFREQMSK_2	mws_scan_frequency can stop BLE Rx,	0x2

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
			no impact on BLE Tx	
		MWSSCANFREQMSK_3	mws_scan_frequency can stop both BLE Tx and BLE Rx	0x3
19:18	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
17:16	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
15:14	MWSTXFREQMSK	MWSTXFREQMSK_0	mws Tx Frequency has no impact	0x0*
		MWSTXFREQMSK_1	mws Tx Frequency can stop BLE Tx, no impact on BLE Rx	0x1
		MWSTXFREQMSK_2	mws Tx Frequency can stop BLE Rx, no impact on BLE Tx	0x2
		MWSTXFREQMSK_3	mws Tx Frequency can stop both BLE Tx and BLE Rx	0x3
13:12	MWSRXFREQMSK	MWSRXFREQMSK_0	mws Tx Frequency has no impact	0x0*
		MWSRXFREQMSK_1	mws Tx Frequency can stop BLE Tx, no impact on BLE Rx	0x1
		MWSRXFREQMSK_2	mws Tx Frequency can stop BLE Rx, no impact on BLE Tx	0x2
		MWSRXFREQMSK_3	mws Tx Frequency can stop both BLE Tx and BLE Rx	0x3
11:10	MWSTXMSK	MWSTXMSK_0	mws_tx has no impact	0x0*
		MWSTXMSK_1	mws_tx can stop BLE Tx, no impact on BLE Rx	0x1
		MWSTXMSK_2	mws_tx can stop BLE Rx, no impact on BLE Tx	0x2
		MWSTXMSK_3	mws_tx can stop both BLE Tx and BLE	0x3

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
			Rx	
9:8	MWSRXMSK	MWSRXMSK_0	mws_tx has no impact	0x0*
		MWSRXMSK_1	mws_tx can stop BLE Tx, no impact on BLE Rx	0x1
		MWSRXMSK_2	mws_tx can stop BLE Rx, no impact on BLE Tx	0x2
		MWSRXMSK_3	mws_tx can stop both BLE Tx and BLE Rx	0x3
7:6	WLANTXMSK	WLANTXMSK_0	wlan_tx has no impact	0x0*
		WLANTXMSK_1	wlan_tx can stop BLE Tx, no impact on BLE Rx	0x1
		WLANTXMSK_2	wlan_tx can stop BLE Rx, no impact on BLE Tx	0x2
		WLANTXMSK_3	wlan_tx can stop both BLE Tx and BLE Rx	0x3
5:4	WLANRXMSK	WLANRXMSK_0	wlan_rx has no impact	0x0
		WLANRXMSK_1	wlan_rx can stop BLE Tx, no impact on BLE Rx	0x1*
		WLANRXMSK_2	wlan_rx can stop BLE Rx, no impact on BLE Tx	0x2
		WLANRXMSK_3	wlan_rx can stop both BLE Tx and BLE Rx	0x3
3	MWSWCI_EN	MWSWCI_EN_0	MWS WCI Interface disabled	0x0*
		MWSWCI_EN_1	MWS WCI Interface enabled	0x1
2	MWSCOEX_EN	MWSCOEX_EN_0	MWS Coexistence interface disabled	0x0*
		MWSCOEX_EN_1	MWS Coexistence interface enabled	0x1
1	SYNCGEN_EN	SYNCGEN_EN_0	ble_sync pulse not generated	0x0*
		SYNCGEN_EN_1	ble_sync pulse generated	0x1
0	WLANCOEX_EN	COEX_EN_0	Coexistence interface disabled	0x0*
		COEX_EN_1	Coexistence interface enabled	0x1

6.6.0.73 BB_COEXIFCNTL1

Bit Field	Read/Write	Field Name	Description
28:24	RW	WLCPRXTHR	Determine the threshold for Rx priority setting (apply on ble_rx if WLCRXPRIOMODE equals "10")
20:16	RW	WLCPTXTHR	Determine the threshold for priority setting (apply on ble_tx if WLCTXPRIOMODE equals "10")
14:8	RW	WLCPDURATION	Determine how many us the priority information must be maintained (apply on ble_tx and ble_rx if WLCTXPRIOMODE and WLCRXPRIOMODE equal "10")
6:0	RW	WLCPDELAY	Determine the delay (in us) in Tx/Rx enables rises the time BLE Tx/Rx priority has to be provided (apply on ble_tx and ble_rx if WLCTXPRIOMODE and WLCRXPRIOMODE equal "10")

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
28:24	WLCPRXTHR	WLCPRXTHR_0	If ble_pti[3:0] output value is greater than WLCPRXTHR, then Rx BLE priority is considered as high, and must be provided to the WLAN coexistence interface	0x0*
20:16	WLCPTXTHR	WLCPTXTHR_0	If ble_pti[3:0] output value is greater than WLCPTXTHR, then Tx BLE priority is considered as high, and must be provided to the WLAN coexistence interface	0x0*
14:8	WLCPDURATION	WLCPDURATION_0	Note that if WLCPDURATION = 0x00, then Tx/Rx priority levels are maintained till Tx/Rx EN are de-asserted.	0x0*
6:0	WLCPDELAY	WLCPDELAY_0		0x0*

6.6.0.74 BB_COEXIFCNTL2

Bit Field	Read/Write	Field Name	Description
11:8	RW	RX_ANT_DELAY	Time (in us) by which is anticipated bt_rx to be provided before effective radio receipt operation
3:0	RW	TX_ANT_DELAY	Time (in us) by which is anticipated bt_tx to be provided before effective radio transmit operation

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
11:8	RX_ANT_DELAY	RX_ANT_DELAY_0		0x0*
3:0	TX_ANT_DELAY	TX_ANT_DELAY_0		0x0*

6.6.0.75 BB_BLEMPRIO0

Bit Field	Read/Write	Field Name	Description
31:28	RW	BLEM7	Set priority value for passive scanning
27:24	RW	BLEM6	Set priority value for non-connectable advertising
23:20	RW	BLEM5	Set priority value for connectable advertising BLE message
19:16	RW	BLEM4	Set priority value for active scanning BLE message
15:12	RW	BLEM3	Set priority value for initiating (scanning) BLE message
11:8	RW	BLEM2	Set priority value for data channel transmission BLE message
7:4	RW	BLEM1	Set priority value for LLCP BLE message
3:0	RW	BLEM0	Set priority value for initiating (connection request response) BLE message

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:28	BLEM7	BLEM7_3		0x3*
27:24	BLEM6	BLEM6_4		0x4*
23:20	BLEM5	BLEM5_8		0x8*
19:16	BLEM4	BLEM4_9		0x9*
15:12	BLEM3	BLEM3_10		0xA*
11:8	BLEM2	BLEM2_13		0xD*
7:4	BLEM1	BLEM1_14		0xE*
3:0	BLEM0	BLEM0_15		0xF*

6.6.0.76 BB_BLEMPRIO1

Bit Field	Read/Write	Field Name	Description
31:28	RW	BLEM15	Set priority value for passive extended passive scanning on secondary advertising channels
27:24	RW	BLEM14	Set priority value for non-connectable extended advertising on secondary advertising channels
23:20	RW	BLEM13	Set priority value for connectable extended advertising on secondary advertising channels
19:16	RW	BLEM12	Set priority value for extended active scanning on secondary advertising channels
15:12	RW	BLEM11	Set priority value for extended initiating on secondary advertising channels
11:8	RW	BLEM10	Set priority value for connection establishment BLE message on

Bit Field	Read/Write	Field Name	Description
			secondary advertising channels
7:4	RW	BLEM9	Set default priority value for ISO channel subsequent Tx/Rx attempt
3:0	RW	BLEM8	Set default priority value for ISO channel first Tx/Rx attempt

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:28	BLEM15	BLEM15_3		0x3*
27:24	BLEM14	BLEM14_4		0x4*
23:20	BLEM13	BLEM13_8		0x8*
19:16	BLEM12	BLEM12_9		0x9*
15:12	BLEM11	BLEM11_10		0xA*
11:8	BLEM10	BLEM10_15		0xF*
7:4	BLEM9	BLEM9_13		0xD*
3:0	BLEM8	BLEM8_12		0xC*

6.6.0.77 BB_BLEMPRIO2

Bit Field	Read/Write	Field Name	Description
31:28	RW	BLEMDEFAULT	Set priority value for extended initiating on secondary advertising channels
11:8	RW	BLEM18	Set priority value for connection establishment BLE message on secondary advertising channels
7:4	RW	BLEM17	Set default priority value for ISO channel subsequent Tx/Rx attempt
3:0	RW	BLEM16	Set default priority value for ISO channel first Tx/Rx attempt

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:28	BLEMDEFAULT	BLEMDEFAULT_3		0x3*
11:8	BLEM18	BLEM18_2		0x2*
7:4	BLEM17	BLEM17_7		0x7*
3:0	BLEM16	BLEM16_7		0x7*

6.6.0.78 BB_RALCNTL

Bit Field	Read/Write	Field Name	Description
23:16	RW	RALNBDEV	Number of devices in RAL structure
13:0	RW	RALBASEPTR	Start address pointer of the RAL structure

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
23:16	RALNBDEV	RALNBDEV_0		0x0*
13:0	RALBASEPTR	RALBASEPTR_0		0x0*

6.6.0.79 BB_RALCURRENTPTR

Bit Field	Read/Write	Field Name	Description
13:0	RW	RALCURRENTPTR	Current pointer of the RAL structure

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
13:0	RALCURRENTPTR	RALCURRENTPTR_0		0x0*

6.6.0.80 BB_RAL_LOCAL_RND

Bit Field	Read/Write	Field Name	Description
31	RW	LRND_INIT	Writing a 1 initializes of local RPA random number generation LFSR
21:0	RW	LRND_VAL	Initialization value for local RPA random generation when LRDN_ INIT is set to 1 (report the current local RPA random number LFSR value otherwise)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31	LRND_INIT	LRND_INIT_0		0x0*
21:0	LRND_VAL	LRND_VAL_4132623		0x3F0F0F*

6.6.0.81 BB_RAL_PEER_RND

Bit Field	Read/Write	Field Name	Description
31	RW	PRND_INIT	Writing a 1 initializes of peer RPA random number generation LFSR
21:0	RW	PRND_VAL	Initialization value for peer RPA random generation when LRDN_ INIT is set to 1 (report the current local RPA random number LFSR value otherwise)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31	PRND_INIT	PRND_INIT_0		0x0*
21:0	PRND_VAL	PRND_VAL_3207408		0x30F0F0*

6.6.0.82 BB_DFCNTL0_1US

Bit Field	Read/Write	Field Name	Description
31:24	RW	RXSAMPSTINST0_1US	Adjustement delay in half us of Rx I and Q sampling start instant for LE 1M PHY (with 1us sampling interval)
23:16	RW	RXSWSTINST0_1US	Adjustement delay in half us of Rx switch start instant for LE 1M PHY (with 1us switching interval)
7:0	RW	TXSWSTINST0_1US	Adjustement delay in half us of Tx switch start instant for LE 1M PHY (with 1us switching interval)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:24	RXSAMPSTINST0_1US	RXSAMPSTINST0_1US_0		0x0*
23:16	RXSWSTINST0_1US	RXSWSTINST0_1US_0		0x0*
7:0	TXSWSTINST0_1US	TXSWSTINST0_1US_0		0x0*

6.6.0.83 BB_DFCNTL0_2US

Bit Field	Read/Write	Field Name	Description
31:24	RW	RXSAMPSTINST0_2US	Adjustement delay in half us of Rx I and Q sampling start instant for LE 1M PHY (with 2us sampling interval)
23:16	RW	RXSWSTINST0_2US	Adjustement delay in half us of Rx switch start instant for LE 1M PHY (with 2us switching interval)
7:0	RW	TXSWSTINST0_2US	Adjustement delay in half us of Tx switch start instant for LE 1M PHY (with 2us switching interval)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:24	RXSAMPSTINST0_2US	RXSAMPSTINST0_2US_0		0x0*
23:16	RXSWSTINST0_2US	RXSWSTINST0_2US_0		0x0*
7:0	TXSWSTINST0_2US	TXSWSTINST0_2US_0		0x0*

6.6.0.84 BB_DFCNTL1_1US

Bit Field	Read/Write	Field Name	Description
31:24	RW	RXSAMPSTINST1_1US	Adjustement delay in half us of Rx I and Q sampling start instant for LE 2M PHY (with 1us sampling interval)
23:16	RW	RXSWSTINST1_1US	Adjustement delay in half us of Rx switch start instant for LE 2M PHY (with 1us switching interval)
7:0	RW	TXSWSTINST1_1US	Adjustement delay in half us of Tx switch start instant for LE 2M PHY (with 1us switching interval)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:24	RXSAMPSTINST1_1US	RXSAMPSTINST1_1US_0		0x0*
23:16	RXSWSTINST1_1US	RXSWSTINST1_1US_0		0x0*
7:0	TXSWSTINST1_1US	TXSWSTINST1_1US_0		0x0*

6.6.0.85 BB_DFCNTL1_2US

Bit Field	Read/Write	Field Name	Description
31:24	RW	RXSAMPSTINST1_2US	Adjustement delay in half us of Rx I and Q sampling start instant for LE 2M PHY (with 2us sampling interval)
23:16	RW	RXSWSTINST1_2US	Adjustement delay in half us of Rx switch start instant for LE 2M PHY (with 2us switching interval)
7:0	RW	TXSWSTINST1_2US	Adjustement delay in half us of Tx switch start instant for LE 2M PHY (with 2us switching interval)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:24	RXSAMPSTINST1_2US	RXSAMPSTINST1_2US_0		0x0*
23:16	RXSWSTINST1_2US	RXSWSTINST1_2US_0		0x0*
7:0	TXSWSTINST1_2US	TXSWSTINST1_2US_0		0x0*

6.6.0.86 BB_DFCURRENTPTR

Bit Field	Read/Write	Field Name	Description
13:0	RW	DFCURRENTPTR	Rx CTE descriptor current pointer

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
13:0	DFCURRENTPTR	DFCURRENTPTR_0		0x0*

6.6.0.87 BB_DFANTCNTL

Bit Field	Read/Write	Field Name	Description
15	RW	RXPRIMIDCNTLEN	Reception primary antenna ID enable control
14:8	RW	RXPRIMANTID	Primary antenna ID to be used on each Rx start instant
7	RW	TXPRIMIDCNTLEN	Transmit primary antenna ID enable control
6:0	RW	TXPRIMANTID	Primary antenna ID to be used on each Tx start instant

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
15	RXPRIMIDCNTLEN	RXPRIMIDCNTLEN_0	RXPRIMANTID not used on each Rx start	0x0*
		RXPRIMIDCNTLEN_1	RXPRIMANTID used on each Rx start	0x1
14:8	RXPRIMANTID	RXPRIMANTID_0		0x0*
7	TXPRIMIDCNTLEN	TXPRIMIDCNTLEN_0	TXPRIMANTID not used on each Tx start	0x0*
		TXPRIMIDCNTLEN_1	TXPRIMANTID used on each Tx start	0x1
6:0	TXPRIMANTID	TXPRIMANTID_0		0x0*

6.6.0.88 BB_DFIFCNTL

Bit Field	Read/Write	Field Name	Description
7	RW	ANTSWITCH_BEH	Define the antenna switch qualifier behavior
6	RW	SAMPREQ_BEH	Define the I and Q sampling request behavior
5:4	RW	SAMPVALID_BEH	Define the I and Q sample qualifier behavior
3:2	RW	IF_WIDTH	Define the I and Q sample interface width
1	RW	MSB_LSB_ORDER	Define whether symbol is sent MSB or LSB first
0	RW	SYMBOL_ORDER	Define whether I sample or Q sample is sent first

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
7	ANTSWITCH_BEH	ANTSWITCH_BEH_0	radio_out[28] is a pulse	0x0*
		ANTSWITCH_BEH_1	radio_out[28] is a toggle	0x1
6	SAMPREQ_BEH	SAMPREQ_BEH_0	radio_out[29] is a pulse	0x0*
		SAMPREQ_BEH_1	radio_out[29] is a toggle	0x1
5:4	SAMPVALID_BEH	SAMPVALID_BEH_0	I and Q sample aligned on radio_in[33] rising edges	0x0
		SAMPVALID_BEH_1	I and Q sample aligned on radio_in[33] falling edges	0x1
		SAMPVALID_BEH_2	I and Q sample aligned on radio_in[33] edges (toggle mode)	0x2
		SAMPVALID_BEH_3	NA	0x3*
3:2	IF_WIDTH	IF_WIDTH_0	NA	0x0
		IF_WIDTH_1	4-bit interface	0x1
		IF_WIDTH_2	8-bit interface	0x2

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		IF_WIDTH_3	16-bit interface	0x3*
1	MSB_LSB_ORDER	MSB_LSB_ORDER_0	MSB sent first	0x0*
		MSB_LSB_ORDER_1	LSB sent first	0x1
0	SYMBOL_ORDER	SYMBOL_ORDER_0	I sample is sent first	0x0*
		SYMBOL_ORDER_1	Q sample is sent first	0x1

CHAPTER 7

Clock Components

All clocks and clock domains in the RSL15 system are derived from the system clock (SYSCLK) or the standby clock (STANDBYCLK).

7.1 Overview

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

1. The internal startup system RC clock (discussed in Section 7.2.1 “RC Oscillator” on page 383)
2. The RF clock provided by the system crystal clock (discussed in Section 7.2.2 “48 MHz Crystal Oscillator” on page 383)
3. STANDBYCLK
4. The SWCLK pad from the SWJ-DP (discussed in "Debug Port Clock" on page 386)
5. EXTCLK, provided at a selected GPIO

For more information about configuring SYSCLK, see Section 7.4.1 “System Clock (SYSCLK)” on page 390.

Similarly, the STANDBYCLK can be generated from two different sources, including:

1. The internal standby RC oscillator (discussed in Section 7.2.3 “Standby RC Oscillator” on page 384)
2. The 32 kHz crystal oscillator (discussed in Section "32 kHz Crystal Oscillator" on page 384)

For more information about configuring STANDBYCLK, see Section 7.4.2 “Standby Clock (STANDBYCLK)” on page 390).

A top-level clock diagram showing the clock generation and distribution of clocks within the RSL15 system is provided in the "Clock Generation and Distribution Diagram" figure (Figure 35).

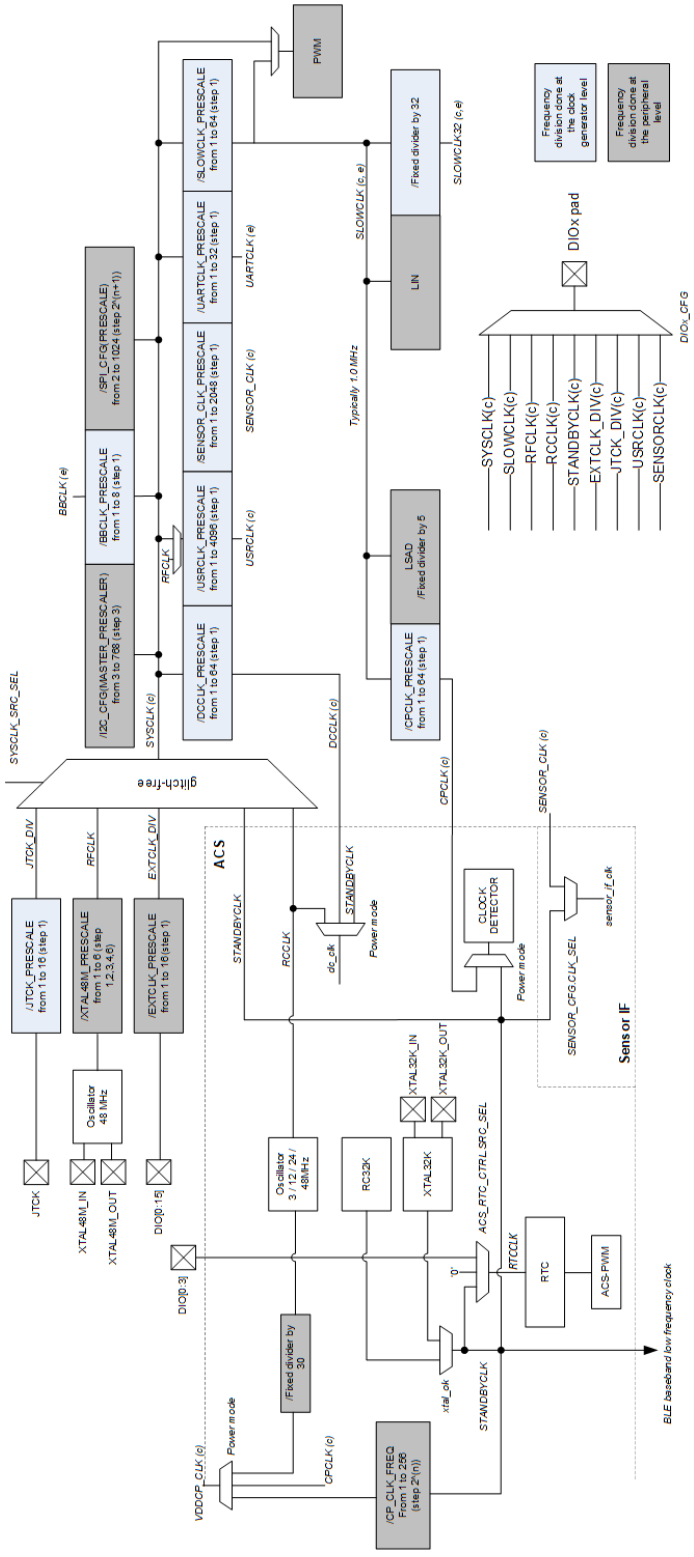


Figure 35. Clock Generation and Distribution Diagram

7.2 Clock Generation

7.2.1 RC Oscillator

The RC oscillator is a ring oscillator that produces a trimmable output clock (RCCLK), which is used by the RSL15 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. Enable the RC oscillator by setting the ACS_RCOSC_CTRL_RC_OSC_EN bit from the ACS_RCOSC_CTRL register.

The RC oscillator is configured using the ASC_RCOSC_CTRL register, including:

- A coarse trimming that selects nominal untrimmed settings between 3 and 48 MHz using the ASC_RCOSC_CTRL_RC_FSEL bit field (defaulting to 3 MHz)
- A fine trimming using the ACS_RCOSC_CTRL_RC_FTRIM bit-field
- If you have used a calibrated value, set the ACS_RCOSC_CTRL_RC_FTRIM_FLAG status flag indicating that a device has been trimmed with a calibrated value.
- The trimming range for this oscillator can be shifted down by approximately 25% by setting the ACS_RCOSC_CTRL_RC_FTRIM_ADJ bit.

Calibrated values for trim settings are provided as part of the manufacturing records in MNVR. For more information, see the *RSL15 Firmware Reference.*

RCCLK can be output through one or more GPIO pads using the GPIO components. For more information about the GPIO configuration, see Section 10.2 “Functional Configuration” on page 512.

IMPORTANT: The RC oscillator supports trimming at up to 24 MHz for all system conditions. For system configurations that ensure VCC ≥ 1.2 V, the RC oscillator supports trimming up to 48 MHz.

7.2.2 48 MHz Crystal Oscillator

The RF front-end for the RSL15 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 5.1 “Overview” on page 140.

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 the rest of the system to use. When this clock is ready, the ANALOG_INFO_CLK_DIG_READY bit from the ANALOG_INFO RF front-end register is 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 is also set. Information about further configuration of this oscillator can be found in Section 5.3 “RFFE System Resources” on page 147.

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 RF_REG33_CK_DIV_1_6_CK_DIV_1_6 bit-field from the RF_REG33 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 GPIO pads using the GPIO components. For more information about the GPIO configuration, see Section 10.2 “Functional Configuration” on page 512.

The 48 MHz crystal oscillator is trimmed using the XTAL_TRIM_XTAL_TRIM_INIT and XTAL_TRIM_XTAL_TRIM bit-fields from the XTAL_TRIM register. When the underlying 48 MHz crystal is specified for the default load capacitance of 8 pF, both of these bit-fields must be set to their defaults (0x60). If a 48 MHz crystal with a different load capacitance is selected, then the crystal must be trimmed by following these steps:

1. Configure and enable the RFCLK, configuring the RF_REG33_CK_DIV_1_6_CK_DIV_1_6 bit-field to prescale the 48 MHz oscillator by a factor of 6 (RFCLK will be 8 MHz).
2. Set up a GPIO to output RFCLK. (For more information about configuring GPIOs, see Chapter 10 "General Purpose Input/Output" on page 511.)
3. Set both XTAL_TRIM_XTAL_TRIM_INIT and XTAL_TRIM_XTAL_TRIM bit-fields to their default trim of 0x60.
4. Measure RFCLK output on the selected GPIO, adjusting XTAL_TRIM_XTAL_TRIM until the output RFCLK is as close to 8 MHz as possible.
5. Repeat this measurement over a population of 48 MHz crystals, having set both the XTAL_TRIM_XTAL_TRIM_INIT and XTAL_TRIM_XTAL_TRIM bit-fields to the trimmed value, to ensure that the expected distribution on the XTAL's initial frequency tolerance remains sufficiently well centered across the population.

NOTE: The accuracy of the selected trim can be validated using Bluetooth Low Energy test case TP/TRM-LE/CA/BV-06-C [Carrier frequency offset and drift at 1 Ms/s]. For this test, the frequency accuracy is expected to be well centered around 0 kHz, and not near the test limits of ±150 kHz.

7.2.3 Standby RC Oscillator

The standby RC oscillator is a ring oscillator that produces a trimmable output clock, which can be used by the RSL15 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_RC32_OSC_EN bit from the ACS_RCOSC_CTRL register.

The frequency of the standby RC oscillator is trimmed using the ACS_RCOSC_CTRL_RC32_FTRIM 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_RC32_FTRIM_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 needs to be measured using the 48 MHz crystal oscillator and the asynchronous clock counter. Make appropriate adjustments 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 7.4.5 “Real Time Clock (RTC)” on page 392. For more information about measuring the standby RC oscillator using the asynchronous clock counter, see Section 13.2 “Asynchronous Clock Counter” on page 686.

7.2.4 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 following:

- 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 do the following:
    - Provide a variety of different startup current levels using the ACS_XTAL32K_CTRL_ITRIM bit-field (which 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 do the following:
    - 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.

If the crystal oscillator temporarily fails and is not okay at any point after becoming ready, the ACS_XTAL32K_CTRL_XTAL_N_OK status bit is set in the ACS_XTAL32K_CTRL register. This sticky status bit is only cleared when the ACS_XTAL32K_CTRL_XTAL_N_OK_RESET bit is set.

IMPORTANT:

There is a notable startup time for the 32 kHz crystal oscillator once it is activated. When this oscillator is ready for use, the ACS_XTAL32K_CTRL_READY bit is set in the ACS_XTAL32K_CTRL register. The startup time for the crystal increases exponentially with increased capacitance, as controlled by using the ACS_XTAL32K_CTRL_CLOAD_TRIM bit-field from the ACS_XTAL32K_CTRL register. For example, with this bit-field configured for a load of 23.2 pF, the startup time is approximately two seconds.

Enabling the ACS_XTAL32K_CTRL_FORCE_READY flag bypasses the crystal oscillator ready circuitry. However, this does not greatly improve the startup time, since the main time contributor is the crystal oscillator's required startup time. Boosting the startup time by bypassing the crystal ready circuit is not a safe approach, and is not recommended.

If accurate timing is required, we recommend waiting for the ACS_XTAL32K_CTRL_READY flag to be set before using the 32 kHz crystal oscillator, as the clock could be unstable or missing for some period of time prior to this flag being set. While waiting for the 32 kHz crystal oscillator to be ready, other system tasks that do not require this oscillator can be completed in all conditions.

ACS_XTAL32K_CTRL_CLOAD_TRIM must match the crystal component CLOAD. The crystal frequency's accuracy is impacted by the parasitic capacitances outside of the chip, and also by the specifications of the chosen external resonator. For any new application, and in a case where the type of resonator is changed, we recommend performing the following calibration:

- **Output the crystal clock frequency on a GPIO to measure the frequency using default trimmings.**
- **Adjust ACS_XTAL32K_CTRL_CLOAD_TRIM to have an exact frequency of 32768 kHz. The trimming steps are only a few ppms. Ideally, ACS_XTAL32K_CTRL_CLOAD_TRIM is intended to be twice the value specified by the resonator provider. But the parasitic capacitance of the board requires using a smaller value.**
- **Once the ACS_XTAL32K_CTRL_CLOAD_TRIM is defined for the application, the startup current (controlled by ACS_XTAL32K_CTRL_ITRIM and ACS_XTAL32K_CTRL_IBOOST) needs to be calibrated. Do this by measuring the startup time of the crystal for all startup current configurations and select the configuration that gives the desirable startup time.**

7.2.5 Debug Port Clock

The JTCK signal from the SWJ-DP interface in the RSL15 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:

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

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 55.

7.3 Clock Generation Registers

Register Name	Register Description	Address
CLK_SYS_CFG	System Clock Configuration Register	0x40000100
CLK_DIV_CFG0	Prescale register for SLOWCLK, BBCLK and UARTCLK clocks	0x40000104
CLK_DIV_CFG1	Prescale register for charge pump clock and sensor clock	0x40000108
CLK_DIV_CFG2	Prescale register for User clock	0x4000010C

7.3.0.1 ACS_RTC_CFG

Bit Field	Read/Write	Field Name	Description
6:4	RW	RTC_CLOCK_SRC	Select the RTC Clock event source
2:0	RW	CLK_SRC_SEL	Select the RTC, standby, bb timer, and sensor block clock source

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
6:4	RTC_CLOCK_SRC	RTC_CLOCK_62MS	RTC clock on counter bit 10 rising edge (62 ms)	0x0*
		RTC_CLOCK_125MS	RTC clock on counter bit 11 rising edge (125 ms)	0x1
		RTC_CLOCK_250MS	RTC clock on counter bit 12 rising edge (250 ms)	0x2
		RTC_CLOCK_500MS	RTC clock on counter bit 13 rising edge (500 ms)	0x3
		RTC_CLOCK_1S	RTC clock on counter bit 14 rising edge (1 s)	0x4
		RTC_CLOCK_2S	RTC clock on counter bit 15 rising edge (2 s)	0x5
		RTC_CLOCK_4S	RTC clock on counter bit 16 rising edge (4 s)	0x6
		RTC_CLOCK_8S	RTC clock on counter bit 17 rising edge (8 s)	0x7
2:0	CLK_SRC_SEL	RTC_CLK_SRC_STANDBY_CLK	Select standby clock that runs at 32kHz as clock source	0x0*
		RTC_CLK_SRC_GPIO0	Select GPIO[0] as clock source	0x1
		RTC_CLK_SRC_GPIO1	Select GPIO[1] as clock source	0x2
		RTC_CLK_SRC_GPIO2	Select GPIO[2] as clock source	0x3
		RTC_CLK_SRC_GPIO3	Select GPIO[3] as clock source	0x4
		RTC_CLK_SRC_NO_CLOCK	Set RTC clock to constant value 0	0x5

7.3.0.2 ACS_RTC_CTRL

Bit Field	Read/Write	Field Name	Description
18	R	ENABLE_CLOCK_EVENT_STATUS	Status of the RTC clock event enable
17	R	ENABLE_ALARM_EVENT_STATUS	Status of the RTC alarm event enable
16	R	ENABLE_STATUS	Status of the RTC enable
7	W	FORCE_CLOCK	Force a clock on RTC timer (Test Purpose)
6	W	DISABLE_CLOCK_EVENT	Disable clock event and its interrupt
5	W	ENABLE_CLOCK_EVENT	Enable clock event and its interrupt
4	W	DISABLE_ALARM_EVENT	Disable alarm event and its interrupt
3	W	ENABLE_ALARM_EVENT	Enable alarm event and its interrupt
2	W	RESET	Reset the RTC timer
1	W	DISABLE	Disable counter and RTC interrupt every 1s
0	W	ENABLE	Enable counter and RTC interrupt every 1s

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
18	ENABLE_CLOCK_EVENT_STATUS	RTC_CLOCK_EVENT_IS_DISABLED	The RTC clock event is disabled	0x0*
		RTC_CLOCK_EVENT_IS_ENABLED	The RTC clock event is enabled	0x1
17	ENABLE_ALARM_EVENT_STATUS	RTC_ALARM_EVENT_IS_DISABLED	The RTC alarm event is disabled	0x0*
		RTC_ALARM_EVENT_IS_ENABLED	The RTC alarm event is enabled	0x1
16	ENABLE_STATUS	RTC_IS_DISABLED	The RTC is disabled	0x0*
		RTC_IS_ENABLED	The RTC is enabled	0x1
7	FORCE_CLOCK	RTC_FORCE_CLOCK	Clock the RTC timer (has an effect only if the source clock is low)	0x1
6	DISABLE_CLOCK_EVENT	RTC_DISABLE_CLOCK_EVENT	Disable RTC clock event	0x1
5	ENABLE_CLOCK_EVENT	RTC_ENABLE_CLOCK_EVENT	Enable RTC clock event	0x1
4	DISABLE_ALARM_EVENT	RTC_DISABLE_ALARM_EVENT	Disable RTC alarm event	0x1
3	ENABLE_ALARM_EVENT	RTC_ENABLE_ALARM_EVENT	Enable RTC alarm event	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
2	RESET	RTC_RESET	The RTC counters are reset (prescale and counter)	0x1
1	DISABLE	RTC_DISABLE	Disable RTC	0x1
0	ENABLE	RTC_ENABLE	Enable RTC	0x1

7.3.0.3 ACS_RTC_COUNT_THRES

Bit Field	Read/Write	Field Name	Description
31:0	RW	THRESHOLD	Compare value for the RTC counter

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:0	THRESHOLD	RTC_COUNT_THRES_0	Compare to 0	0x0
		RTC_COUNT_THRES_1	Compare to 1	0x1
		RTC_COUNT_THRES_32767	Compare to 32,767	0x7FFF
		RTC_COUNT_THRES_4294967295	Compare to 4,294,967,295	0xFFFFFFFF*

7.3.0.4 ACS_RTC_COUNT

Bit Field	Read/Write	Field Name	Description
31:0	R	VALUE	RTC timer current value

7.3.0.5 ACS_RTC_SECONDS

Bit Field	Read/Write	Field Name	Description
16:0	R	VALUE	RTC timer current value in seconds

7.3.0.6 ACS_RTC_COUNT_LOAD

Bit Field	Read/Write	Field Name	Description
31:0	W	VALUE	Load the RTC timer current value

7.3.0.7 ACS_BB_TIMER_CTRL

Bit Field	Read/Write	Field Name	Description
0	RW	BB_TIMER_NRESET	nReset signal for the baseband timer

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
0	BB_TIMER_NRESET	BB_TIMER_RESET	Baseband timer is in reset state	0x0*
		BB_TIMER_NRESET	Baseband timer reset is released	0x1

7.4 Clock Distribution

7.4.1 System Clock (SYSCLK)

The system clock (SYSCLK) is the primary clock for the RSL15 system and for 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 STANDBYCLK)
- The 32 kHz crystal oscillator (through STANDBYCLK)
- The divided 48 MHz crystal oscillator output (RFCLK)
- The JTCK (SWCLK) input signal
- EXTCLK, as provided at the GPIO specified by the GPIO_SRC_EXTCLK register

SYSCLK is typically 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 SWCLK/JTCK input, the frequency of the input clock must be controlled to ensure SYSCLK remains valid.

SYSCLK can be output through one or more GPIO pads using the GPIO components. For more information about the GPIO configuration, see Section 10.2 “Functional Configuration” on page 512.

IMPORTANT: When the RSL15 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 "Baseband Timer Registers" on page 330.

NOTE: For optimal performance of the RSL15 system, user applications need to use the standard calibration configurations of the RC clock sources provided for the device whenever possible, as described in the Manufacturing Records section of the *RSL15 Firmware Reference*.

7.4.2 Standby Clock (STANDBYCLK)

The RSL15 system includes a standby clock (STANDBYCLK) that is used as the source for the RTC (see Section 7.4.5 “Real Time Clock (RTC)” on page 392), and can be used as the source for SYSCLK in standby operating modes.

STANDBYCLK is sourced from one of either the RC32K oscillator, or the XTAL32 oscillator.

The selection of the STANDBYCLK is done by XTAL32K ready. When XTAL32K is ready, the XTAL32K is selected. If XTAL32 is not ready, the RC32K is selected. As the switch of these clock sources is not glitch free, enable only one of them. This avoids possible glitches on the standby clock, and saves power by keeping both oscillators from being enabled at the same time. The selection of the STANDBYCLK clock source is also not glitch-free. Enable/disable the STANDBYCLK clock source only when SYSCLK is not set to STANDBYCLK.

If both sources are disabled, the RC32K becomes enabled during startup and while entering standby mode. If the XTAL is enabled but not ready (i.e. it is not ready, the external XTAL is not present, or there is a defect), the enabling of RC32K is set to guarantee that the system will operate.

We recommend that you use the 32 kHz crystal oscillator instead of the standby RC oscillator, due to:

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

NOTE: STANDBYCLK is typically configured to be sourced from a 32768 kHz clock source. If STANDBYCLK is supplied at a different frequency, a correction factor can be applied to the timers that drive RF traffic to support input clock frequencies in the range from 25 kHz to 100 kHz. For more information, see Chapter 6 "Bluetooth Low Energy Baseband Controller" on page 301.

IMPORTANT: In typical configurations, the RSL15 Bluetooth stack is provided with a 32 kHz crystal oscillator source that provides 32768 Hz frequency with a variation of up to 500 ppm. If STANDBYCLK is within these tolerances, add the `LowPowerClock_Source_Set(0)` API function call from the Bluetooth stack library to the firmware Bluetooth Low Energy stack initialization procedure, to inform the Bluetooth stack that standard STANDBYCLK meets the low-power clock requirements for the stack.

If STANDBYCLK does not meet these operating assumptions, the stack must be informed using the `LowPowerClock_Source_Set(1)` API function call. In such a case, the RSL15 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. Otherwise, issue the `LowPowerClock_Source_Set(1)` function call, and provide the frequency of STANDBYCLK via the `RTCCLK_Period_Value_Set()` function. In addition, the following conditions need to be met:

- **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 that have a higher error rate, or that might vary over time due to changes in environmental conditions, the application needs to periodically measure and update the RTC clock period. For more information on measuring the RTC clock period, see Section 13.2 “Asynchronous Clock Counter” on page 686.**

STANDBYCLK can be output through one or more GPIO pads using the GPIO components. For more information about the GPIO configuration, see Section 10.2 “Functional Configuration” on page 512.

7.4.3 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 must always be set to 1 MHz, and is used as the clock source for:

- CPCLK (Section 7.4.7 “Power Supply Clocks” on page 395)
- LIN (see Section 11.2 “LIN” on page 588)
- LSAD (see Section 12.4 “LSAD” on page 651)

SLOWCLK can also be optionally used as the source for PWM (see Section 11.4 “PWM” on page 598).

Further divided forms of SLOWCLK (SLOWCLK_DIV2, SLOWCLK_DIV32) are used as the source for:

- The general purpose timers, which select between SLOWCLK_DIV2 and SLOWCLK_DIV32 (see Section 13.7 “Timers” on page 708)
- The watchdog timer, which uses SLOWCLK_DIV32 (see Section 13.8 “Watchdog Timer” on page 710)

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:

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

SLOWCLK can be output through one or more GPIO pads using the GPIO components. For more information about the GPIO configuration, see Section 10.2 “Functional Configuration” on page 512.

7.4.4 Baseband Clock (BBCLK)

The Bluetooth low energy baseband uses three clock sources:

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 6.5 “Baseband Timer Registers” on page 330.

7.4.5 Real Time Clock (RTC)

The real-time clock (RTC) allows a device to track the current time or the time elapsed since the application has started tracking time. The RTC is based on the RTC timer, which is a 32-bit free-running count-up timer.

The RTC is controlled using the `ACS_RTC_CTRL` register in these ways:

- The RTC is enabled by setting the `ACS_RTC_CTRL_ENABLE` bit.
- The RTC is disabled by setting the `ACS_RTC_CTRL_DISABLE` bit.
- The current RTC counter value is reset by setting the `ACS_RTC_CTRL_RESET` bit.

The current state of the RTC can be queried using the `ACS_RTC_CTRL_ENABLE_STATUS` bit from the `ACS_RTC_CTRL` register.

IMPORTANT: In the RSL15device, back-to-back writes to ACS block, or mixing writes to the ACS block with a DMA access to a peripheral can corrupt the values stored to the `ACS_RTC_COUNT` and `ACS_RTC_SECONDS` registers when the RTC is updated. To avoid these issues, the `Sys_ACS_WriteRegister()` function from the HAL library must be used for all writes to ACS registers when using the RTC.

The RTC timer is incremented based on STANDBYCLK or a GPIO input (selected from GPIO0 to GPIO3). To select the clock source for the RTC timer (including selecting no clock as the source for the RTC timer), configure the ACS_RTC_CFG_CLK_SRC_SEL bit-field from the ACS_RTC_CFG register. Only change the source for the RTC when the RTC is disabled. For information about configuring the clock source used by the RTC and RTC timer, see Section 7.4.2 "Standby Clock (STANDBYCLK)" on page 390.

The RTC timer updates the RTC once per second. The current RTC value can be read uing the ACS_RTC_SECONDS register.

IMPORTANT: The current RTC timer value can be read through the ACS_RTC_COUNT register, but reads of this register from the ACS are transferred byte-wise from the analog components to the digital systems. As such, reads of the ACS_RTC_COUNT register are non-atomic and the `Sys_RTC_Value()` function from the hardware abstraction layer (HAL) needs to be used when reading this register. For more information about the HAL library, refer to the *RSL15 Firmware Reference*.

When a wakeup event occurs during sleep mode, the eight least significant bits of the current RTC timer counter value are copied into the ACS_WAKEUP_STATE_RTC_VALUE bit-field in the ACS_WAKEUP_STATE register. This field can be used to precisely identify when a wakeup event has occurred relative to the current time.

NOTE: To simplify testing, RTC timer pulses can be manually applied by writing the ACS_RTC_CTRL_FORCE_CLOCK bit in the ACS_RTC_CTRL register. Each time this bit is written, an RTC timer clock pulse is generated. The current RTC timer count can also be manually set by loading the desired value to the ACS_RTC_COUNT_LOAD register. Only use this test bit and register when no clock source is selected for the RTC.

The RTC supports three events, as configured using the ACS_RTC_CTRL register. These events are:

RTC Clock

The RTC clock event is enabled when the ACS_RTC_CTRL_ENABLE_CLOCK_EVENT bit is set, and disabled when the ACS_RTC_CTRL_DISABLE_CLOCK_EVENT bit is set. The current configuration of the RTC clock event can be queried using the ACS_RTC_CTRL_ENABLE_CLOCK_EVENT_STATUS bit.

If enabled, the RTC clock event generates a wakeup signal and an interrupt when the RTC timer update triggers a rising edge on a bit between 10 to 17 of the RTC timer counter, supporting a configuration of the RTC clock event for an update interval of between 62.5 ms and 8 s. The bit selected for this update is configured using the ACS_RTC_CFG_RTC_CLOCK_SRC bit-field from the ACS_RTC_CFG register.

NOTE: The ACS_RTC_CFG_RTC_CLOCK_SRC bit must not be updated while the RTC clock event is enabled.

The wakeup event flag for the RTC clock event must be cleared between events to allow the generation of a new wakeup signal and interrupt.

RTC Alarm

The RTC alarm event is enabled when the ACS_RTC_CTRL_ENABLE_ALARM_EVENT bit is set, and disabled when the ACS_RTC_CTRL_DISABLE_ALARM_EVENT bit is set. The current configuration of the RTC alarm event can be queried using the ACS_RTC_CTRL_ENABLE_ALARM_EVENT_STATUS bit.

If enabled, the RTC alarm event generates a wakeup signal and an interrupt when the RTC timer exceeds the threshold defined by the value stored to the ACS_RTC_COUNT_THRES register.

NOTE: The ACS_RTC_COUNT_THRES register must not be updated while the RTC alarm event is enabled.

The wakeup event flag for the RTC alarm event must be cleared between events to allow the generation of a new wakeup signal and interrupt.

RTC Overflow

The RTC overflow event is enabled using the ACS_WAKEUP_CFG_RTC_OVERFLOW_EN bit in the ACS_WAKEUP_CFG register. If enabled, the RTC overflow event generates a wakeup signal when the RTC timer counter wraps to 0. The wakeup event flag for the RTC overflow event must be cleared between events to allow the generation of a new wakeup signal.

7.4.5.1 RTC Output Control

The ASC_AOUT_CTRL register can be used to periodically output the RTC on GPIO 0 to control the timing of an external device. Configuration options include:

RTC Start

The ACS_AOUT_CTRL_RTC_CLOCK_GPIO0_START bit-field is used to configure how often the RTC output starts. This is configurable in power of 2 multiples of 125 ms (assuming a 32.768 kHz clock frequency).

RTC Stop Source and Edge

The ACS_AOUT_CTRL_RTC_CLOCK_GPIO0_STOP_SRC bit-field and ACS_AOUT_CTRL_RTC_CLOCK_GPIO0_STOP_EDGE bit are used to configure when the RTC output signal stops. The stop signal can be triggered on an event on GPIO 0 to 3, and on either a rising or a falling edge.

NOTE: Selecting GPIO 0 as the stop source triggers a single pulse. Use of the rising edge as the stop event trigger results in a 1 cycle high pulse, and use of the falling edge of the stop event results in a half period high pulse.

IMPORTANT: The ACS_AOUT_CTRL register is also used to configure the analog test output (AOUT) signal. Care must be taken when using this register for both use cases, to avoid zeroing out the AOUT configuration when configuring for RTC output (and vice versa).

7.4.6 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 RSL15 system, so its usage can depend entirely on the outside needs of the larger system containing RSL15.

USRCLK is derived from SYSCLK or RF BBCLK through a 12-bit integer division by the CLK_DIV_CFG2_USRCLK_PRESCALE bit field in the CLK_DIV_CFG2 register. The source for USERCLK can be selected using the CLK_DIV_CFG2_USRCLK_SOURCE_SEL bit from the CLK_DIV_CFG2 register. This prescaler provides a clock prescaled from SYSCLK or RF BBCLK by 1 to 4096, and results in a USRCLK with a frequency defined by the following equation:

$$f_{USRCLK} = \frac{f_{SYSCLK\ or\ RF\ BBCLK}}{(\text{CLK\_DIV\_CFG2\_USRCLK\_PRESCALE} + 1)}$$

USRCLK can be output through one or more GPIO pads using the GPIO components. For more information about the GPIO configuration, see Section 10.2 “Functional Configuration” on page 512.

7.4.7 Power Supply Clocks

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

- The VDDCP charge pump is clocked using CPCLK, which is divided from SLOWCLK. Configuration and restrictions on this clock are described in Section 8.2.2.7 “Charge Pump Supply Voltage (VDDCP)” on page 416.
- 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 8.2.2.1 “DC-DC Converter (VCC)” on page 408.

7.4.8 Interface Clocks

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

I^2C

The I^2C clock is only used by the I^2C interface when the interface is configured for controller mode. For information about configuring the I^2C clock, see Section 11.1 “I2C Interfaces” on page 575.

PWM

For more information about this interface and its clocks, see Section 11.4 “PWM” on page 598.

SPI

For more information about these interfaces and their clocks, see Section 11.5 “Serial Peripheral Interfaces (SPI)” on page 600.

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 11.6 “Universal Asynchronous Receiver-Transmitter (UART) Interfaces” on page 607.

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

LIN

For more information about this interface and its clocks, see Section 11.2 “LIN” on page 588.

LSAD

The LSAD channels use either SLOWCLK or SLOWCLK divided by a fixed divisor of 5 to sample analog signals. Information on the LSAD interface clocking and sample configuration can be found in Section 12.4 “LSAD” on page 651.

The following interface components are special cases, which have more flexible clock assignments:

PCM

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 in the RSL15 system by routing a clock output to the same GPIO that is acting as the PCM clock input (see Section 10.2 “Functional Configuration” on page 512 for more information about configuring a GPIO as both a clock output and a PCM clock input; see Section 11.3 “Pulse Code Modulation (PCM) Interface” on page 590 for more information about the PCM interface).

Sensor Interfaces

The SAR-ADC, pulse counter, and ultra-low-power (ULP) data acquisition subsystem are all clocked using the SENSOR_CLK signal. This clock is enabled using the `CLK_CFG1_SENSOR_CLK_DISABLE` bit from the `CLK_DIV_CFG1` register. The source for this clock signal is selected using the `SENSOR_CFG_CLK_SEL` bit from the `SENSOR_CFG` register.

If standby clock (STANDBYCLK) is selected as the source for this clock, these interfaces always use STANDBYCLK.

If SYSCLK is selected as the source for this clock, SENSOR_CLK is derived from SYSCLK in run mode and uses STANDBYCLK directly when the system is in a hardware-based low power mode. SENSOR_CLK is derived from SYSCLK through an 11-bit integer division by the `CLK_DIV_CFG1_SENSOR_CLK_PRESCALE` bit field in the `CLK_DIV_CFG1` register. This prescaler provides a clock prescaled from SYSCLK by 1 to 2048, and results in a SENSOR_CLK with a frequency defined by the following equation:

$$f_{SENSOR\_CLK} = \frac{f_{SYSCLK}}{(\text{CLK\_DIV\_CFG1\_SENSOR\_CLK\_PRESCALE} + 1)}$$

For more information about these interfaces and their clocks, see Chapter 12 "Sensor Interfaces" on page 643.

7.5 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 UARTCLK clocks	0x40000104
CLK_DIV_CFG1	Prescale register for charge pump clock and sensor clock	0x40000108
CLK_DIV_CFG2	Prescale register for User clock	0x4000010C
ACS_RCOSC_CTRL	RC Oscillator control register	0x40001B24
ACS_XTAL32K_CTRL	Xtal 32 kHz configuration register	0x40001B28
ACS_CLK_DET_CTRL	Clock Detector Configuration register	0x40001B58

7.5.0.1 CLK_SYS_CFG

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

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
19:16	EXTCLK_PRESCALE	EXTCLK_PRESCALE_1	Divide by 1	0x0*
		EXTCLK_PRESCALE_2	Divide by 2	0x1
		EXTCLK_PRESCALE_4	Divide by 4	0x3
		EXTCLK_PRESCALE_8	Divide by 8	0x7
		EXTCLK_PRESCALE_15	Divide by 15	0xE
		EXTCLK_PRESCALE_16	Divide by 16	0xF
11:8	SWCLK_PRESCALE	SWCLK_PRESCALE_1	Divide by 1	0x0*
		SWCLK_PRESCALE_2	Divide by 2	0x1
		SWCLK_PRESCALE_4	Divide by 4	0x3
		SWCLK_PRESCALE_8	Divide by 8	0x7
		SWCLK_PRESCALE_15	Divide by 15	0xE
		SWCLK_PRESCALE_16	Divide by 16	0xF
2:0	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_SWCLK	Select the SWCLK clock as SYSCLK clock source	0x3
		SYSCLK_CLKSRC_EXTCLK	Select the EXTCLK clock asSYSCLK clock source	0x4

7.5.0.2 CLK_DIV_CFG0

Bit Field	Read/Write	Field Name	Description
20:16	RW	UARTCLK_PRESCALE	Prescale value for the UART peripheral clock (1 to 32 in steps of 1)
10:8	RW	BBCLK_PRESCALE	Prescale value for the Baseband peripheral clock (1 to 8 in steps of 1)
5:0	RW	SLOWCLK_PRESCALE	Prescale value for the SLOWCLK clock (1 to 64 in steps of 1)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
20:16	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
10:8	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
5:0	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_6	Divide by 6	0x5
		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

7.5.0.3 CLK_DIV_CFG1

Bit Field	Read/Write	Field Name	Description
27	RW	SENSOR_CLK_DISABLE	Sensor clock disable
26:16	RW	SENSOR_CLK_PRESCALE	Prescale value for the sensor clock
15	RW	CPCLK_DISABLE	Charge pump clock disable
13:8	RW	CPCLK_PRESCALE	Prescale value for the charge pump clock from the SLOWCLK clock (1 to 64 in steps of 1)
7	RW	DCCLK_DISABLE	DC-DC converter clock disable
5:0	RW	DCCLK_PRESCALE	Prescale value for the DC-DC converter clock (1 to 64 in steps of 1)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
27	SENSOR_CLK_ DISABLE	SENSOR_CLK_ENABLE	Sensor clock enabled	0x0
		SENSOR_CLK_DISABLE	Sensor clock disabled	0x1*
26:16	SENSOR_CLK_ PRESCALE	SENSOR_PRESCALE_1	Divide by 1	0x0*
		SENSOR_PRESCALE_2	Divide by 2	0x1
		SENSOR_PRESCALE_3	Divide by 3	0x2
		SENSOR_PRESCALE_1464	Divide by 1464	0x5B8
		SENSOR_PRESCALE_1465	Divide by 1465	0x5B9
		SENSOR_PRESCALE_1466	Divide by 1466	0x5BA
		SENSOR_PRESCALE_2046	Divide by 2046	0x7FD
		SENSOR_PRESCALE_2047	Divide by 2047	0x7FE
		SENSOR_PRESCALE_2048	Divide by 2048	0x7FF
15	CPCLK_DISABLE	CPCLK_ENABLE	Charge pump clock enabled	0x0*
		CPCLK_DISABLE	Charge pump clock disabled	0x1
13:8	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*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		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
7	DCCLK_DISABLE	DCCLK_ENABLE	DC-DC converter clock enabled	0x0*
		DCCLK_DISABLE	DC-DC converter clock disabled	0x1
5:0	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_12	Divide by 12	0xB
		DCCLK_PRESCALE_63	Divide by 63	0x3E
		DCCLK_PRESCALE_64	Divide by 64	0x3F

7.5.0.4 CLK_DIV_CFG2

Bit Field	Read/Write	Field Name	Description
20	RW	PWM_CLK_SRC	PWM clock source
16	RW	USRCLK_SRC_SEL	USR clock source selection
11:0	RW	USRCLK_PRESCALE	Prescale value for the USR clock (1 to 4096 in steps of 1)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
20	PWM_CLK_SRC	PWM_SRC_SYSCLK	PWM clock based on system clock	0x0*
		PWM_SRC_SLOWCLK	PWM clock based on SLOW clock	0x1
16	USRCLK_SRC_SEL	USRCLK_SRC_SYSCLK	User clock based on system clock	0x0*
		USRCLK_SRC_RFCLK	User clock based on RF clock	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
11:0	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

7.5.0.5 ACS_RCOSC_CTRL

Bit Field	Read/Write	Field Name	Description
26:25	RW	RC_FSEL	Select RC oscillator frequency
24	RW	RC_OSC_EN	Enable/Disable the RC Oscillator
23	RW	RC_FTRIM_FLAG	RC Oscillator Trimming flag
22	RW	RC_FTRIM_ADJ	Adjust RC oscillator frequency range
21:16	RW	RC_FTRIM	RC oscillator frequency trimming
15:14	RW	RC32_VDDLOC_ITRIM	VDDLOC current trimming
13:12	RW	RC32_VDDLOC_VTRIM	VDDLOC voltage trimming
9	RW	RC32_VDDLOC_OD_EN	Enable the overdrive of vddloc with VBAT
8	RW	RC32_OSC_EN	Enable/Disable the 32 kHz RC Oscillator
7	RW	RC32_TEMP_COMP_EN	Enable/Disable the 32 kHz RC Oscillator
6	RW	RC32_FTRIM_ADJ	Adjust 32 kHz RC oscillator frequency range
5:0	RW	RC32_FTRIM	32 kHz RC oscillator frequency trimming

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
26:25	RC_FSEL	RC_OSC_3MHZ	The RC Oscillator is at 3 MHz	0x0*
		RC_OSC_12MHZ	The RC Oscillator is at 12 MHz	0x1
		RC_OSC_24MHZ	The RC Oscillator is at 24 MHz	0x2
		RC_OSC_48MHZ	The RC Oscillator is at 48 MHz	0x3
24	RC_OSC_EN	RC_OSC_AUTO	The RC Oscillator is in auto mode	0x0*
		RC_OSC_ENABLE	The RC Oscillator is enabled	0x1
23	RC_FTRIM_FLAG	RC_OSC_UNCALIBRATED	The oscillators are not calibrated	0x0*
		RC_OSC_CALIBRATED	The oscillators are calibrated	0x1
22	RC_FTRIM_ADJ	RC_OSC_RANGE_NOM	The RC Oscillator frequency range is	0x0*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
			nominal	
		RC_OSC_RANGE_M15	The RC Osc higher trim freq is decreased by 15%	0x1
21:16	RC_FTRIM	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
9	RC32_VDDLOC_OD_EN	RC32_OSC_VDDLOC_OD_DISABLE		0x0*
		RC32_OSC_VDDLOC_OD_ENABLE		0x1
8	RC32_OSC_EN	RC32_OSC_DISABLE	The 32kHz RC Oscillator is disabled	0x0*
		RC32_OSC_ENABLE	The 32kHz RC Oscillator is enabled	0x1
7	RC32_TEMP_COMP_EN	RC32_TEMP_COMP_DISABLE	The 32kHz RC Oscillator temperature compensation is disabled	0x0*
		RC32_TEMP_COMP_ENABLE	The 32kHz RC Oscillator temperature compensation is enabled	0x1
6	RC32_FTRIM_ADJ	RC32_OSC_RANGE_NOM	The 32 kHz RC Oscillator frequency range is nominal	0x0*
		RC32_OSC_RANGE_M25	The 32 kHz RC Oscillator frequency range is lowered by 25%	0x1
5:0	RC32_FTRIM	RC32_OSC_M48	-48 % trimming	0x0
		RC32_OSC_M46P5	-46.5% trimming	0x1
		RC32_OSC_NOM	Nominal frequency	0x20*
		RC32_OSC_P46P5	+46.5% trimming	0x3F

7.5.0.6 ACS_XTAL32K_CTRL

Bit Field	Read/Write	Field Name	Description
26	W	XTAL_N_OK_RESET	Reset Xtal not ok sticky flag
25	R	XTAL_N_OK	Xtal not ready sticky flag
24	R	READY	Xtal ready status
18	RW	XIN_CAP_BYPASS_EN	Switch to bypass the added XIN serial cap to reduce the leakage
17	RW	EN_AMPL_CTRL	Xtal enable amplitude control (regulation)
16	RW	FORCE_READY	Xtal bypass the ready detector
13:8	RW	CLOAD_TRIM	Xtal load capacitance configuration

Bit Field	Read/Write	Field Name	Description
7:4	RW	ITRIM	Xtal current trimming
1	RW	IBOOST	Xtal current boosting (4x)
0	RW	ENABLE	Enable the Xtal 32 kHz oscillator

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
26	XTAL_N_OK_RESET	XTAL32K_NOT_OK_RESET	Xtal 32K not reset	0x1
25	XTAL_N_OK	XTAL_OK	Xtal 32K NOT OK sticky flag was not set	0x0*
		XTAL_N_OK	Xtal 32K NOT OK sticky flag was set	0x1
24	READY	XTAL32K_NOT_OK	Xtal 32K not available	0x0*
		XTAL32K_OK	Xtal 32K is OK	0x1
18	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
17	EN_AMPL_CTRL	XTAL32K_AMPL_CTRL_DISABLE	Xtal 32K amplitude control disabled	0x0
		XTAL32K_AMPL_CTRL_ENABLE	Xtal 32K amplitude control enabled	0x1*
16	FORCE_READY	XTAL32K_NOT_FORCE_READY	Xtal 32K ready not forced	0x0*
		XTAL32K_FORCE_READY	Xtal 32K ready forced	0x1
13:8	CLOAD_TRIM	XTAL32K_CTRIM_0P0PF	0 pF internal capacitor	0x0
		XTAL32K_CTRIM_0P4PF	0.4 pF internal capacitor	0x1
		XTAL32K_CTRIM_8P8PF	8.8 pF internal capacitor	0x16*
		XTAL32K_CTRIM_25P2PF	25.2 pF internal capacitor	0x3F
7:4	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
1	IBOOST	XTAL32K_IBOOST_DISABLE	Disable the Xtal 32 kHz current boosting mode	0x0*
		XTAL32K_IBOOST_ENABLE	Disable the Xtal 32 kHz enable boosting mode (4x itrim currents)	0x1
0	ENABLE	XTAL32K_DISABLE	Disable the Xtal 32 kHz oscillator	0x0*
		XTAL32K_ENABLE	Enable the Xtal 32 kHz oscillator	0x1

7.6 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 8.4 “Resets” on page 427.

The clock source monitored by the clock detector depends on the power mode (see Section 8.6 “Power Modes” on page 431), and the state of the system power supplies in that mode. The clock detector indicates 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 RSL15 SoC, selecting the following clock sources for each mode:

System startup
: RC oscillator (see Section 7.2.1 “RC Oscillator” on page 383)

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 7.4.7 “Power Supply Clocks” on page 395)

Sleep or Standby Mode
: RTC clock (see Section 7.4.5 “Real Time Clock (RTC)” on page 392)

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

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 caused by the clock detector, or to disable the clock detector itself, use the `ACS_CLK_DET_CTRL` register to perform these steps:

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 has triggered a reset.

7.7 Clock Detector Registers

Register Name	Register Description	Address
`ACS_CLK_DET_CTRL`	Clock Detector Configuration register	0x40001B58

7.7.0.1 ACS_CLK_DET_CTRL

Bit Field	Read/Write	Field Name	Description
8	R	CLOCK_PRESENT	Clock present flag
1	RW	RESET_IGNORE	Clock detector reset condition ignore
0	RW	ENABLE	Clock detector enable

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

CHAPTER 8

Power Components

The power supply is a critical component of the RSL15 system. Supplied power has significant effects on both RF and other types of system performance.

The components that make up the power supply include:

- Power supplies
- Power management, including management of several power modes
- Reset handling

Power supplies themselves can be divided into two types of supply voltages:

1. Power supply input voltages, described further in Section 8.2.1 “Power Supply Inputs” on page 407
2. Internal power supply voltages, described further in Section 8.2 “Power Supply Overview” on page 406

8.1 Power Management Unit

The power management unit contains the following:

- The power supplies, as described in Section 8.2 “Power Supply Overview” on page 406
- The power supervisory blocks and reset blocks, as described in Section 8.4 “Resets” on page 427
- Blocks that support configuration of the power supplies, and supervisory blocks for use in different power modes and under different wakeup conditions, as described in Section 8.6 “Power Modes” on page 431

8.2 Power Supply Overview

The power supply tree is powered by the system supply voltage (VCC), which is sourced from one of three 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
- Voltage supplied for the digital I/O pads, including the GPIO and debug port (SWJ-DP) pads (VDDO)

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 (VREF)
- 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 (VDDCRET, VDDMRET)
- A regulator used in Sleep Power Mode without Retention Mode supplies enabled (VDDACS)
- 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)
- A configurable regulated supply for the flash memory (VDDFLASH)

The "Power Supply Diagram" figure (Figure 36) illustrates the RSL15 power supply and related components at a high level.

The digital pins on the device, including the GPIOs, the SWIO & SWCLK pins, and the NRESET pin are also powered by another power supply, VDDO. VDDO is supplied externally to the dedicated VDDO pad.

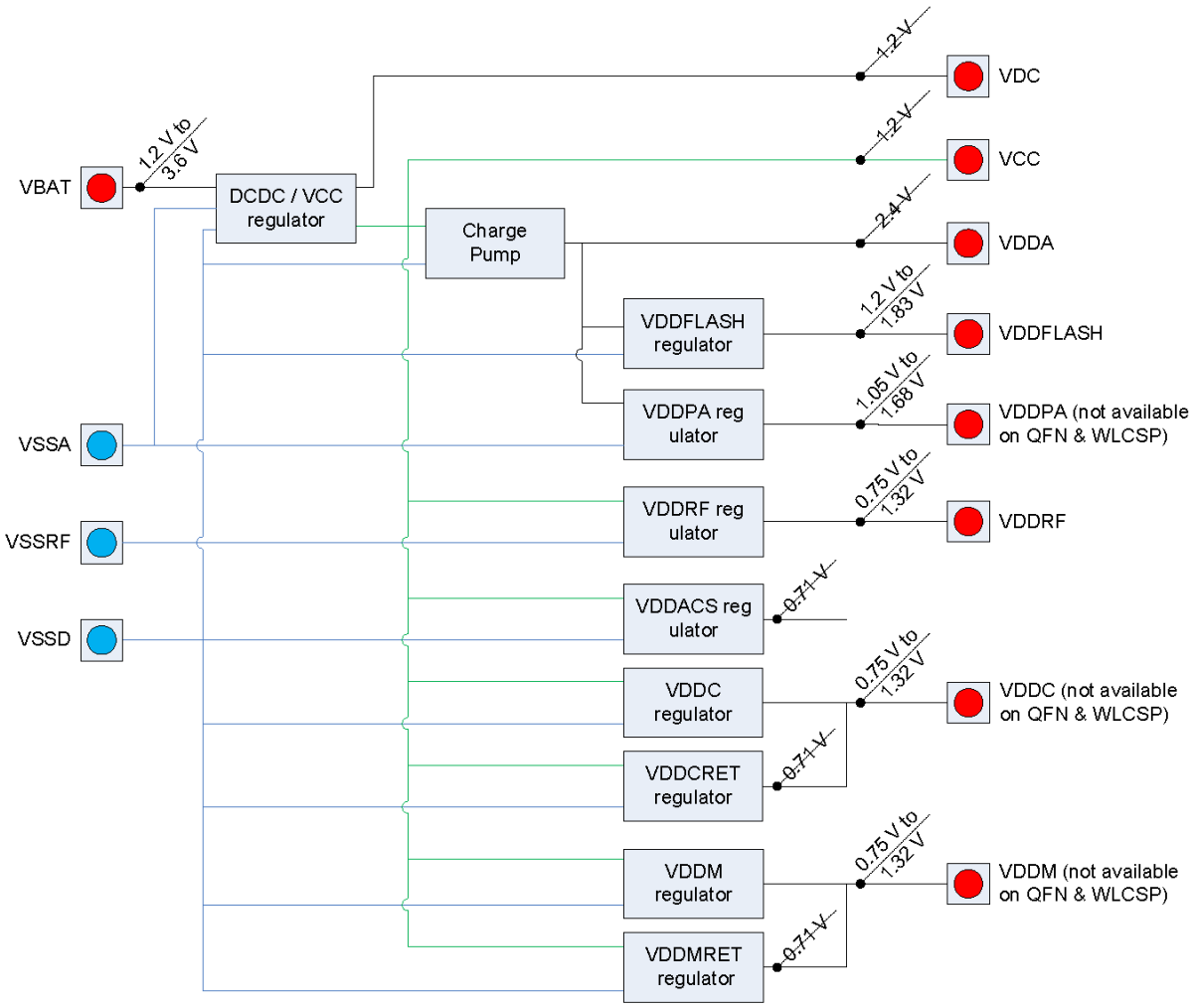


Figure 36. Power Supply Diagram

NOTE: VDDC, VDDM, and VDDPA are not available on all packages.

8.2.1 Power Supply Inputs

8.2.1.1 Battery Supply Voltage (VBAT)

The primary voltage supplied to an RSL15 chip is known as the battery supply voltage, because this voltage is supplied by a battery in a typical application.

VBAT directly supplies each of the regulators and charge pumps included in the RSL15 device. This supply voltage is monitored by the power management unit and the voltage monitor to ensure safe system operation, and to signal the current voltage level to user applications. See Section 8.1 “Power Management Unit” on page 406 for more information.

The minimum VBAT voltage required for basic operation is 1.2 V. The maximum VBAT voltage allowed before damage to the device can result is 3.6 V.

8.2.1.2 Digital Output Supply Voltage (VDDO)

The digital output supply voltages are attached to the I/O pads on the RSL15 device. These pads include:

- All GPIO pads, as described in Chapter 10 "General Purpose Input/Output" on page 511
- The NRESET input pad, described in Section 8.4 “Resets” on page 427
- The SWCLK and SWIO ports, as described in Chapter 10 "General Purpose Input/Output" on page 511 and Section 3.2 “Debug Port” on page 55

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 input is generally connected externally to VBAT or VDDA, based on the required voltages for digital communications using the associated I/O pads.

IMPORTANT: To minimize power consumption, the VDDO power domain needs to be powered at the lowest voltage possible.

The minimum VDDO voltage required for basic operation is 1.2 V. The maximum VDDO voltage allowed before damage to the device can result is 3.6 V.

8.2.2 Internal Power Supply Voltages

8.2.2.1 DC-DC Converter (VCC)

The System Supply Voltage (VCC) is used as the source for all of the internally generated supply voltages in the RSL15 SoC, and is supplied from VBAT. This power supply is used to reduce the battery voltage from a high voltage range (from 1.2 to 3.6 V) down to a supply voltage in the range from 1.0 to 1.31 V.

The voltage supplied by VCC is 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 the "VCC Supply Configuration" table (Table 11).

Table 11. VCC Supply Configuration

VCC Trim Configuration	VBAT Supply & 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 (Buck 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 RSL15 only uses VBAT or the internal LDO to supply VCC. An external VCC filtering capacitor is required in both modes. An external inductor is not required when using the LDO, as it is only used by the DC-DC converter when using the switching DC-DC operation.

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. Do not enable the DC-DC converter when the supplied VBAT is 1.4 V or less. When the supplied VBAT exceeds 1.4 V, the DC-DC converter can be enabled by setting the ACS_VCC_CTRL_BUCK_ENABLE bit from the ACS_VCC_CTRL register. Setting this bit configures the VCC to select Buck Mode. Use of the DC-DC converter requires both an external VCC filtering capacitor and the DC-DC converter's charge transferring inductor. In applications that only use the LDO Mode, the external inductor is not mandatory. For applications using a battery providing less than 1.4 V under any battery conditions, the recommendation is to only use the LDO Mode and not the Buck Mode.

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 using the CLK_DIV_CFG1_DCCLK_PRESCALE bit field from the CLK_DIV_CFG1 register. This prescaler provides a division of between 1 and 64 from SYSCLK, with a frequency defined by the following equation:

$$F_{DCCLK} = \frac{F_{SYSCLK}}{(CLK\_CFG\_DIV1\_DCCLK\_PRESCALE + 1)}$$

This clock needs to be configured for an update frequency of 4 MHz.

When the DC-DC converter is disabled, the CLK_DIV_CFG1_DCCLK_ENABLE bit-field from the CLK_DIV_CFG1 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). with a recommended default value of 1 (VCC_CONSTANT_IMAX). This setting selects between:
    - 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 RSL15 system. This configuration can only be used if VBAT exceeds VCC by 0.2 V.
    - 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 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 RSL15 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.

NOTE: Custom calibration settings for ICH_TRIM can be stored to NVR6, as described in the *RSL15 Firmware Reference* Resource Usage table. If available, these values are loaded by the sample code provided in the SDK.

- 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_CFG1_DCCLK_PRESCALE bit-field from the CLK_DIV_CFG1 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 results 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 RSL15 system, we recommend not using this mode for most use cases.

The converter has the following additional features:

- An activated output signal, which is high while the converter is in a charging/discharging cycle. This signal can be used to measure the converter output current but can only be monitored via the AOUT multiplexed output feature; see Chapter 10 "General Purpose Input/Output" on page 511 for more information.
- An overload output signal, which is set high if the peak inductor current goes above about 300 mA. This signal can be used to detect inductor saturation. When enabled through the ACS_WAKEUP_CFG_DCDC_OVERLOAD_EN control, the overload detection also generates a wakeup signal in Low Power Mode and interrupts the core, and the block automatically switches to LDO Mode. The corresponding ACS_WAKEUP_CTRL_DCDC_OVERLOAD_WAKEUP flag must be cleared to disable LDO Mode.
- In Sleep Mode, the DC-DC converter is not shut down, but regulates its output voltage to a lower level. Selection of Buck Mode or LDO Mode works the same way as it does in Run Mode. As the quiescent current of the Buck Mode is very low, the current savings are significant even for loads below 100 nA.

8.2.2.2 Internal Bandgap Reference Voltages

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 a nominal voltage of 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 and stored in NVR7 during device production, and use of this calibrated setting is recommended for all use cases and most operating conditions. The ROM loads and uses the calibrated bandgap trim setting for 0.75 V from NVR7.

IMPORTANT: The bandgap reference voltage increases at the extreme temperatures of the operating range. If there is an increase in the bandgap reference voltage, the system typically experiences an increased system current consumption.

To reduce this impact for devices that are continuously operating at a temperature extreme (below -30 °C and above +70 °C), the bandgap can be re-calibrated while operating at the temperature extreme. If at any point the device is no longer operating at an extreme temperature, we recommend reverting back to the calibrated setting stored in NVR7.

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

8.2.2.3 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 a nominal voltage of 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.

CAUTION: VDDRF must not exceed 1.32 V, or damage to the device can occur.

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. The "Voltage Supply Values and Target Voltages" table (Table 12) shows the image supply values for achieving certain TX power levels.

Table 12. Voltage Supply Values and Target Voltages

TX Power (dBm)	Source	Target Voltage (V)
0	VDDRF	1.10
1	VDDRF or VDDPA (low VCC cases)	1.15
2	VDDRF or VDDPA (low VCC cases)	1.20
3	VDDPA	1.29
4	VDDPA	1.39
5	VDDPA	1.52
6	VDDPA	1.66

In the ACS_VDDRF_CTRL register, the ACS_VDDRF_CTRL_READY bit configures 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. We recommend that some margin from VCC be kept, in order to ensure correct performance. This margin is typically 50 mV. For example, if VCC is set to 1.2 V, VDDRF ought to be set no higher than 1.15 V.

8.2.2.4 VDDPA

VDDPA is an optional RF TX power ampl ifier supply voltage. This block is used to provide a regulated voltage, trimmable from a nominal voltage of 1.1 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.

8.2.2.4.1 Connecting and Using VDDPA

When using the VDDPA regulator, care must be taken to ensure that VDDPA and VDDRF are connected in such a way that conflicts between the two supplies cannot result, thus avoiding damage to the device. The connections between VDDRF, VDDPA and the RF front-end (including the RF front end's power amplifier) are shown in Figure 37.

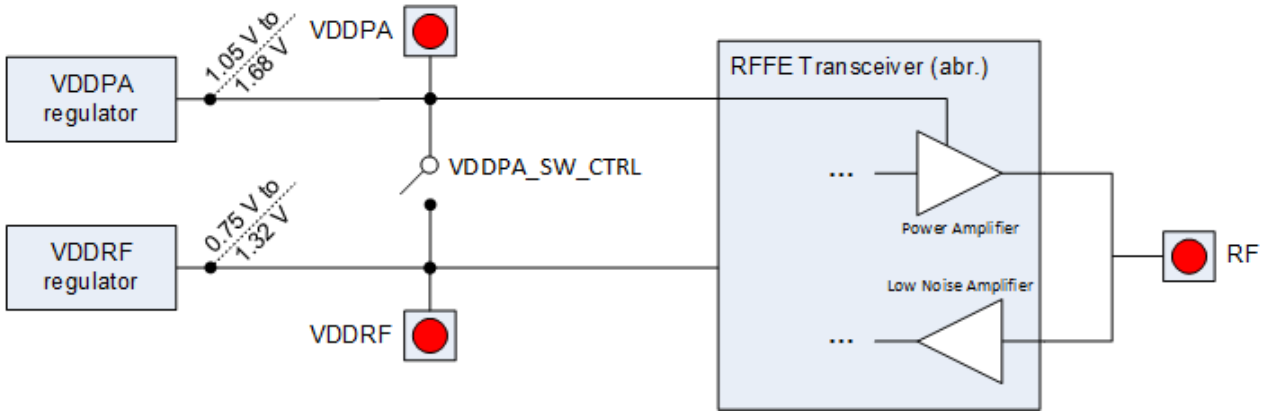


Figure 37. VDDRF and VDDPA powering the RFFE

When VDDPA is enabled (or in Dynamic Mode — see Section 8.2.2.4.2 “Dynamic VDDPA Control” on page 412), the ACS_VDDPA_CTRL_VDDPA_SW_CTRL bit must be reset, to set the switch in High Impedance Mode (which connects VDDPA to the RF power amplifier, if enabled). When using only VDDRF to power the RF power amplifier (VDDPA is disabled), close the switch by setting the VDDPA_CTRL_VDDPA_SW_CTRL bit.

8.2.2.4.2 Dynamic VDDPA Control

The RSL15 system is a dynamic VDDPA feature. The VDDPA dynamic control is enabled by setting the SYSCTRL_VDDPA_CFG0_DYNAMIC_CTRL bit in the SYSCTRL_VDDPA_CFG0 register. VDDPA itself must be disabled in order for the dynamic VDDPA feature to work correctly. If the VDDPA is enabled, it stays on consistently, bypassing the dynamic control feature. The purpose of the mechanism is twofold:

- To dynamically enable the VDDPA regulator during TX operations and keep it disabled otherwise, in order to minimize the power consumption. The mechanism automatically controls the enabling/switching of the VDDPA regulator when the ACS_VDDPA_CTRL_VDDPA_ENABLE bit is reset and the ACS_VDDPA_CTRL_VDDPA_SW_CTRL bit is set, both in the ACS_VDDPA_CTRL register.
- To mitigate possible out-of-band issues by applying a progressive voltage ramp up/down on the VDDPA regulator supplying the RF front-end power amplifier (PA). The mechanism dynamically controls the VDDPA voltage in the range specified by the ACS_VDDPA_CTRL_VDDPA_INITIAL_VTRIM and ACS_VDDPA_CTRL_VDDPA_VTRIM fields of the ACS_VDDPA_CTRL register.

The different voltage ramp up/down steps need to be properly synchronized with the RF front-end TX operations, as illustrated in the "Timing Diagram of the Dynamic VDDPA Operation" figure (Figure 38). All of the time durations and time delays mentioned in the figure are specified in system clock cycles.

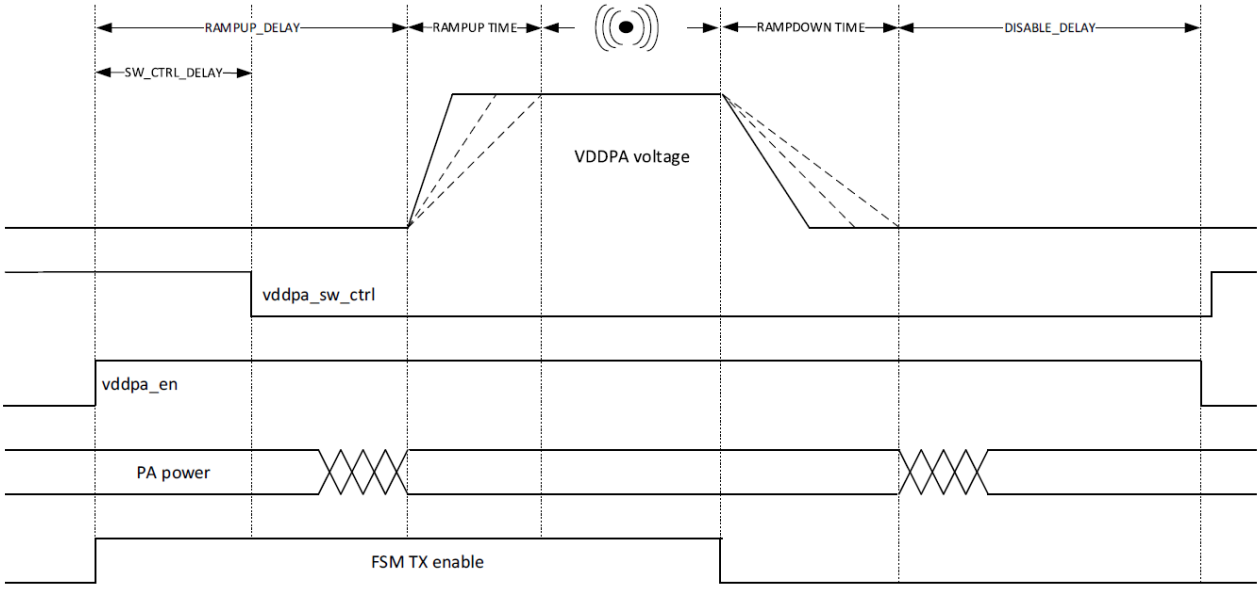


Figure 38. Timing Diagram of the Dynamic VDDPA Operation

When dynamic VDDPA control is enabled, the ramp up is characterized by several parameters that can be configured in SYSCTRL_VDDPA_CFG0 and SYSCTRL_VDDPA_CFG1.

SYSCTRL_VDDPA_CFG0_SW_CTRL_DELAY

This setting controls the delay before the power source for the RF power amplifier switches from VDDRF to VDDPA, once an RF transmission has been initiated. It is configured in the SYSCTRL_VDDPA_CFG0 register.

SYSCTRL_VDDPA_CFG0_RAMPUP_DELAY

This setting controls the delay before the VDDPA voltage ramp begins, once an RF transmission has been initiated. It is configured in the SYSCTRL_VDDPA_CFG0 register.

SYSCTRL_VDDPA_CFG1_RAMPUP_TIME

This setting is a multiplication of the values in the SYSCTRL_VDDPA_CFG1_RAMPUP_STEP and SYSCTRL_VDDPA_CFG1_RAMPUP_STEP_TIME bit fields from the SYSCTRL_VDDPA_CFG1 register. It represents the total time taken to ramp the VDDPA voltage from the value of the ACS_VDDPA_CTRL_INITIAL_VTRIM field up to the value of the ACS_VDDPA_CTRL_VTRIM setting in the same register.

SYSCTRL_VDDPA_CFG1_VDDPA_VOLTAGE

This setting is controlled by the ACS_VDDPA_CTRL_VTRIM field of the ACS_VDDPA_CTRL register. It is the final voltage that the VDDPA regulator reaches after the time value found in the SYSCTRL_VDDPA_CFG1_RAMPUP_TIME field of the SYSCTRL_VDDPA_CFG1 register. This voltage is used to supply the RF power amplifier during the actual RF transmission.

SYSCTRL_VDDPA_CFG1_RAMPDOWN_TIME

This setting is a multiplication of the values in the SYSCTRL_VDDPA_CFG1_RAMPDOWN_STEP and SYSCTRL_VDDPA_CFG1_RAMPDOWN_STEP_TIME fields from the SYSCTRL_VDDPA_CFG1 register. It represents the total time taken to ramp the VDDPA voltage down from the nominal trim setting (the value of the ACS_VDDPA_CTRL_TRIM field in the ACS_VDDPA_CTRL) to the initial trm setting (the value of the ACS_VDDPA_CTRL_INITIAL_VTRIM field in the same register).

SYSCTRL_VDDPA_CFG0_DISABLE_DELAY

This setting controls the total delay from the end of the time value in the SYSCTRL_VDDPA_CFG0_RAMPDOWN_TIME field of the SYSCTRL_VDDPA_CFG0 register, at which point the VDDPA is back to the voltage level value in the ACS_VDDPA_CTRL_INITIAL_VTRIM field from the ACS_VDDPA_CTRL register, to the point at which the dynamic VDDPA control ends and the RF power amplifier is switched back over to VDDRF (one cycle after VDDPA is disabled). This delay is configured in SYSCTRL_VDDPA_CFG0.

In order to avoid unexpected behavior, we recommend that you have the VDDPA initial trim value (in the ACS_VDDPA_CTRL_INITIAL_VTRIM field from the ACS_VDDPA_CTRL register) as close to the value of the ACS_VDDRF_CTRL_VTRIM bit from the ACS_VDDRF_CTRL register as possible, and that you only enable or disable dynamic VDDPA control when the VDDPA is disabled and the radio is idle.

8.2.2.5 Digital Supply Voltages

The RSL15 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 RSL15 system's digital components.
- VDDCRET replaces VDDC in power modes that use state retention of the RSL15 system's digital components.
- VDDM is the memory digital voltage that is used for the ROM and RAM memories in the RSL15 system, as well as the connections between the core and flash/GPIOs.
- VDDMRET replaces VDDM in power modes that use state retention of memories and memory-mapped elements of the RSL15 system.
- VDDT is the supply for the baseband timer used in Low Power Modes. When VDDT is enabled (by the ACS_VDDRET_CTRL_VDDTRET_EN bit from the ACS_VDDRET_CTRL register), the VDDT power switch connects VDDT to VDDCRET (when the VDDTRET regulator is enabled) or VCCACS (otherwise).

This block is used twice to provide two regulated voltages derived from the VCC supply. These supplies are trimmable from a nominal voltage of 0.75 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. The user must not use a voltage higher than 1.32 V, as this can result in damage to the device.

CAUTION: Voltage must not exceed 1.32 V, as damage to the device can occur.

NOTE: For divide packages where VDDC and VDDM pads are accessible, 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 (VDDCRET) 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 ACS_VDDM_CTRL (for Run Mode).

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

- The 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.
- The 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.
- The ACS_VDDRET_CTRL_VDDACSRET_VTRIM bit controls the VDDACS regulator voltage trimming value. ACS_VDDRET_CTRL_VDDTRET_ENABLE connects or disconnects VDDT to/from the VDDACS regulator.

8.2.2.6 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. 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, VDDCP, is shorted to VDDA.

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[1:0] bit in the ACS_VDDCP_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 7.4.3 “Slow Clock (SLOWCLK)” on page 391) using the CLK_DIV_CFG1_CPCLK_PRESCALE bit field from the CLK_DIV_CFG1 register. This prescaler provides a division of between 1 and 64 from SLOWCLK, with a frequency defined by the following equation:

$$F_{CPCLK} = \frac{F_{SLOWCLK}}{(CLK\_DIV\_CFG1\_CPCLK\_PRESCALE + 1)}$$

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_CFG1_CPCLK_ENABLE bit-field from the CLK_DIV_CFG1 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.

The charge pump can stay active during Sleep Mode as long as the STANDBYCLK is active in Sleep Mode. To enable the charge pump before entering Sleep Mode, set the VDDCP_ENABLE bit in the ACS_SLEEP_MODE_CFG register. The frequency for the charge pump in Sleep Mode can be configured in the ACS_VDDCP_CTRL_CPCLK_FREQ field in the ACS_VDDCP_CTRL register, with the maximum setting being 32.768 kHz.

8.2.2.7 Charge Pump Supply Voltage (VDDCP)

The VDDCP charge pump is clocked using CPCLK, which is divided from SYSCLK using the CLK_DIV_CFG1_CPCLK_PRESCALE bit-field from the CLK_DIV_CFG1 register.

IMPORTANT: Important: When VDDPA is enabled, the CPCLK frequency must be configured to a frequency other than 125 kHz for optimal RF performance (otherwise transmit modulation failures may occur).

8.2.2.8 Flash Memory Supply Voltage

The VDDFLASH regulator is used to provide a regulated voltage from 0.75 to 2.3 V in 25 mV steps. The voltage can be adjusted using the ACS_VDDFLASH_CTRL_VTRIM field in the ACS_VDDFLASH_CTRL register. When the regulator is disabled, the output pin can also be driven by an external source, and the external VDDFLASH is in a high impedance state, allowing the external source to provide power. The regulator is enabled or disabled by the ACS_VDDFLASH_CTRL_ENABLE bit in the ACS_VDDFLASH_CTRL register. An external decoupling capacitor is required on the VDDFLASH pad. We highly recommended using the default 1.6 V setting for proper flash operation.

A current limiter can be enable to limit the maximum output current. Two current limit settings are available: one to reduce the short-circuit current to 70 mA, and a second one to reduce the inrush current during boot-up to 4 mA. The default setting is 4 mA and can be configured by the ACS_VDDFLASH_CTRL_SOFT_START bit from the ACS_VDDFLASH_CTRL register. The current limiter functionality can be configured by the ACS_VDDFLASH_CTRL_ENABLE_LIMITER bit from the same register.

The circuit has a comparator to indicate when the output voltage reaches 90% of the target value. This comparator feedback can be connected to the reset circuitry to assert a reset if necessary. This functionality can be enabled or disabled by the ACS_VDDFLASH_CTRL_MASK_READY bit from the ACS_VDDFLASH_CTRL register.

8.3 Power Supply Registers

VCC and DC_DC Converter Registers

Register Name	Register Description	Address
ACS_VCC_CTRL	DC-DC / LDO Supply Configuration / Control register	0x40001B04

Bandgap Converter Registers

Register Name	Register Description	Address
ACS_BG_CTRL	Bandgap Control register	0x40001B00

RF Block Registers

Register Name	Register Description	Address
ACS_VDDRF_CTRL	RF Regulator control register	0x40001B18
ACS_VDDPA_CTRL	RF Block Regulator Configuration / Control register	0x40001B14

Digital Supply Registers

Register Name	Register Description	Address
ACS_VDDC_CTRL	Digital Core Voltage Regulator control register	0x40001B0C
ACS_VDDM_CTRL	Digital Core Voltage Regulator control register	0x40001B10
ACS_VDDRET_CTRL	Retention Regulator control register	0x40001B20

Charge Pump Control Registers

Register Name	Register Description	Address
ACS_VDDCP_CTRL	Charge Pump control register	0x40001B08

VDDFLASH Block Regulator Control Registers

Register Name	Register Description	Address
ACS_VDDFLASH_CTRL	VDDFLASH Block Regulator Configuration / Control register	0x40001B1C

WEDAC Control Registers

Register Name	Register Description	Address
ACS_WEDAC_CTRL	WEDAC Control register	0x40001B38

8.3.0.1 ACS_VCC_CTRL

Bit Field	Read/Write	Field Name	Description
24	R	READY	Supply ready (only makes sense in test mode)
21:16	RW	VTRIM_LIMIT	Max VTRIM value that can be used. If VTRIM value is greater it will be limited to this value
15:12	RW	ICH_TRIM	Inductor charge current trimming
11	RW	CCM_ENABLE	Enable CCM mode
10	RW	PULSE_CTRL	Pulse mode control
9	RW	CHARGE_CTRL	Charge mode control
8	RW	BUCK_ENABLE	Enable buck converter mode
5:0	RW	VTRIM	Output voltage trimming configuration in 10 mV steps

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
24	READY	VCC_NOT_READY	Supply voltage not ready	0x0*
		VCC_READY	Supply voltage ready	0x1
21:16	VTRIM_LIMIT	VCC_TRIM_LIMIT_1P00V	Output voltage limit 1.00V	0x0

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		VCC_TRIM_LIMIT_1P05V	Output voltage limit 1.05V	0x5
		VCC_TRIM_LIMIT_1P10V	Output voltage limit 1.10V	0xA
		VCC_TRIM_LIMIT_1P15V	Output voltage limit 1.15V	0xF
		VCC_TRIM_LIMIT_1P20V	Output voltage limit 1.20V	0x14
		VCC_TRIM_LIMIT_1P25V	Output voltage limit 1.25V	0x19
		VCC_TRIM_LIMIT_1P31V	Output voltage limit 1.31V (This should be the max value allowed)	0x1F*
		VCC_TRIM_LIMIT_1P35V	Output voltage limit 1.35V (This is the max value allowed only during RF operation at max output power)	0x23
		VCC_TRIM_LIMIT_1P63V	Output voltage limit 1.63V	0x3F
15:12	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
11	CCM_ENABLE	VCC_DCM_MODE	Discontinous current mode	0x0*
		VCC_CCM_MODE	Continuous current mode enabled	0x1
10	PULSE_CTRL	VCC_SINGLE_PULSE	Single pulse per clock cycle	0x0*
		VCC_MULTI_PULSE	Multi pulses enabled (until VCC > VCC_ TRIM)	0x1
9	CHARGE_CTRL	VCC_CONSTANT_CHARGE	Constant charge transfer (valid for VBAT > VCC+0.2)	0x0
		VCC_CONSTANT_IMAX	Constant inductor maximum charge current	0x1*
8	BUCK_ENABLE	VCC_LDO	Linear mode	0x0*
		VCC_BUCK	Buck converter mode enabled	0x1
5:0	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*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		VCC_TRIM_1P25V	Output voltage 1.25V	0x19
		VCC_TRIM_1P31V	Output voltage 1.31V (This should be the max value allowed)	0x1F
		VCC_TRIM_1P35V	Output voltage 1.35V (This is the absolute max value allowed, only for RF operation at max output power)	0x23
		VCC_TRIM_1P63V	Output voltage 1.63V	0x3F

8.3.0.2 ACS_BG_CTRL

Bit Field	Read/Write	Field Name	Description
31	R	READY	Bandgap ready
28:24	RW	SLOPE_ITRIM	Current temperature coefficient trimming
21:16	RW	ITRIM	Reference current trimming
13	RW	SEL_VTRIM_SRC	Select trim source for SLOPE_VTRIM and VTRIM
12:8	RW	SLOPE_VTRIM	Voltage temperature coefficient trimming
5:0	RW	VTRIM	Reference voltage trimming (5 mV steps)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31	READY	BG_NOT_READY	Bandgap reference not ready	0x0*
		BG_READY	Bandgap reference ready	0x1
28:24	SLOPE_ITRIM	BG_SLOPE_ITRIM_VALUE	Current temperature dependency 0 ppm/C	0x17*
21:16	ITRIM	BG_ITRIM_0P64UA	0.64 uA	0x0
		BG_ITRIM_0P65UA	0.65 uA	0x1
		BG_ITRIM_0P99UA	0.99 uA	0x23
		BG_ITRIM_1P00UA	1.00 uA	0x24*
		BG_ITRIM_1P01UA	1.01 uA	0x25
		BG_ITRIM_1P26UA	1.26 uA	0x3E
		BG_ITRIM_1P27UA	1.27 uA	0x3F
13	SEL_VTRIM_SRC	BG_VTRIM_SEL_REGISTER	Select register value	0x0
		BG_VTRIM_SEL_METAL	Select metal value	0x1*
12:8	SLOPE_VTRIM	BG_SLOPE_VTRIM_VALUE	Voltage temperature dependency 0 ppm/C	0x18*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
5:0	VTRIM	BG_VTRIM_MIN	bandgap voltage minimum trimming	0x0
		BG_VTRIM_DEFAULT	bandgap voltage default trimming	0x15*
		BG_VTRIM_MAX	bandgap voltage maximum trimming	0x3F

8.3.0.3 ACS_VDDRF_CTRL

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

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
24	READY	VDDRF_NOT_READY	Supply voltage not ready	0x0*
		VDDRF_READY	Supply voltage ready	0x1
12	CLAMP	VDDRF_DISABLE_HIZ	Set the output HIZ (floating) in disable mode	0x0*
		VDDRF_DISABLE_GND	Clamp output to ground in disable mode	0x1
8	ENABLE	VDDRF_DISABLE	Disable the VDDRF regulator	0x0*
		VDDRF_ENABLE	Enable the VDDRF regulator	0x1
5:0	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_1P20V	1.2 V	0x2D
		VDDRF_TRIM_1P32V	1.32 V	0x39
		VDDRF_TRIM_1P38V	1.38 V	0x3F

8.3.0.4 ACS_VDDPA_CTRL

Bit Field	Read/Write	Field Name	Description
19:16	RW	INITIAL_VTRIM	Initial output voltage trimming configuration in 10 mV steps
12	RW	VDDPA_SW_CTRL	Power amplifier supply control
9	RW	ENABLE_ISENSE	Enable current sensing circuit
8	RW	ENABLE	Enable control
5:0	RW	VTRIM	Output voltage trimming configuration in 10 mV steps

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
19:16	INITIAL_VTRIM	VDDPA_INITIAL_TRIM_1P05V	1.05 V	0x0*
		VDDPA_INITIAL_TRIM_1P10V	1.10 V	0x5
		VDDPA_INITIAL_TRIM_1P20V	1.20 V	0xF
12	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
9	ENABLE_ISENSE	VDDPA_ISENSE_DISABLE	Disable the VDDPA regulator current sensing circuit	0x0*
		VDDPA_ISENSE_ENABLE	Enable the VDDPA regulator current sensing circuit	0x1
8	ENABLE	VDDPA_DISABLE	Disable the VDDPA regulator	0x0*
		VDDPA_ENABLE	Enable the VDDPA regulator	0x1
5:0	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

8.3.0.5 ACS_VDDC_CTRL

Bit Field	Read/Write	Field Name	Description
24	R	READY	Supply ready
21:16	RW	STANDBY_VTRIM	VDDC standby voltage trimming (10 mV steps)
12	RW	ENABLE_LOW_BIAS	Low power mode control
8	RW	SLEEP_CLAMP	Sleep mode clamp control
5:0	RW	VTRIM	Output voltage trimming configuration in 10 mV steps

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
24	READY	VDDC_NOT_READY	Supply voltage not ready	0x0*
		VDDC_READY	Supply voltage ready	0x1
21:16	STANDBY_VTRIM	VDDC_STANDBY_TRIM_0P75V	0.75 V	0x0
		VDDC_STANDBY_TRIM_0P76V	0.76 V	0x1
		VDDC_STANDBY_TRIM_1P20V	1.2 V	0x2D*
		VDDC_STANDBY_TRIM_1P32V	1.32 V	0x39
		VDDC_STANDBY_TRIM_1P38V	1.38 V	0x3F
12	ENABLE_LOW_BIAS	VDDC_NOMINAL_BIAS	Nominal regulator biasing	0x0*
		VDDC_LOW_BIAS	Low regulator biasing	0x1
8	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
5:0	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_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

8.3.0.6 ACS_VDDM_CTRL

Bit Field	Read/Write	Field Name	Description
24	R	READY	Supply ready
21:16	RW	STANDBY_VTRIM	VDDM standby voltage trimming (10 mV steps)
12	RW	ENABLE_LOW_BIAS	Low power mode control
8	RW	SLEEP_CLAMP	Sleep mode clamp control
5:0	RW	VTRIM	Output voltage trimming configuration in 10 mV steps

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
24	READY	VDDM_NOT_READY	Supply voltage not ready	0x0*
		VDDM_READY	Supply voltage ready	0x1
21:16	STANDBY_VTRIM	VDDM_STANDBY_TRIM_0P75V	0.75 V	0x0
		VDDM_STANDBY_TRIM_0P76V	0.76 V	0x1
		VDDM_STANDBY_TRIM_1P20V	1.2 V	0x2D*
		VDDM_STANDBY_TRIM_1P32V	1.32 V	0x39
		VDDM_STANDBY_TRIM_1P38V	1.38 V	0x3F
12	ENABLE_LOW_BIAS	VDDM_NOMINAL_BIAS	Nominal regulator biasing	0x0*
		VDDM_LOW_BIAS	Low regulator biasing	0x1
8	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
5:0	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

8.3.0.7 ACS_VDDRET_CTRL

Bit Field	Read/Write	Field Name	Description
18:17	RW	VDDMRET_VTRIM	VDDMRET retention regulator voltage trimming
16	RW	VDDMRET_ENABLE	Enable/Disable the VDDMRET retention regulator
10:9	RW	VDDACS_VTRIM	VDDACS regulator voltage trimming
8	RW	VDDTRET_ENABLE	Enables the VDDT retention power (activate switch from VDDACS)
2:1	RW	VDDCRET_VTRIM	VDDCRET Retention regulator voltage trimming
0	RW	VDDCRET_ENABLE	Enables the VDDCRET Retention regulator

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
18:17	VDDMRET_VTRIM	VDDMRET_TRIM_VALUE	VDDMRET trimming value	0x3*
16	VDDMRET_ENABLE	VDDMRET_DISABLE	The VDDMRET retention regulator is disabled	0x0*
		VDDMRET_ENABLE	The VDDMRET retention regulator is enabled	0x1
10:9	VDDACS_VTRIM	VDDACS_TRIM_VALUE	VDDACS trimming value	0x3*
8	VDDTRET_ENABLE	VDDTRET_DISABLE	Disable BB timer retention supply (VDDT)	0x0*
		VDDTRET_ENABLE	Enable BB timer retention supply (VDDT)	0x1
2:1	VDDCRET_VTRIM	VDDCRET_TRIM_VALUE	VDDCRET trimming value	0x3*
0	VDDCRET_ENABLE	VDDCRET_DISABLE	The VDDCRET retention regulator is disabled	0x0*
		VDDCRET_ENABLE	The VDDCRET retention regulator is enabled	0x1

8.3.0.8 ACS_VDDCP_CTRL

Bit Field	Read/Write	Field Name	Description
24	R	READY	Supply ready
13	RW	COMP_ENABLE	Force cp comparator enable
11:8	RW	CPCLK_FREQ	Charge Pump clock frequency during power down modes
1:0	RW	PTRIM	Output power trimming

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
24	READY	VDDCP_NOT_READY	Charge pump voltage not ready	0x0*
		VDDCP_READY	Charge pump voltage ready	0x1
13	COMP_ENABLE	VDDCP_COMP_AUTO	Comparator and resistive divider turned off when not required	0x0
		VDDCP_COMP_ENABLED	Comparator and resistive divider turned on (retro-compatible)	0x1*
11:8	CPCLK_FREQ	VDDCP_CPCLK_32KHZ	CP clock = 32 kHz (WDG_SoC clock)	0x0
		VDDCP_CPCLK_16KHZ	CP clock = 16 kHz	0x1
		VDDCP_CPCLK_8KHZ	CP clock = 8 kHz	0x2*
		VDDCP_CPCLK_4KHZ	CP clock = 4 kHz	0x3
		VDDCP_CPCLK_2KHZ	CP clock = 2 kHz	0x4
		VDDCP_CPCLK_1KHZ	CP clock = 1 kHz	0x5
		VDDCP_CPCLK_0P5KHZ	CP clock = 512 Hz	0x6
		VDDCP_CPCLK_0P25KHZ	CP clock = 256 Hz	0x7
		VDDCP_CPCLK_0P12KHZ	CP clock = 128 Hz	0x8
1:0	PTRIM	VDDCP_PTRIM_4MA	CP max current to 4 mA	0x0
		VDDCP_PTRIM_8MA	CP max current to 8 mA	0x1
		VDDCP_PTRIM_12MA	CP max current to 12 mA	0x2
		VDDCP_PTRIM_16MA	CP max current to 16 mA	0x3*

8.3.0.9 ACS_VDDFLASH_CTRL

Bit Field	Read/Write	Field Name	Description
24	R	READY	Supply ready
11	RW	MASK_READY	Apply a mask to reset logic to ignore VDDFLASH ready signal
10	RW	SOFT_START	Current limiter threshold setting
9	RW	ENABLE_LIMITER	Enable current limiter circuit
8	RW	ENABLE	Enable control
5:0	RW	VTRIM	Output voltage trimming configuration in 25 mV steps

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
24	READY	VDDFLASH_NOT_READY	Supply voltage not ready	0x0*
		VDDFLASH_READY	Supply voltage ready	0x1
11	MASK_READY	VDDFLASH_USE_READY	Reset logic use VDDFLASH ready	0x0*
		VDDFLASH_MASK_READY	Reset ignore VDDFLASH ready	0x1
10	SOFT_START	VDDFLASH_LIMITER_70MA	Limiter current threshold at 70mA	0x0
		VDDFLASH_LIMITER_4MA	Limiter current threshold at 4mA	0x1*
9	ENABLE_LIMITER	VDDFLASH_LIMITER_DISABLE	Disable the VDDFLASH regulator current limiter circuit	0x0
		VDDFLASH_LIMITER_ENABLE	Enable the VDDFLASH regulator current limiter circuit	0x1*
8	ENABLE	VDDFLASH_DISABLE	Disable the VDDFLASH regulator	0x0
		VDDFLASH_ENABLE	Enable the VDDFLASH regulator	0x1*
5:0	VTRIM	VDDFLASH_TRIM_0P750V	0.750V	0x0
		VDDFLASH_TRIM_0P775V	0.775V	0x1
		VDDFLASH_TRIM_1P725V	1.725V	0x27
		VDDFLASH_TRIM_1P750V	1.750V	0x28*
		VDDFLASH_TRIM_1P775V	1.775V	0x29
		VDDFLASH_TRIM_2P300V	2.300V	0x3E
		VDDFLASH_TRIM_2P325V	2.325V	0x3F

8.3.0.10 ACS_WEDAC_CTRL

Bit Field	Read/Write	Field Name	Description
5:0	RW	WEDAC	WEDAC setting

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
5:0	WEDAC	WEDAC_0P000V	WEDAC = 0.000V	0x0
		WEDAC_0P016V	WEDAC = 0.016V	0x1
		WEDAC_0P032V	WEDAC = 0.032V	0x2
		WEDAC_0P496V	WEDAC = 0.496V	0x1F
		WEDAC_0P504V	WEDAC = 0.504V	0x20
		WEDAC_0P600V	WEDAC = 0.600V	0x26*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		WEDAC_0P616V	WEDAC = 0.616V	0x27
		WEDAC_0P984V	WEDAC = 0.984V	0x3E
		WEDAC_1P000V	WEDAC = 1.000V	0x3F

8.4 RESETS

The RSL15 SoC contains a variety of reset sources that can be used to reset the entire RSL15 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 RESET_DIG_STATUS registers. The reset bits and their encoding can be seen in the "Reset Sources and Flag Decoding" figure (Figure 39), 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 RESET_DIG_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.

The NRESET pad can be used to reset the device. This pad is connected by an internal pullup resistor to VDDO. Therefore, we recommend that VDDO is powered in all your applications. Otherwise the NRESET pin floats, and unexpected behavior of the device can result.

ACS_RESET_STATUS										RESET_DIG_STATUS					
POR_RESET_FLAG	PAD_RESET_FLAG	VDDC_RESET_FLAG	VDDM_RESET_FLAG	VDDFLASH_RESET_FLAG	BGVREF_RESET_FLAG	CLK_DET_RESET_FLAG	TIMEOUT_RESET_FLAG	WRONG_STATE_RESET_FLAG	CCAO_REBOOT_RESET_FLAG	ACS_RESET_FLAG	CPU_SW_RESET_FLAG	WATCHDOG_RESET_FLAG	LOCKUP_FLAG	DEU_RESET_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 VDDFLASH
0	-	-	-	-	1	-	-	-	-	1	-	-	-	-	Reset due to BG-Vref
0	-	-	-	-	-	1	-	-	-	1	-	-	-	-	Reset due to system clock detector
0	-	-	-	-	-	-	1	-	-	1	-	-	-	-	Reset due to power state machine timeout
0	-	-	-	-	-	-	-	1	-	1	-	-	-	-	Reset due to a wrong power mode state
0	-	-	-	-	-	-	-	-	1	1	-	-	-	-	Reset due to a power off-on of the CryptoCell-AO
0	0	0	0	0	0	0	0	0	0	1	-	-	-	-	Reset due to wake-up from sleep/storage mode, in sleep mode the reset occurs only when VDDC retention is disabled.
-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	Reset due to the CPU system reset
-	-	-	-	-	-	-	-	-	-	-	-	1	0	-	Reset due to the Core watchdog with the CPU not in lockup state
-	-	-	-	-	-	-	-	-	-	-	-	1	1	-	Reset due to the Core watchdog with the CPU in lockup state
-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	Reset due to the DEU
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	JTRST pad

Figure 39. Reset Sources and Flag Decoding

IMPORTANT: For best practices in error, fault, and watchdog interrupt handling see Chapter 1 "Diagnostic Strategies" on page 1 from the *RSL15 Developer's Guide*.

8.5 RESETS REGISTERS

Register Name	Register Description	Address
ACS_RESET_STATUS	ACS Reset Source Status Register	0x40001B78

8.5.0.1 ACS_RESET_STATUS

Bit Field	Read/Write	Field Name	Description
25	R	CCAO_REBOOT_RESET_FLAG	Sticky flag that detects that a CryptoCell Alway ON reboot reset occurred
24	R	WRONG_STATE_RESET_FLAG	Sticky flag that detects that a wrong state reset occurred
23	R	TIMEOUT_RESET_FLAG	Sticky flag that detects that a timeout in the power up sequence occurred
22	R	CLK_DET_RESET_FLAG	Sticky flag that detects that a clock detector reset occurred
21	R	VDDFLASH_RESET_FLAG	Sticky flag that detects that a VDDFLASH reset occurred (triggered by VDDFLASH_ready = 0)
20	R	VDDM_RESET_FLAG	Sticky flag that detects that a VDDM reset occurred (triggered by VDDM_ready = 0)
19	R	VDDC_RESET_FLAG	Sticky flag that detects that a VDDC reset occurred (triggered by VDDC_ready = 0)
18	R	BG_VREF_RESET_FLAG	Sticky flag that detects that a Bandagap reference voltage reset occurred
17	R	PAD_RESET_FLAG	Sticky flag that detects that a reset occurred due to pad NRESET
16	R	POR_RESET_FLAG	Sticky flag that detects that a POR reset occurred
9	W	CCAO_REBOOT_RESET_FLAG_CLEAR	Reset the sticky CCAO_REBOOT_RESET flag.
8	W	WRONG_STATE_RESET_FLAG_CLEAR	Reset the sticky WRONG_STATE_RESET flag.
7	W	TIMEOUT_RESET_FLAG_CLEAR	Reset the sticky TIMEOUT_RESET flag.
6	W	CLK_DET_RESET_FLAG_CLEAR	Reset the sticky CLK_DET_RESET flag.
5	W	VDDFLASH_RESET_FLAG_CLEAR	Reset the sticky VDDFLASH_RESET flag.
4	W	VDDM_RESET_FLAG_CLEAR	Reset the sticky VDDM_RESET flag.
3	W	VDDC_RESET_FLAG_CLEAR	Reset the sticky VDDC_RESET flag.
2	W	BG_VREF_RESET_FLAG_CLEAR	Reset the sticky BG_VREF_RESET flag.
1	W	PAD_RESET_FLAG_CLEAR	Reset the sticky PAD_RESET flag.
0	W	POR_RESET_FLAG_CLEAR	Reset the sticky POR_RESET flag.

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
25	CCAO_REBOOT_ RESET_FLAG	CCAO_REBOOT_RESET_FLAG_ NOT_SET	The CryptoCell Always ON reboot reset has not triggered at least once	0x0
		CCAO_REBOOT_RESET_FLAG_SET	The CryptoCell Always ON reboot reset was triggered at least once since this status bit was last cleared	0x1*
24	WRONG_STATE_ RESET_FLAG	WRONG_STATE_RESET_FLAG_ NOT_SET	The wrong state reset has not triggered at least once	0x0
		WRONG_STATE_RESET_FLAG_SET	The wrong state reset was triggered at least once since this status bit was last cleared	0x1*
23	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*
22	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*
21	VDDFLASH_RESET_ FLAG	VDDFLASH_RESET_FLAG_NOT_ SET	The VDDFLASH reset has not triggered at least once	0x0
		VDDFLASH_RESET_FLAG_SET	The VDDFLASH reset was triggered at least once since this status bit was last cleared	0x1*
20	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*
19	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*
18	BG_VREF_RESET_FLAG	BG_VREF_RESET_FLAG_NOT_SET	The band-gap reference voltage reset has not triggered at least once	0x0
		BG_VREF_RESET_FLAG_SET	The band-gap reference voltage was triggered at least once since this status bit was last cleared	0x1*
17	PAD_RESET_FLAG	PAD_RESET_FLAG_NOT_SET	The NRESET pad reset has not triggered	0x0

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
			at least once	
		PAD_RESET_FLAG_SET	The NRESET pad reset was triggered at least once since this status bit was last cleared	0x1*
16	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*
9	CCAO_REBOOT_ RESET_FLAG_CLEAR	CCAO_REBOOT_RESET_FLAG_ CLEAR	Reset the sticky CCAO_REBOOT_ RESET flag.	0x1
8	WRONG_STATE_ RESET_FLAG_CLEAR	WRONG_STATE_RESET_FLAG_ CLEAR	Reset the sticky WRONG_STATE_ RESET flag.	0x1
7	TIMEOUT_RESET_ FLAG_CLEAR	TIMEOUT_RESET_FLAG_CLEAR	Reset the sticky TIMEOUT_RESET flag.	0x1
6	CLK_DET_RESET_ FLAG_CLEAR	CLK_DET_RESET_FLAG_CLEAR	Reset the sticky CLK_DET_RESET flag.	0x1
5	VDDFLASH_RESET_ FLAG_CLEAR	VDDFLASH_RESET_FLAG_CLEAR	Reset the sticky VDDFLASH_RESET flag.	0x1
4	VDDM_RESET_FLAG_ CLEAR	VDDM_RESET_FLAG_CLEAR	Reset the sticky VDDM_RESET flag.	0x1
3	VDDC_RESET_FLAG_ CLEAR	VDDC_RESET_FLAG_CLEAR	Reset the sticky VDDC_RESET flag.	0x1
2	BG_VREF_RESET_ FLAG_CLEAR	BG_VREF_RESET_FLAG_CLEAR	Reset the sticky BG_VREF_RESET flag.	0x1
1	PAD_RESET_FLAG_ CLEAR	PAD_RESET_FLAG_CLEAR	Reset the sticky PAD_RESET flag.	0x1
0	POR_RESET_FLAG_ CLEAR	POR_RESET_FLAG_CLEAR	Reset the sticky POR_RESET flag.	0x1

8.6 Power Modes

RSL15 has three main power modes: Run Mode, Sleep Mode, and Standby Mode. Run Mode is the default functional mode. Sleep Mode and Standby Mode power down and/or reconfigure subsystems to achieve lower system-level power consumption; they are generally categorized as Low Power Modes.

An overview of the power modes and operating modes is shown in the "Power Modes" table (Table 13). States and configurability of subsystems in each power mode are summarized in the "Power Mode Subsystems" table (Table 14).

Table 13. Power Modes

Power Mode	Description
Run Mode	Processor, RF subsystem, and memory are powered as needed by the user application – clocks are active, all peripherals available. Idle mode, a subset of run mode, configures the system as though entering standby mode, where the Arm Cortex-M33 processor is held waiting for an interrupt or event instead of actually entering standby mode.
Sleep Mode	The lowest power mode. Processor and RF subsystem are powered down and not clocked. Only selected wakeup sources are powered. Memory retention (and amount of memory retained) is optional. Some peripherals are available in Sleep Mode. On wakeup, the ROM restores the system before program execution begins. Smart Sense Mode is a subset of Sleep Mode, highlighted by the use of the ultra low power data acquisition subsystem in a configuration that spans both Sleep and Run Modes.
Standby Mode	A low power mode with faster wakeup time than Sleep Mode. Processor and RF subsystem are powered with lower voltage and not clocked. Only selected wakeup sources are powered. Memory retention (and amount of memory retained) is configurable. Some peripherals are available in Standby Mode. On wakeup, the program execution is resumed (i.e., the instruction right after the LWFI statement is executed).

Table 14. Power Mode Subsystems

Sub-Blocks		Power Modes		
		Run Mode	**Sleep Mode**	**Standby Mode**
Core	Arm Cortex-M33 Processor	Clock, normal voltage	No clock, low voltage or OFF	No clock, low voltage
	Arm Cortex-M33 Processor FPU	Clock, normal voltage or OFF	No clock, low voltage or OFF	No clock, low voltage or OFF
	Arm Cortex-M33 Processor Debug	Clock, normal voltage or OFF	No clock, low voltage or OFF	No clock, low voltage or OFF
	Baseband	Clock, normal voltage or OFF	No clock, low voltage or OFF	No clock, low voltage or OFF
	Baseband timer	Clock, normal voltage	Clock, low voltage or No clock, OFF	Clock, low voltage
	Arm CryptoCell-312 (+ its memories)	Clock, normal voltage or OFF	No clock, low voltage or OFF	No clock, low voltage or OFF
	Arm CryptoCell-312 Always On	Clock, normal voltage	No clock, low voltage or OFF	No clock, low voltage
	Individual RAM instances	Clock, normal voltage or OFF	No clock, low voltage retention mode or OFF	No clock, low voltage retention mode
	Flash instance	Clock, normal voltage or OFF	No clock, OFF	No clock, low voltage, OFF
ACS	Analog registers	Active	Active	Active
	Wakeup GPIOs	SW programmable	SW programmable	SW programmable
	RTC clock	SW programmable	SW programmable	SW programmable

Table 14. Power Mode Subsystems (Continued)

Sub-Blocks		Power Modes		
		Run Mode	**Sleep Mode**	**Standby Mode**
Analog	Bandgap	ON	SW programmable	ON
	VCC regulator	ON	SW programmable	ON
	VDDC regulator	Normal voltage	OFF	Low voltage
	VDDM regulator	Normal voltage	OFF	Low voltage
	VDDC retention regulator	SW programmable	SW programmable	SW programmable
	VDDM retention regulator	SW programmable	SW programmable	SW programmable
	VDDCP charge pump	ON	SW programmable	ON
	VDDFLASH regulator	ON	OFF	SW programmable
	RF regulator	SW programmable	OFF	SW programmable
	PA regulator	SW programmable	OFF	SW programmable
	Temp sensor	SW programmable	OFF	OFF
	Current source	SW programmable	OFF	OFF
	LSAD	ON	OFF	OFF
	Ultra Low Power (ULP) Data Acquisition Subsystem	SW programmable	SW programmable	SW programmable
	SAR-ADC	SW programmable	SW programmable	SW programmable
	RF front-end	SW programmable	OFF	SW programmable
	SDAC	SW programmable	OFF	OFF
	ACOMP	SW programmable	SW programmable	SW programmable
RF	RF front-end	SW programmable	OFF	SW programmable

NOTE: The power modes in descending order of power consumption are Run, Idle, Standby, Smart Sense, and Sleep.

8.6.1 Keeping the Debugger Connected in Low Power Modes

If the Arm Cortex-M33-Debug power domain remains active, and a debug connection has been previously established to the device, the debug link will remain active across different power modes. This allows the device to be debugged while in Sleep or Standby Mode.

To support debugging in Sleep and Standby Modes, the following items are forced to remain enabled during these modes:

- VDDC and VDDM regulators
- Pads used by the debug port, by masking pad retention configuration for only the active debug port pads (in SW debug mode this includes only the SWCLK and SWDIO pads, in JTAG debug mode this includes JTCK, JTMS, JTDI, JTDO, and optionally JTRESET).

IMPORTANT: The device will consume more power when a debug connection is used than the device would consume if the debug connection is not used.

NOTE: To maintain the same behavior as when the debug port is disabled in Sleep or Standby Mode, VDDC will be reset if the ACS_VDDRET_CTRL_VDDCRET_ENABLE bit from the ACS_VDDRET_CTRL register is not set. As it is not possible to match the VDDM behavior between cases where debug is enabled and debug is not enabled in Sleep and Standby Modes, take extra care in applications to appropriately set the ACS_VDDRET_CTRL_VDDMRET_ENABLE bit from the ACS_VDDRET_CTRL register if the contents of any RAM instances are to be retained when the debug port is disabled.

8.6.2 Run Mode

In Run Mode, all the circuitry is powered on.

8.6.2.1 Idle Mode

Idle Mode is a subset of the Run Mode configurations, allowing for some power savings relative to Run Mode, with the fastest wakeup time of any power mode. In Idle Mode, the device is configured as though entering into Standby Mode, but rather than disabling the system and isolating memories, the Arm Cortex-M33 processor is held waiting for an interrupt or event instead of actually entering Standby Mode, to minimize the time needed to return to normal Run Mode execution.

8.6.3 Sleep Mode

When operating in Sleep Mode, RSL15 shows very low current consumption. Only the wakeup logic (through pad, RTC, or sensor interface) is kept powered. The regulators, RF block, etc., are disabled. The bandgap, digital core, and memories can be optionally powered, at low voltage.

If the digital core is not powered (i.e., the core retention regulator is not enabled), a reset is generated when entering Sleep Mode, and all registers in the digital core are reset to their default values. Registers in the Analog Control Sub-system (ACS) always keep their values over power mode cycles.

The ACS_BOOT_CFG_PADS_RETENTION_EN bit in the ACS_BOOT_CFG register needs to be set prior to entering Sleep Mode, in cases where VDDC and VDDM are not powered during this mode. This ensures that the pads keep their

configuration (direction, state, etc.) during sleep time. Upon wakeup, the boot PROM code is executed. The initial pad configuration (the one used before entering the Sleep Mode) needs to be restored by the software before resetting the PADS_RETENTION_EN bit, to avoid toggling the pads.

NOTE: Setting the ACS_BOOT_CFG_PADS_RETENTION_EN bit in the ACS_BOOT_CFG register disables all RSL15 input pads. Disabling an input pad causes its logic level state to go low internally. All digital interfaces with at least one input signal must be disabled before enabling pad retention in order to prevent a false interrupt from aborting Sleep Mode.

The RF block (the analog section of the RF block) can be disconnected from its supply through the ACS_VDDRF_CTRL register. The RF (the digital section of the RF block) and Baseband, CryptoCell, CM33-FPU, and CM33-Debug power domains can be powered down through the SYSCTRL_RF_POWER_CFG, SYSCTRL_CRYPTOCELL_PWR_CFG, SYSCTRL_FPU_PWR_CFG, and SYSCTRL_DBG_PWR_CFG registers, respectively.

The following wakeup sources are typically selected using the configuration registers of the ACS:

1. Wakeup through an external event on the GPIO[3:0] pads
2. Wakeup through an ACOMP event
3. Wakeup through the baseband timer
4. Wakeup through the RTC, clocked either by the internal 32 kHz RC oscillator or the 32 kHz XTAL oscillator. The sleep time can be programmed using the RTC configuration registers of the ACS (see Section 7.4.5 “Real Time Clock (RTC)” on page 392).
5. Wakeup through a Sensor interface threshold event or FIFO full event.

Entering Sleep Mode 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 and RAM) are isolated from the core (AND gates).
4. Memories are powered off. However, if the VDDM retention regulator is enabled, RAM instances enabled in the memory enable retention latches are put into retention mode.
5. The logic is disconnected from its supply unless the core retention regulator is enabled.
6. The baseband timer is disconnected from its supply unless the VDDT power switch is enabled; to do this, short VDDT to VDDC when VDDCRET is enabled, or to VDDACS when VDDCRET is disabled.
7. The RF block is disconnected from its supplies. Note that the RF block needs to be isolated manually if the VDDC retention regulator is enabled.
8. The charge pump, VDDC, VDDM, and VDDRF regulators are disabled.

NOTE: To keep the charge pump enabled during Sleep Mode, set the ACS_SLEEP_MODE_CFG_VDDCP_ENABLE bit of the ACS_SLEEP_MODE_CFG register.

9. The VCC regulator converter is disabled.

NOTE: To keep the bandgap enabled during Sleep Mode, set the ACS_SLEEP_MODE_CFG_BG_ENABLE bit of the ACS_SLEEP_MODE_CFG register.

10. The bandgap is disabled.

NOTE: To keep the bandgap enabled during Sleep Mode, set the ACS_SLEEP_MODE_CFG_BG_ENABLE bit of the ACS_SLEEP_MODE_CFG register.

At wakeup, the following sequence restarts the system:

1. Eight LSBs of the RTC counter are captured in the ACS_WAKEUP_STATE_RTC_VALUE bit field of the ACS_WAKEUP_STATE register, to record the wakeup time.
2. The bandgap is enabled.
3. The other regulators are set according to the configuration registers when VDDC and VDDM are ready.
4. Memories are powered back on when VDDM is ready.
5. The wakeup delay is applied.
6. Memory isolation is removed.
7. The system clock is enabled.
8. The digital reset is released, enabling boot PROM execution, unless the core retention regulator has been enabled.

8.6.3.1 Smart Sense Mode

Smart Sense Mode is a low power features that can work as a subset of the Sleep Mode configurations. Smart Sense takes advantage of the low power capability of Sleep Mode, while keeping some digital and analog peripherals, including the ultra low power (ULP) data acquisition subsystem, active with minimal processor intervention. Smart Sense allows an RSL15 device not only to remain responsive to external events, but also to monitor and acquire data from external sensors with very low system-level power consumption. Typical wakeup sources used with Smart Sense Mode include the analog comparitor (ACOMP) (see Section 12.1 “Analog Comparator” on page 645) and the FIFO (FIFO_FULL) and threshold (THRESHOLD) sources from the ultra low power data acquisition subsystem (see Section 12.8 “Ultra-Low Power Data Acquisition Subsystem” on page 660). If you are using Smart Sense Mode, we recommend configuring the ULP data acquisition subsystem to be clocked from STANDBYCLK for consistent data collection in both Run Mode and Smart Sense Mode.

8.6.4 Standby Mode

Standby Mode can be used to reduce the average power consumption for short intervals of inactivity. In this mode, the logic and memories are not clocked, and are powered at a reduced voltage to minimize the leakage current. The ACS, bandgap, and digital regulator are active in this mode.

Most of the blocks can be powered down individually using memory-mapped registers.

The RF block (the analog section of the RF block) can be disconnected from its supply through the ACS_VDDRF_CTRL register. The RF (the digital section of the RF block) and Baseband, CryptoCell, CM33-FPU, and CM33-Debug power domains can be powered down through the SYSCTRL_RF_POWER_CFG, SYSCTRL_CRYPTOCELL_PWR_CFG, SYSCTRL_FPU_PWR_CFG, and SYSCTRL_DBG_PWR_CFG registers, respectively.

The reduced voltage level can be programmed in the STANDBY_VTRIM field of the ACS_VDDC_CTRL and ACS_VDDM_CTRL registers. (See Section 8.3 “Power Supply Registers” on page 416.

Entering Standby Mode starts the following sequence:

1. The system clock is stopped.
2. All memories (flash, PROM and RAM) are isolated from the core (AND gates).
3. ROM and flash are powered off, and the relevant RAM instances are placed in retention mode.
4. The VDDC and VDDM regulators’ output voltages are set to their standby voltages.

At wakeup, the following sequence restarts the system:

1. Eight LSBs of the RTC counter are captured in the ACS_WAKEUP_STATE_RTC_VALUE field of the ACS_WAKEUP_STATE register, to record the wakeup time.
2. The VDDC and VDDM regulator output voltages are set to the normal voltage.
3. Memories are powered back on.

4. The wakeup delay is applied.
5. Memory isolation is removed.
6. Clock is enabled and system execution is resumed.

8.6.5 Entering a Low Power Mode

NOTE: Wakeup event flags must be cleared before entering any Low Power Mode. If the application does not do this, and if a wakeup event flag is still set, the system wakes up immediately. These flags are not reset by the core watchdog reset. (See Section 8.6 “Power Modes” on page 431.)

Entering any of the Low Power Modes is performed with these steps:

1. For Sleep Mode, if custom boot is selected in the ACS_BOOT_CFG_BOOT_SELECT field of the ACS_BOOT_CFG register, store the desired memory access configuration in the ACS_BOOT_GP_DATA register.
2. Enable the wakeup source if needed. For example, the GPIO[3:0] wakeup source can be enabled by setting the relevant ACS_WAKEUP_CFG_GPIO*_EN and ACS_WAKEUP_CFG_GPIO*_POL bits in the ACS_WAKEUP_CFG register. RTC alarm wakeup source can be enabled by configuring and enabling it through the ACS_RTC_CFG and ACS_RTC_CTRL registers (Section 7.4.5 “Real Time Clock (RTC)” on page 392). Using the baseband timer as a wakeup source requires configuring the baseband controller registers.
3. Set an appropriate value of wakeup delay in the ACS_WAKEUP_CFG_DELAY field of the ACS_WAKEUP_CFG register.
4. Clear all sticky wakeup event flags in the ACS_WAKEUP_CTRL register. If this is not done, and if a wakeup event flag is still set, the system wakes up immediately.
5. For Sleep Mode, if the logic content is to be kept, set the ACS_VDDRET_CTRL_VDDCRET_ENABLE bit in the ACS_VDDRET_CTRL register (see Section 8.3 “Power Supply Registers” on page 416).
6. For Sleep Mode, if RAM content is to be kept, set the ACS_VDDRET_CTRL_VDDMRET_ENABLE bit in the ACS_VDDRET_CTRL register, and configure the RAM instances to be kept in retention mode in the SYSCTRL_MEM_POWER_STARTUP and SYSCTRL_MEM_POWER_ENABLE registers (see Section 9.5 “Memory Registers” on page 504).
7. For Sleep Mode, put the pads in retention mode by setting the ACS_BOOT_CFG_PADS_RETENTION_EN bit in the ACS_BOOT_CFG register.
8. For Sleep Mode, if the baseband low power timer is kept powered, enable the ACS_VDDRET_CTRL_VDDTRET_EN bit in the ACS_VDDRET_CTRL register (see Section 8.3 “Power Supply Registers” on page 416).
9. For Sleep Mode, if the Arm CryptoCell-312 Always On is kept powered, enable the ACS_PWR_CTRL_CCAO_PWR_EN bit in the ACS_PWR_CTRL register.
10. For Sleep Mode, if the ULP data acquisition subsystem is kept powered, enable the ACS_PWR_CTRL_SENSOR_PWR_EN bit in the ACS_PWR_CTRL register.
11. Switch the SYSCLK to the RC oscillator by setting the ACS_RCOSC_CTRL_RC_FSEL field of the ACS_RCOSC_CTRL register to RC_OSC_3MHZ. If the RC oscillator is calibrated for a frequency different from 3 MHz, clear the ACS_RCOSC_CTRL_RC_FTRIM_FLAG field of the ACS_RCOSC_CTRL register, as it must be correct for the boot ROM.
12. For Standby Mode, including Smart Sense Mode, if the RFCLK needs to stay active (typically when the standby period is very short), leave the oscillator enabled and set the RFCLK clock divider in the RF block to Disabled (i.e., no clock output). If the RFCLK does not need to stay active (typically when the standby period is longer), disable it; it restarts after coming out of Standby Mode.
13. When entering Sleep Mode with the digital logic in retention (VDDCRET enabled), ensure that the RF is isolated (access bit disabled).
14. Write the relevant 32-bit power-down key in the ACS_PWR_MODES_CTRL register.
15. Put the CPU in power down mode with the WFI statement. This is required for entering all Low Power Modes.

IMPORTANT: The ACS waits for the Arm Cortex-M33 processor to enter into a WFI state before starting to enter into a low power mode. This has two potential impacts:

- **If an interrupt is pending when the processor reaches the WFI instruction, the processor does not enter a WFI state and does not enter a low power mode. Execution continues with the instruction following the WFI statement.**
- **If no interrupt is pending when the processor reaches the WFI instruction, the processor enters a WFI state. The ACS then takes 4.5 system clock cycles to enter into a Low Power Mode. Due to this delay, a race condition results if an interrupt occurs during the first three cycles of this process, while the digital logic and memories are being placed into retention. If an interrupt occurs during this process, the Arm Cortex-M33 processor resumes, and if this interrupt occurred in the first three cycles there is a danger that the processor might access a powered down memory prior to going to a Low Power Mode. If the interrupt occurs in the fourth cycle, the processor does not have sufficient time to execute any instructions before transitioning to the low power mode.**

  If this race condition is triggered it can cause either a bus or memory fault, or an access to a partially powered down memory, which would cause a corrupted memory access. To avoid this potential fault condition, follow the WFI instruction with at least three cycles of dummy operations that can be safely accessed while memories remain in retention, and which do not count on any data read for proper operation (as shown in the supplied firmware and sample code).

IMPORTANT: In the firmware sample code, the values defined by `TWOSC` (in the *app.h* file) is used to configure the time required for analog wake-up, 48 MHz XTAL oscillator stabilization, and execution of software wakeup functions. Those values (in microseconds) are based on use of the RSL15 EVBs, SDK sample code, and room temperature. You might need to tune the `TWOSC` value based on your system capacitance, characteristics of the 48 MHz XTAL oscillator, wakeup function execution time, sleep duration, and operating temperatures.

8.6.6 Waking Up from a Low Power Mode

If an event occurs on any selected wakeup source while the system is entering a Low Power Mode, the system wakes up immediately.

After wakeup, by reading the `ACS_WAKEUP_CTRL` register, the Arm Cortex-M33 processor knows which event source has wakened the device up. (See Section 8.6.7 “Wakeup Sources” on page 440 for a list of wakeup sources.) In case another event occurs during the wakeup sequence, multiple wakeup source bits can be set when the `ACS_WAKEUP_CTRL` register is read. The read-only `ACS_WAKEUP_STATE_WAKEUP_SRC` field from the `ACS_WAKEUP_STATE` register indicates the first wakeup source that actually wakes up the system.

The `ACS_BOOT_CFG` register contains the `ACS_BOOT_CFG_BOOT_SELECT` field which, when configured to `BOOT_FLASH_XTAL_DEFAULT_TRIM` or `BOOT_FLASH_XTAL_CUSTOM_TRIM` by the software, tells the system (handled by the system booting software) to enable the RF block and start the 48 MHz XTAL oscillator directly after boot. This can reduce the boot time considerably, as the 48 MHz XTAL oscillator can have a long start-up time.

The `ACS_BOOT_CFG_BOOT_SELECT` field of the `ACS_BOOT_CFG` register, when configured to `BOOT_CUSTOM` by the software, tells the system (handled by the system booting software) to execute the code from the flash or RAM

using information stored in a RAM instance that has been kept in retention. The VDDM retention regulator must be enabled to keep content of the RAM instance valid on wakeup. For details on how the wakeup from retention memory works, see Section 9.1.1 “Memory Instances” on page 466.

The remaining ACS_BOOT_CFG_BOOT_SELECT configuration is BOOT_FLASH_XTAL_DISABLE, which does not configure the 48 MHz XTAL oscillator, and boots the system like the first boot.

The ACS_BOOT_CFG register contains the ACS_BOOT_CFG_BOOT_ROT_BYPASS bit. If this bit is set, the boot ROM skips the Root of Trust procedure when coming out of a Low Power Mode. This bit then needs to be reset by writing 0 to it.

Finally, the ACS_BOOT_CFG register contains the ACS_BOOT_CFG_BOOT_PWR_CAL_BYPASS bit. If this bit is set, the boot ROM skips the calibration of the analog part when coming out of a Low Power Mode. The bit then needs to be reset by writing 0 to it.

The ACS_WAKEUP_CFG_DELAY field in the ACS_WAKEUP_CFG register defines a delay from the point when VDDC and VDDM are ready to the point when the digital reset is released; this delay ensures that VDDC and VDDM have sufficient time to settle into a safe operating voltage before full wakeup.

IMPORTANT: For proper operation, the ACS_WAKEUP_CFG_DELAY field **in the** ACS_WAKEUP_CFG **register must be set to a minimum delay of** WAKEUP_DELAY_2**.**

8.6.7 Wakeup Sources

The following are the wakeup sources that can wake RSL15 from a Low Power Mode:

- GPIO[3:0] pad
- Baseband timer
- RTC timer alarm, RTC clock tick, and RTC overflow errors
- DC-DC overload
- Analog comparator
- ULP data acquisition subsystem FIFO full
- ULP data acquisition subsystem threshold

8.6.8 Adding a Low-Power Mode to an Existing Application

The Hardware Abstraction Layer provides a library for the Power Modes and a library for the Low-Power Clocks. These libraries can be used to conveniently add a Low Power Mode to an existing application. The following steps describe how this can be done with the *ble_peripheral_server* sample application. Along the way, some details are provided to give a deeper understanding of how the Power Modes work, and what needs to be taken into account with them for different types of applications (e.g., applications that use Bluetooth vs those that do not use Bluetooth).

NOTE: All steps after Section 8.6.8.1 “Setting Up the Application” on page 440 can be performed in any order you chose. This guide presents the steps in a top-down approach, to provide a better understanding of how the changes are integrated into the existing application and what is required from the functions that are added, to ensure that the Low Power Mode functions correctly.

8.6.8.1 Setting Up the Application

Before making changes to the application code, we need to set up our project with the libraries and files that we need:

1. While in the **CMSIS-Pack Manager** perspective of the onsemi IDE, copy the *ble_peripheral_server* sample application into your workspace, as shown in the "Power Modes" figure (8.6).

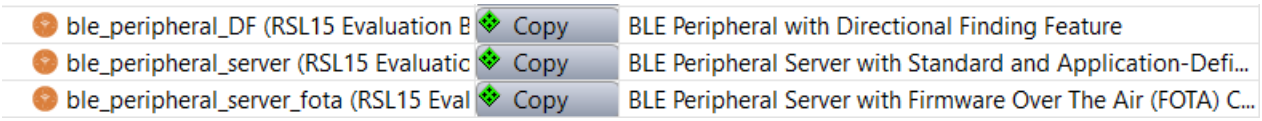


Figure 40. Copying the Project into the Workspace

2. Open the *ble_peripheral_server.rteconfig* file using the RTE Configuration Wizard and verify that the HAL libraries are selected, as shown in the "Adding HAL Libraries and New Files" figure (Figure 41). This ensures that you can use the libraries for the Power Modes and the Low-Power Clock. Save the file if any changes have been made.

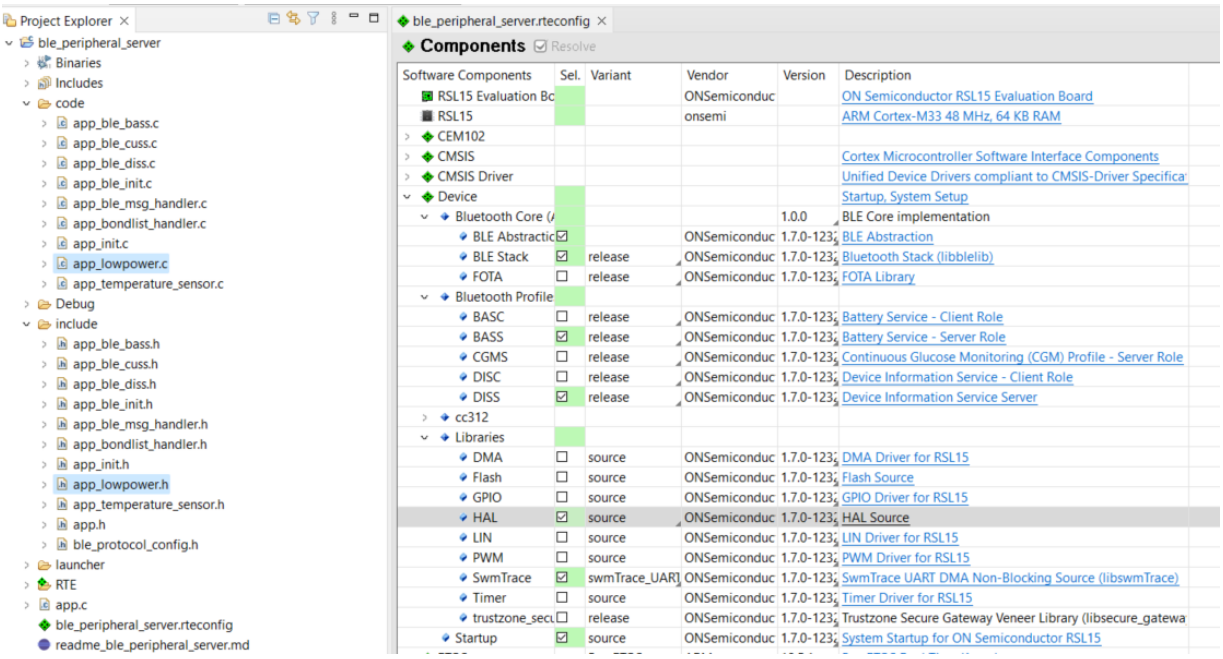


Figure 41. Adding HAL Libraries and New Files

3. Create a new source and header file called *app_lowpower.c* and *app_lowpower.h*, respectively. These new files will hold all the application-level functions and settings that we'll need to add for the low-power state to work.

8.6.8.2 Updating the Application's Control Flow

The additions made for the Low Power Mode are mostly contained in the new source file we created. However, integration with the existing application needs to be done in the main application file, *app.c*. The integration involves initializing the required components and updating the flow of the main spin loop to attempt to enter a low-power state whenever possible. Open *app.c* and make the following changes:

1. Include the new header file we just created. This will allow you to access the functions that iscreated later.

```
#include "app_lowpower.h"
```

2. In the main function, add a function call to `App_LowPower_Init()`. This function call needs to be placed between the function call for the GPIO initialization, `App_GPIOInit()`, and re-enabling the interrupts and exceptions with the call to `App_EnableInterrupts()`.

```
int main(void)
{

    ...

    /* Configure the application's power mode and wakeup sources */
    App_LowPower_Init(
        &App_MainLoop,
        &App_LowPower_SavePeripheralStates,
        &App_LowPower_RestorePeripheralStates
    );

    ...

}
```

3. In the function call to `App_LowPower_Init()`, add the three function pointers `&App_MainLoop`, `&App_LowPower_SavePeripheralStates`, and `&App_LowPower_RestorePeripheralStates`. The first pointer is for the main spin loop and the other two are defined by us later.
4. In the main spin loop, `App_MainLoop()`, add a conditional to the Bluetooth kernel processing to make sure that the Bluetooth baseband is awake and ready to perform the processing. Add a function call to `App_LowPower_SleepCheck()` (to be defined later); this function attempts to place the system into a low-power state once the kernel processing is complete. Lastly, if the baseband is not awake we can place the system into Idle Mode.

```
static void App_MainLoop(void)
{
    while (1)
    {
        /* Refresh the watchdog timer */
        SYS_WATCHDOG_REFRESH();

        if (BLE_Baseband_Is_Awake())
        {
            /* Perform pending Bluetooth processes */
            BLE_Kernel_Process();

            /* Check if the system is ready to go to sleep, do so if possible */
            App_LowPower_SleepCheck();
        }
        else
        {
            /* Go into Idle Mode and wait for an interrupt */
            __WFI();
```

```
            }
        }
    }
```

NOTE: When adding a Low Power Mode to an application that does not feature Bluetooth, you can skip the check to see if the baseband is awake and the call to perform pending Bluetooth processes. The application can proceed directly with the `App_LowPower_SleepCheck()` function call.

8.6.8.3 Configuring the Low-Power Module

For the Low Power Mode to work with the application, we first need to configure these things:

- The low-power clock
- The wakeup sources
- The general low-power configuration structure

In this example, we keep everything as basic as possible and use default configurations as much as we can.

Add the following to the *app_lowpower.c* module and its header file:

1. Include the libraries and files needed, as follows:The module's header, *app_lowpower.h*, stores the function prototypes. The *app_init.h* header is required for the definition of the clock frequencies that are used by the application. The *lowpower_clock.h* library allows us to initialize the low-power clock, and the *power_modes.h* will give us access to the default low-power configuration structure that we'll modify for our application. It also gives us access to the `Sys_PowerModes_SetWakeupConfig()` function needed to set the wakeup sources that we want to use.

```
/* Device and library headers */
#include <ble_abstraction.h>
#include <lowpower_clock.h>
#include <power_modes.h>

/* Application headers */
#include "app_init,h"
#include "app_lowpower.h"
```

NOTE: For applications that do not use Bluetooth, the Bluetooth stack and libraries are not needed nor selected. The *ble_abstraction.h* header can be omitted for those applications.

2. Define the initialization function that we added to *app.c* and add a prototype for the function to the *app_lowpower.h* header file.

```
void App_LowPower_Init(AppResumeAddress_t p_app_addr, AppPeripheralFunc_t p_save_
peripherals, AppPeripheralFunc_t p_restore_peripherals)
{
    ...
}
```

3. In the new function, add a call to `Sys_LPClock_Init()`. This initializes the XTAL32K low-power clock that is used by the system in a Low Power Mode. If desired, you could change the clock configuration and source, but we recommend that you use the default settings when getting started with the EVB.

```
void App_LowPower_Init(AppResumeAddress_t p_app_addr, AppPeripheralFunc_t p_save_
peripherals, AppPeripheralFunc_t p_restore_peripherals)
{
    /* Configure and initialize the system low-power clock to default values */
    Sys_LPClock_Init();

    ...
```

4. Next we need to configure the wakeup sources we want to use. Since this application is Bluetooth enabled, any processes that use the kernel timer are automatically considered a wakeup source and do not require further configuration. For our example, we add GPIO0 as a wakeup source, which means it is not used for any other purpose. Reset GPIO0 to the default settings and disable its interrupt. The wakeup events triggered by GPIO0 are also captured while the system is in Run Mode, so you do not need to use the interrupt for the pin.

```
   /* Disable and reset the GPIO back to its default settings */
    SYS_GPIO_CONFIG(GPIO0, (GPIO_MODE_DISABLE | GPIO_LPF_DISABLE | GPIO_WEAK_PULL_
UP | GPIO_6X_DRIVE));

    /* Disable the NVIC interrupt for GPIO0. Since GPIO0 is used as a wakeup
 source, it will be captured
     * by the WAKEUP_IRQn interrupt which works during the low-power mode and the
regular Run Mode. */
    NVIC_DisableIRQ(GPIO0_IRQn);

   ...
```

NOTE: If the application enables the GPIO interrupts somewhere else in the code (e.g. during the initialization of the rest of the system), make sure to remove those statements so that the interrupt is not re-enabled after the low-power initialization is complete.

5. Now we customize the low-power configuration structure to suit our application. We are using Sleep Mode with Memory Retention and have GPIO0 as our only wakeup source (besides the baseband). The configuration structure is initialized by the Power Modes library with commonly used settings for this type of mode, so our modifications will be minimal. See the Power Modes library for more details about the default configuration and the different settings.
    a. Set the three function pointers that are passed as parameters into the function. The first pointer, to `App_MainLoop`, is set to `app_lowpower_mode_cfg.p_app_resume`. When the system wakes from a low-power state, it first restores crucial components that are required for Run Mode, including initializing the peripherals that we are using (see the next step in this walk-through). With crucial components running, it then services interrupts that have been triggered, including the `WAKEUP_IRQn` that is triggered by wakeup sources during a low-power mode. Once the interrupts have been serviced, the system resumes running the application from the address we have provided to `app_lowpower_mode_cfg.p_app_resume`. In this case, we want it to continue running through the application's main spin loop.

```
    /* Application will resume from this address after wakeup from BOOT_
CUSTOM
  with memory retention */
     app_lowpower_mode_cfg.p_app_resume = p_app_addr;

     ...
```

b. The second and third pointers, to the `App_LowPower_SavePeripheralStates` and `App_LowPower_RestorePeripheralState` functions (not yet defined), are set to `app_lowpower_mode_cfg.p_save_peripherals` and `app_lowpower_mode_cfg.p_restore_peripherals`, respectively. These functions are called by the Power Modes library before it places the system into a low-power state and after waking up from a low-power state. The functions are intended to save and restore the states of any peripherals that are used by the application and that are reset when the system enters the low-power state. For example, if the system enters Sleep Mode with Memory Retention, the GPIO configuration is reset, so we need to save the values of the GPIO configuration registers to a set of variables and then restore those values in our save and restore functions. If we are using Standby Mode, the GPIO registers are not reset during the low-power state, so we do not need to save and restore the register values.

```
    /* Set the peripheral configuration functions that will be called before
  sleep and after wakeup */
     app_lowpower_mode_cfg.p_save_peripherals = p_save_peripherals;
     app_lowpower_mode_cfg.p_restore_peripherals = p_restore_peripherals;

     ...
```

c. Set the boot configuration to `BOOT_CUSTOM`, which indicates to the Power Modes library that we want our application to resume running from the function we provided to `app_lowpower_mode_cfg.p_app_resume` (i.e. `App_MainLoop`).

```
    /* Set the boot configuration */
     app_lowpower_mode_cfg.boot_cfg = BOOT_CUSTOM;

     ...
```

d. Set the clock configurations that are used to reconfigure the system clocks after waking up. These clock settings must match what our application was using before, so we are setting them using the values defined in *app_init.h*.

```
    /* Set the clock configurations for use during Run Mode */
     app_lowpower_mode_cfg.clock_cfg.sensorclk_freq = SENSOR_CLK_HZ;
     app_lowpower_mode_cfg.clock_cfg.systemclk_freq = SYSTEM_CLK_HZ;
     app_lowpower_mode_cfg.clock_cfg.uartclk_freq = UART_CLK_HZ;
     app_lowpower_mode_cfg.clock_cfg.userclk_freq = USER_CLK_HZ;

     ...
```

NOTE: For applications that do not use Bluetooth, we also need to set the flag indicating that Bluetooth is not being used, i.e., `app_lowpower_mode_cfg.ble_present_flag = POWER_MODES_BLE_NOT_PRESENT;`

NOTE: If the baseband is not utilized in the application (e.g., Bluetooth is not used and the kernel timer is not used), VDDT retention can be disabled to reduce power consumption by setting `app_lowpower_mode_cfg.vddret_ctrl.vddt_ret = VDDTRETENTION_DISABLE;`

6. Initialize the wakeup sources that have been set (in our case the default sources) by calling `Sys_PowerModes_SetWakeupConfig()`. This function will need to be called whenever our wakeup sources change in the application. In our example, the wakeup sources remain static.

```
    /* Initialize the wakeup configuration */
     Sys_PowerModes_SetWakeupConfig(app_lowpower_mode_cfg.wakeup_cfg);

     ...
```

7. Clear the `WAKEUP_IRQn` interrupt, in case the interrupt is pending, and enable the interrupt.

```
    /* Clear any pending WAKEUP interrupt and enable the interrupt */
     NVIC_ClearPendingIRQ(WAKEUP_IRQn);
     NVIC_EnableIRQ(WAKEUP_IRQn);
}
```

8.6.8.4 Attempting to Enter a Low-Power State

Before the system can enter a low-power state, checks must be peformed to make sure that the system is ready for the transition.

1. Check if there is a swmTrace transmission in progress, and verify that the low-power clock is ready for use.

   The swmTrace check ensures that log messages are not abruptly cut off when the system goes to sleep. If we are not concerned about the log messages and just want the system to enter a low-power state as quickly as possible, then we can remove the check for the swmTrace (and possibly disable the swmTrace altogether).

   The low-power clock check is crucial if we are using the RC32K as our low-power clock source. In such a case, we need to periodically measure the clock frequency, and the system must remain in Run Mode until this measurement is complete and the low-power clock is ready again. If the XTAL32K is used as the low-power clock source (the default setting, which we are using in this example), then the clock remains in the ready state after it is first initialized, and the check is a bit superfluous (though still recommended).

```
void App_LowPower_SleepCheck(void)
{
    /* If not in the middle of a swmTrace UART transfer, allow the application /*
    /* to enter into a low-power mode */
    if (!swmTrace_txInProgress() && LPClock_state == LPCLOCK_READY)
    {
        ...
    }
    else
    {
        __WFI();
    }
```

```
}
```

2. If the checks for swmTrace and the low-power clock fail, we can still place the system into Idle Mode, which conserves some power but keeps the system in a ready state.

3. If the checks for swmTrace and the low-power clock pass, we can check if the baseband is ready to be placed into a low-power state. To do so we first initialize a structure with our request for entering a low-power state, the minimum sleep duration, and the maximum sleep duration for the low-power state. The minimum sleep duration is measured in microseconds and the value must exceed the time required for the system to wake up from the low-power state. The maximum sleep duration is measured in multiples of 312.5 microseconds.

```
/* If not in the middle of a swmTrace UART transfer, allow the application /*
/* to enter into a low-power mode */
if (!swmTrace_txInProgress() && LPClock_state == LPCLOCK_READY)
{
    /* Define the parameters used for putting the device to sleep */
    struct ble_sleep_api_param_tag ble_sleep_api_param =
    {
        .app_sleep_request = 1,        /* request to go to sleep */
        .max_sleep_duration = 96000,   /* 30s = (96000 * 312.5us) */
        .min_sleep_duration = 3500,    /* 3500us */
    }

    ...


}
```

NOTE: For non-Bluetooth applications, the baseband check does not need to be performed, and as such this structure does not need to be defined.

4. Disable interrupts globally. This needs to be done before we can check the baseband. After the baseband check, enable the interrupts again.

```
/* Checks for sleep have to be done with interrupts disabled */
 GLOBAL_INT_DISABLE();

 ...

/* Checks for sleep have to be done with interrupts disabled */
 GLOBABL_INT_RESTORE();
```

5. Check if the baseband can be placed into a low-power state. Do this by calling the `BLE_Baseband_Sleep()` function and passing the structure that is defined with the minimum and maximum sleep durations. The function checks when the next baseband process is scheduled to occur (this includes any processes that are set to use the kernel timer and all Bluetooth processes). The function then cross-references the baseband process timing with the desired sleep interval. If the next process occurs before the minimum sleep duration, the baseband avoids entering the low-power state. If the next process occurs after the maximum sleep duration, the baseband sets the sleep time to the maximum duration that has been set. The function calculates the number of low-power clock cycles that it can go to sleep for, and assigns this number to `ble_sleep_api_param.calculated_sleep_`

duration.

```
        /* Check if processor clock can be gated */
         switch (BLE_Baseband_Sleep(&ble_sleep_api_param))
         {
             ...
         }
```

NOTE: For non-Bluetooth applications, this step and the next step can be skipped.

6. The `BLE_Baseband_Sleep()` function returns one of three states that indicates if the system can enter a low-power state.

   If `RWIP_ACTIVE` is returned, it means that there is an active baseband process and the system cannot enter a low-power state.

   If `RWIP_CPU_SLEEP` is returned, it indicates that the system can be placed into Idle Mode but must not be placed into a lower-power state. This is likely to occur if the next baseband process is scheduled to take place within the minimum sleep duration that we set, but there is no baseband process currently active.

   If `RWIP_DEEP_SLEEP` is returned, it mean that there are no active or pending baseband processes and the baseband is ready to enter a low-power state.

```
            case RWIP_DEEP_SLEEP:
             {
                 /* The baseband is ready to be placed into a low-power mode. */

                 ...

                 break;
             }
             case RWIP_CPU_SLEEP:
             {
                 /* The baseband cannot be placed into a low-power mode. Instead,
let the CPU wait in Idle Mode
                  * until the next task is read (i.e. wait for interrupt).
                 __WFI();
                 break;
             }
             case RWIP_ACTIVE:
             {
                 /* Some activity is being process or pending; the system /*
                 /* cannot enter a low-power mode. */
                 break;
             }
```

7. Since our application is Bluetooth enabled and entering Sleep Mode, we need to save some RF registers before we enter the low-power state. This is not required when using Standby Mode.

```
                /* Save the RF registers before entering low-power mode. These */
```

```
                /* states need to be restored after wakeup */
                 App_LowPower_RFSaveStates();

                 ...
```

NOTE: For non-Bluetooth applications, the RF is not used so this step can be skipped.

8. Call the Power Modes function and pass our modified configuration structure to enter the low-power state. The Power Modes library takes care of the remaining processing required to place the system into the chosen low-power state.

```
                /* Enter the low-power mode, as configured */
                 Sys_PowerModes_EnterPowerMode(&app_lowpower_mode_cfg);
```

NOTE: For non-Bluetooth applications, if the GPIO is the only wakeup source, the RTC clock and the system clock are not present, which can cause a reset generated by the clock detector. To avoid this, the clock detector can be disabled by setting the `ACS_CLK_DET_CTRL_RESET_IGNORE` bit from the `ACS_CLK_DET_CTRL` register, and resetting the `ACS_CLK_DET_CTRL_ENABLE` bit from the same regiersr, before entering the low-power mode.

8.6.8.5 Handling Wakeup Sources

Besides the baseband processes (e.g. the Bluetooth processes and the kernel timer processes), the application can also be configured to wake up earlier based on several available wakeup sources. The most commonly used wakeup sources include receiving an input signal on a GPIO, the FIFO becoming full, an RTC timer event, and a sensor input threshold event. For our application, we want to be able to handle GPIO wakeup events, and we need to define the `WAKEUP_IRQHandler()` to do so. Since the wakeup interrupt handler is called first, before the system returns to the `App_MainLoop()`, we need to do some basic re-initialization to ensure the system functions as we wish. This interrupt handler is also called whenever a baseband event occurs, so we can use this to re-initialize anything needed for baseband processes as well.

1. Refresh the system watchdog to avoid a timeout, re-initialize the swmTrace to re-enable logging messages, and restore the RF registers saved before entering the low-power state.

```
void WAKEUP_IRQHandler(void)
{
    SYS_WATCHDOG_REFRESH();

    /* Re-initialize the swmTrace on wakeup */
    App_SWMTraceInit();

   /* Restore the VDDPA settings to keep the TX power consistent */
    App_LowPower_RFRestoreStates();

    ...
```

NOTE: For non-Bluetooth applications, the RF is not used, so the registers do not need to be restored.

2. Check if the wakeup event has occurred from one of our configured sources. Since the same interrupt occurs for all of the wakeup sources, we need to check each one individually and perform the corresponding actions. It is also possible for multiple wakeup events to be set (e.g. a GPIO input is provided just as the FIFO becomes full), so make sure to check each in a separate `if` block.

```
    ...

     /* Check for a GPIO0 wakeup event */
     if (ACS->WAKEUP_CTRL & WAKEUP_GPIO0_EVENT_SET)
     {
         /* Clear the flag for the GPIO wakeup event */
         WAKEUP_GPIO0_FLAG_CLEAR();


         /* Perform the associated processing */
         Sys_GPIO_Toggle(USER_LED_GPIO);
     }

     ...
```

3. If the wakeup event has occurred for our wakeup source, we need to clear the wakeup flag and process the event. In our case, the GPIO wakeup event toggles the EVB's green LED on GPIO8.
4. At the end of the interrupt handler we can place a safety check that resets the interrupt flag, so that the interrupt handler is called again if there is a pending wakeup event. This is a good check to add because it ensures that no events are missed, especially if they are set during the processing of another event.

```
    ...

     /* Check if any wakeup events are still pending and reset the IRQ handler */
    if (ACS->WAKEUP_CTRL && (!NVIC_GetPendingIRQ(WAKEUP_IRQn)))
     {
         NVIC_SetPendingIRQ(WAKEUP_IRQn);
     }
}
```

NOTE: If you forget to reset the wakeup flags for the processed events or forget to processes events altogether, then the interrupt is continuously reset by the statement above, and the system becomes stuck, unable to continue other tasks or re-enter the low-power state.

8.6.8.6 Saving and Restoring RF Register States

The function calls to save and to restore the RF registers before entering a low-power state and after waking up, respectively, have already been added in the previous steps. Now we need to define those functions and the variables to hold the values while the system is in the low-power state.

NOTE: For non-Bluetooth applications, the RF is not used, so these steps can be skipped.

1. Define a set of global variables to store the register values.

```
/* Storage variables for the RF registers */
```

```
static uint32_t vddpa_ctrl = 0;
static uint32_t vddrf_ctrl = 0;
static uint8_t vddpa_power = 0;
static SYSCTRL_VDDPA_CFG0_Type vddpa_cfg0 = {0};
```

2. Define the RF save function.

```
/* Save the RF registers */
static void App_LowPower_RFSaveStates(void)
{
    vddrf_ctrl = ACS->VDDRF_CTRL;
    vddpa_ctrl = ACS->VDDPA_CTRL;
    vddpa_power = RF0_REG1A->PA_PWR_PA_PWR_BYTE;
    vddpa_cfg0 = (*(SYSCTRL_VDDPA_CFG0));
}
```

3. Define the RF restore function.

```
/* Restore the RF settings */
static void App_LowPower_RFRestoreStates(void)
{
    if (vddrf_ctrl != 0)
    {
        /* Restore VDDRF supply without changing trimming settings */
        Sys_ACS_WriteRegister(&ACS->VDDRF_CTRL, vddrf_ctrl);

        /* Wait until VDDRF supply has powered up */
        while (!(ACS->VDDRF_CTRL & VDDRF_READY))
        {
            /* Waiting */
        }

        /* Restore VDDPA */
        Sys_ACS_WriteRegister(&ACS->VDDPA_CTRL, vddpa_ctrl);

        RF0_REG1A->PA_PWR_PA_PWR_BYTE = vddpa_power;
        (*(SYSCTRL_VDDPA_CFG0)) = vddpa_cfg0;
    }
}
```

8.6.8.7 Saving and Restoring Peripheral States

In the previous steps we have made several mentions of saving and restoring peripheral states. Similar to the save and restoring for the RF registers, we need to define functions that save and restore the peripheral registers that are used by our application. However, unlike the RF registers, the peripheral state functions are handled by the Power Modes library, which is why we provide pointers to these functions in the configuration structure.

These save and restore functions need to be customized heavily to match the application that they are created for and the low-power state that the system is trying to enter. For example, if the application uses LSAD and enters Sleep Mode with Memory Retention, then we need to save and restore the LSAD registers. However, if the application uses LSAD and enters Sleep Mode with Core Retention, then the LSAD registers are retained and we do not need to save and restore the registers. In our example, the application uses GPIO and Sleep Mode with Memory Retention, so we need to save and restore the GPIO registers.

1. Define a set of global variables to store the GPIO register values, and define a save function that assigns the register values to those variables.

```
/* Storage variables for the GPIO registers */
static uint32_t gpio_cfg[GPIO_PAD_COUNT] = {0};
static uint32_t gpio_output = 0;
static uint32_t gpio_jtag_sw_pad_cfg = 0;

/* Save the GPIO registers */
void App_LowPower_SavePeripherals(void)
{
    for (uint8_t i = 0; i < GPIO_PAD_COUNT; i++)
    {
        gpio_cfg[i] = GPIO->CFG[i];
    }
    gpio_output = GPIO->OUTPUT_DATA;
    gpio_jtag_sw_pad_cfg = GPIO->JTAG_SW_PAD_CFG;
}
```

2. Define a restore function that reads the global variable values and re-assigns them back to the GPIO registers.

```
/* Restore the GPIO registers */
void App_LowPower_RestorePeripherals(void)
{
    for (uint8_t i = 0; i < GPIO_PAD_COUNT; i++)
    {
        GPIO->CFG[i] = gpio_cfg[i];
    }
    GPIO->OUTPUT_DATA = gpio_output;
    GPIO->JTAG_SW_PAD_CFG = gpio_jtag_sw_pad_cfg;
}
```

The modifications to the *ble_peripheral_server* sample application are now complete, and the application is set to enter a low-power state. If we run the application on an EVB, we can measure the current consumption and confirm that it is entering the low-power state, and we can confirm that GPIO0 causes a wakeup event and toggles the green LED whenever pressed.

8.7 Power Modes Registers

Register Name	Register Description	Address
SYSCTRL_WAKEUP_ADDR	Wake-up Restore Address in Unpacked 32-bit Format	0x4000001C
ACS_PWR_MODES_CTRL	Power Modes Control Register	0x40001B5C
ACS_SLEEP_MODE_CFG	Sleep power mode configuration	0x40001B60
ACS_WAKEUP_CTRL	Wake-Up control / Status Register	0x40001B64
ACS_WAKEUP_CFG	Wake-up Configuration Register	0x40001B68
ACS_WAKEUP_STATE	RTC Timer wake-up value and wakeup source	0x40001B6C

Register Name	Register Description	Address
ACS_BOOT_CFG	Boot configuration Register	0x40001B70
ACS_BOOT_GP_DATA	Boot control Register Register	0x40001B74
ACS_PWR_CTRL	ACS Power Control Register	0x40001B8C

8.7.0.1 SYSCTRL_WAKEUP_ADDR

Bit Field	Read/Write	Field Name	Description
31:0	RW	WAKEUP_ADDR	Wake-up restore address in unpacked 32-bit format.

8.7.0.2 ACS_PWR_MODES_CTRL

Bit Field	Read/Write	Field Name	Description
31:0	W	POWER_MODE	32-bit key to enter STANDBY or SLEEP mode. This register must be written using a 32-bit access.

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:0	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

8.7.0.3 ACS_SLEEP_MODE_CFG

Bit Field	Read/Write	Field Name	Description
2	RW	BG_ENABLE	Keep Band-gap enabled during sleep
1	RW	VCC_ENABLE	Keep VCC low power enabled during sleep
0	RW	VDDCP_ENABLE	Keep VDDCP charge pump enabled during sleep

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
2	BG_ENABLE	BG_DISABLE_IN_SLEEP	Band-gap disabled in sleep	0x0*
		BG_ENABLE_IN_SLEEP	Band-gap enabled in sleep	0x1
1	VCC_ENABLE	VCC_DISABLE_IN_SLEEP	VCC disabled in sleep	0x0*
		VCC_ENABLE_IN_SLEEP	VCC enabled in sleep	0x1
0	VDDCP_ENABLE	VDDCP_DISABLE_IN_SLEEP	Charge pump disabled in sleep	0x0*
		VDDCP_ENABLE_IN_SLEEP	Charge pump enabled in sleep	0x1

8.7.0.4 ACS_WAKEUP_CTRL

Bit Field	Read/Write	Field Name	Description
27	R	RTC_OVERFLOW_WAKEUP	Indicate if RTC overflow has triggered a wake-up event
26	R	RTC_CLOCK_WAKEUP	Indicate if RTC clock has triggered a wake-up event
25	R	RTC_ALARM_WAKEUP	Indicate if RTC alarm has triggered a wake-up event
24	R	THRESHOLD_WAKEUP	Indicate if the sensor interface threshold has triggered a wake-up event
23	R	FIFO_FULL_WAKEUP	Indicate if the sensor interface FIFO has triggered a wake-up event
22	R	ACOMP_WAKEUP	Indicate if ACOMP has triggered a wake-up event
21	R	DCDC_OVERLOAD_WAKEUP	Indicate if DCDC overload has triggered a wake-up event
20	R	BB_TIMER_WAKEUP	Indicate if baseband timer has triggered a wake-up event
19	R	GPIO3_WAKEUP	Indicate if GPIO3 has triggered a wake-up event
18	R	GPIO2_WAKEUP	Indicate if GPIO2 has triggered a wake-up event
17	R	GPIO1_WAKEUP	Indicate if GPIO1 has triggered a wake-up event
16	R	GPIO0_WAKEUP	Indicate if GPIO0 has triggered a wake-up event
11	W	THRESHOLD_WAKEUP_CLEAR	Reset the sticky WAKEUP_THRESHOLD_EVENT flag
10	W	FIFO_FULL_WAKEUP_CLEAR	Reset the sticky WAKEUP_FIFO_FULL_EVENT flag
9	W	ACOMP_WAKEUP_CLEAR	Reset the sticky WAKEUP_ACOMP_EVENT flag
8	W	DCDC_OVERLOAD_WAKEUP_CLEAR	Reset the sticky WAKEUP_DCDC_OVERLOAD flag
7	W	RTC_OVERFLOW_WAKEUP_CLEAR	Reset the sticky WAKEUP_RTC_OVERFLOW_EVENT flag
6	W	RTC_CLOCK_WAKEUP_CLEAR	Reset the sticky WAKEUP_RTC_CLOCK_EVENT flag
5	W	RTC_ALARM_WAKEUP_CLEAR	Reset the sticky WAKEUP_RTC_ALARM_EVENT flag
4	W	BB_TIMER_WAKEUP_CLEAR	Reset the sticky WAKEUP_BB_TIMER_EVENT flag
3	W	GPIO3_WAKEUP_CLEAR	Reset the sticky WAKEUP_GPIO3_EVENT flag
2	W	GPIO2_WAKEUP_CLEAR	Reset the sticky WAKEUP_GPIO2_EVENT flag
1	W	GPIO1_WAKEUP_CLEAR	Reset the sticky WAKEUP_GPIO1_EVENT flag
0	W	GPIO0_WAKEUP_CLEAR	Reset the sticky WAKEUP_GPIO0_EVENT flag

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
27	RTC_OVERFLOW_WAKEUP	WAKEUP_RTC_OVERFLOW_EVENT_NOT_SET	RTC overflow has not triggered a wake-up event	0x0*
		WAKEUP_RTC_OVERFLOW_EVENT_SET	RTC overflow has triggered a wake-up event at least once	0x1
26	RTC_CLOCK_WAKEUP	WAKEUP_RTC_CLOCK_EVENT_NOT_SET	RTC clock has not triggered a wake-up event	0x0*
		WAKEUP_RTC_CLOCK_EVENT_SET	RTC clock has triggered a wake-up event at least once	0x1
25	RTC_ALARM_WAKEUP	WAKEUP_RTC_ALARM_EVENT_NOT_SET	RTC alarm has not triggered a wake-up event	0x0*
		WAKEUP_RTC_ALARM_EVENT_SET	RTC alarm has triggered a wake-up event at least once	0x1
24	THRESHOLD_WAKEUP	WAKEUP_THRESHOLD_EVENT_NOT_SET	The ULP data acquisition subsystem threshold has not triggered a wake-up event	0x0*
		WAKEUP_THRESHOLD_EVENT_SET	The ULP data acquisition subsystem threshold has triggered a wake-up event at least once	0x1
23	FIFO_FULL_WAKEUP	WAKEUP_FIFO_FULL_EVENT_NOT_SET	The ULP data acquisition subsystem FIFO has not triggered a wake-up event	0x0*
		WAKEUP_FIFO_FULL_EVENT_SET	The ULP data acquisition subsystem FIFO has triggered a wake-up event at least once	0x1
22	ACOMP_WAKEUP	WAKEUP_ACOMP_EVENT_NOT_SET	The ACOMP has not triggered a wake-up event	0x0*
		WAKEUP_ACOMP_EVENT_SET	The ACOMP has triggered a wake-up event at least once	0x1
21	DCDC_OVERLOAD_WAKEUP	WAKEUP_DCDC_OVERLOAD_EVENT_NOT_SET	DCDC overload has not triggered a wake-up event	0x0*
		WAKEUP_DCDC_OVERLOAD_EVENT_SET	DCDC overload has triggered a wake-up event at least once	0x1
20	BB_TIMER_WAKEUP	WAKEUP_BB_TIMER_EVENT_NOT_SET	BB timer has not triggered a wake-up event	0x0*
		WAKEUP_BB_TIMER_EVENT_SET	BB timer has triggered a wake-up event at least once	0x1
19	GPIO3_WAKEUP	WAKEUP_GPIO3_EVENT_NOT_SET	GPIO3 has not triggered a wake-up event	0x0*
		WAKEUP_GPIO3_EVENT_SET	GPIO3 has triggered a wake-up event at	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
			least once	
18	GPIO2_WAKEUP	WAKEUP_GPIO2_EVENT_NOT_SET	GPIO2 has not triggered a wake-up event	0x0*
		WAKEUP_GPIO2_EVENT_SET	GPIO2 has triggered a wake-up event at least once	0x1
17	GPIO1_WAKEUP	WAKEUP_GPIO1_EVENT_NOT_SET	GPIO1 has not triggered a wake-up event	0x0*
		WAKEUP_GPIO1_EVENT_SET	GPIO1 has triggered a wake-up event at least once	0x1
16	GPIO0_WAKEUP	WAKEUP_GPIO0_EVENT_NOT_SET	GPIO0 has not triggered a wake-up event	0x0*
		WAKEUP_GPIO0_EVENT_SET	GPIO0 has triggered a wake-up event at least once	0x1
11	THRESHOLD_WAKEUP_CLEAR	THRESHOLD_FULL_EVENT_CLEAR	Reset the sticky WAKEUP_THRESHOLD_EVENT flag	0x1
10	FIFO_FULL_WAKEUP_CLEAR	WAKEUP_FIFO_FULL_EVENT_CLEAR	Reset the sticky WAKEUP_FIFO_FULL_EVENT flag	0x1
9	ACOMP_WAKEUP_CLEAR	WAKEUP_ACOMP_EVENT_CLEAR	Reset the sticky WAKEUP_ACOMP_EVENT flag	0x1
8	DCDC_OVERLOAD_WAKEUP_CLEAR	WAKEUP_DCDC_OVERLOAD_EVENT_CLEAR	Reset the sticky WAKEUP_DCDC_OVERLOAD flag	0x1
7	RTC_OVERFLOW_WAKEUP_CLEAR	WAKEUP_RTC_OVERFLOW_EVENT_CLEAR	Reset the sticky WAKEUP_RTC_OVERFLOW_EVENT flag	0x1
6	RTC_CLOCK_WAKEUP_CLEAR	WAKEUP_RTC_CLOCK_EVENT_CLEAR	Reset the sticky WAKEUP_RTC_CLOCK_EVENT flag	0x1
5	RTC_ALARM_WAKEUP_CLEAR	WAKEUP_RTC_ALARM_EVENT_CLEAR	Reset the sticky WAKEUP_RTC_ALARM_EVENT flag	0x1
4	BB_TIMER_WAKEUP_CLEAR	WAKEUP_BB_TIMER_CLEAR	Reset the sticky WAKEUP_BB_TIMER_EVENT flag	0x1
3	GPIO3_WAKEUP_CLEAR	WAKEUP_GPIO3_EVENT_CLEAR	Reset the sticky WAKEUP_GPIO3_EVENT flag	0x1
2	GPIO2_WAKEUP_CLEAR	WAKEUP_GPIO2_EVENT_CLEAR	Reset the sticky WAKEUP_GPIO2_EVENT flag	0x1
1	GPIO1_WAKEUP_CLEAR	WAKEUP_GPIO1_EVENT_CLEAR	Reset the sticky WAKEUP_GPIO1_EVENT flag	0x1
0	GPIO0_WAKEUP_CLEAR	WAKEUP_GPIO0_EVENT_CLEAR	Reset the sticky WAKEUP_GPIO0_EVENT flag	0x1

8.7.0.5 ACS_WAKEUP_CFG

Bit Field	Read/Write	Field Name	Description
18:16	RW	DELAY	Delay from VDDD ready to digital clock enable (power of 2)
11	RW	DCDC_OVERLOAD_EN	Enable / Disable the Wake-up functionality on the DCDC overload flag
9	RW	FIFO_FULL_EN	Enable the wake-up on full FIFO
8	RW	RTC_OVERFLOW_EN	Enable / Disable the Wake-up functionality on RTC overflow flag
7	RW	GPIO3_POL	Wake-up polarity on the GPIO3 pad
6	RW	GPIO2_POL	Wake-up polarity on the GPIO2 pad
5	RW	GPIO1_POL	Wake-up polarity on the GPIO1 pad
4	RW	GPIO0_POL	Wake-up polarity on the GPIO0 pad
3	RW	GPIO3_EN	Enable the wake-up functionality on the GPIO3 pad
2	RW	GPIO2_EN	Enable the wake-up functionality on the GPIO2 pad
1	RW	GPIO1_EN	Enable the wake-up functionality on the GPIO1 pad
0	RW	GPIO0_EN	Enable the wake-up functionality on the GPIO0 pad

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
18:16	DELAY	WAKEUP_DELAY_1	Wait for 1 32kHz clock cycle	0x0
		WAKEUP_DELAY_2	Wait for 2 32kHz clock cycles	0x1
		WAKEUP_DELAY_4	Wait for 4 32kHz clock cycles	0x2
		WAKEUP_DELAY_8	Wait for 8 32kHz clock cycles	0x3
		WAKEUP_DELAY_16	Wait for 16 32kHz clock cycles	0x4*
		WAKEUP_DELAY_32	Wait for 32 32kHz clock cycles	0x5
		WAKEUP_DELAY_64	Wait for 64 32kHz clock cycles	0x6
		WAKEUP_DELAY_128	Wait for 128 32kHz clock cycles	0x7
11	DCDC_OVERLOAD_EN	WAKEUP_DCDC_OVERLOAD_DISABLE	Disable the wake-up functionality on the DCDC overload flag	0x0*
		WAKEUP_DCDC_OVERLOAD_ENABLE	Enable the wake-up functionality on the DCDC overload flag	0x1
9	FIFO_FULL_EN	WAKEUP_FIFO_DISABLE	Disable the wake-up functionality on the ULP data acquisition subsystem FIFO	0x0*
		WAKEUP_FIFO_ENABLE	Enable the wake-up functionality on the ULP data acquisition subsystem FIFO	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
8	RTC_OVERFLOW_EN	WAKEUP_RTC_OVERFLOW_DISABLE	Disable the wake-up functionality on the RTC overflow flag	0x0*
		WAKEUP_RTC_OVERFLOW_ENABLE	Enable the wake-up functionality on the RTC overflow flag	0x1
7	GPIO3_POL	WAKEUP_GPIO3_RISING	Wake-up on the GPIO3 rising edge	0x0*
		WAKEUP_GPIO3_FALLING	Wake-up on the GPIO3 falling edge	0x1
6	GPIO2_POL	WAKEUP_GPIO2_RISING	Wake-up on the GPIO2 rising edge	0x0*
		WAKEUP_GPIO2_FALLING	Wake-up on the GPIO2 falling edge	0x1
5	GPIO1_POL	WAKEUP_GPIO1_RISING	Wake-up on the GPIO1 rising edge	0x0*
		WAKEUP_GPIO1_FALLING	Wake-up on the GPIO1 falling edge	0x1
4	GPIO0_POL	WAKEUP_GPIO0_RISING	Wake-up on the GPIO0 rising edge	0x0*
		WAKEUP_GPIO0_FALLING	Wake-up on the GPIO0 falling edge	0x1
3	GPIO3_EN	WAKEUP_GPIO3_DISABLE	Disable the wake-up functionality on the GPIO3 pad	0x0*
		WAKEUP_GPIO3_ENABLE	Enable the wake-up functionality on the GPIO3 pad	0x1
2	GPIO2_EN	WAKEUP_GPIO2_DISABLE	Disable the wake-up functionality on the GPIO2 pad	0x0*
		WAKEUP_GPIO2_ENABLE	Enable the wake-up functionality on the GPIO2 pad	0x1
1	GPIO1_EN	WAKEUP_GPIO1_DISABLE	Disable the wake-up functionality on the GPIO1 pad	0x0*
		WAKEUP_GPIO1_ENABLE	Enable the wake-up functionality on the GPIO1 pad	0x1
0	GPIO0_EN	WAKEUP_GPIO0_DISABLE	Disable the wake-up functionality on the GPIO0 pad	0x0*
		WAKEUP_GPIO0_ENABLE	Enable the wake-up functionality on the GPIO0 pad	0x1

8.7.0.6 ACS_WAKEUP_STATE

Bit Field	Read/Write	Field Name	Description
19:16	R	WAKEUP_SRC	Status register indicates the last wake-up source
7:0	R	RTC_VALUE	RTC counter value captured at wakeup event (only 8 LSBs, corresponds to 7.8 ms)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
19:16	WAKEUP_SRC	WAKEUP_DUE_TO_RTC_OVERFLOW	The last wake-up was due to the RTC Timer overflow	0xB
		WAKEUP_DUE_TO_RTC_CLOCK	The last wake-up was due to the RTC Timer clock	0xA
		WAKEUP_DUE_TO_RTC_ALARM	The last wake-up was due to the RTC Timer alarm	0x9
		WAKEUP_DUE_TO_THRESHOLD	The last wake-up was due to the ULP data acquisition subsystem threshold	0x8
		WAKEUP_DUE_TO_FIFO	The last wake-up was due to the ULP data acquisition subsystem FIFO	0x7
		WAKEUP_DUE_TO_ACOMP	The last wake-up was due to the Analog comparator	0x6
		WAKEUP_DUE_TO_DCDC_ OVERLOAD	The last wake-up was due to the DCDC overload	0x5
		WAKEUP_DUE_TO_BB_TIMER	The last wake-up was due to the baseband timer alarm	0x4
		WAKEUP_DUE_TO_GPIO3	The last wake-up was due to the GPIO3 pad	0x3
		WAKEUP_DUE_TO_GPIO2	The last wake-up was due to the GPIO2 pad	0x2
		WAKEUP_DUE_TO_GPIO1	The last wake-up was due to the GPIO1 pad	0x1
		WAKEUP_DUE_TO_GPIO0	The last wake-up was due to the GPIO0 pad	0x0*

8.7.0.7 ACS_BOOT_CFG

Bit Field	Read/Write	Field Name	Description
8	RW	PADS_RETENTION_EN	Enable / Disable the retention mode of the pads
6	RW	BOOT_ROT_BYPASS	Boot bypass execution of root of trust
5	RW	BOOT_PWR_CAL_BYPASS	Boot bypass execution of system calibration
4:3	R	RC_CLOCK_FSEL	RC oscillator clock multiplier read only flag (mirror of CLOCK_ MULT of ACS_RCOSC_CTRL register)
2	R	RC_FTRIM_FLAG	RC oscillator trimming read only flag (mirror of FTRIM_FLAG of ACS_RCOSC_CTRL register
1:0	RW	BOOT_SELECT	Boot selection to indicate boot source

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
8	PADS_RETENTION_EN	PADS_RETENTION_DISABLE	Disable the pad retention mode	0x0*
		PADS_RETENTION_ENABLE	Enable the pad retention mode	0x1
6	BOOT_ROT_BYPASS	BOOT_ROT_BYPASS_DISABLE	ROM will execute RoT	0x0*
		BOOT_ROT_BYPASS_ENABLE	ROM will not execute RoT	0x1
5	BOOT_PWR_CAL_ BYPASS	BOOT_PWR_CAL_BYPASS_ DISABLE	ROM will execute calibration of system	0x0*
		BOOT_PWR_CAL_BYPASS_ENABLE	ROM will not execute calibration of system	0x1
4:3	RC_CLOCK_FSEL	RC_OSC_STATUS_3MHZ	The RC Oscillator is at 3 MHz	0x0*
		RC_OSC_STATUS_12MHZ	The RC Oscillator is at 12 MHz	0x1
		RC_OSC_STATUS_24MHZ	The RC Oscillator is at 24 MHz	0x2
		RC_OSC_STATUS_48MHZ	The RC Oscillator is at 48 MHz	0x3
2	RC_FTRIM_FLAG	RC_OSC_STATUS_UNCALIBRATED	The oscillators are not calibrated	0x0*
		RC_OSC_STATUS_CALIBRATED	The oscillators are calibrated	0x1
1:0	BOOT_SELECT	BOOT_FLASH_XTAL_DISABLE	The CM33 executes code from the flash and the XTAL will not be started at boot	0x0*
		BOOT_CUSTOM	The CM33 executes 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 CM33 executes code from the flash and the XTAL will be started at boot with the default trim	0x2
		BOOT_FLASH_XTAL_CUSTOM_ TRIM	The CM33 executes code from the flash and the XTAL will be started at boot with trim from ACS_WAKEUP_GP_DATA	0x3

8.7.0.8 ACS_BOOT_GP_DATA

Bit Field	Read/Write	Field Name	Description
31:0	RW	GP_DATA	32-bit General-Purpose RW Data

8.7.0.9 ACS_PWR_CTRL

Bit Field	Read/Write	Field Name	Description
27	RW	SENSOR_PWR_EN	Sensor power control
26	RW	SENSOR_ISOLATE	Sensor isolation control
25	RW	CCAO_PWR_EN	CryptoCell always on power control
24	RW	CCAO_ISOLATE	CryptoCell always on isolation control
23:0	W	POWER_KEY	Write a key to enable the write to power controls

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
27	SENSOR_PWR_EN	ACS_SENSOR_SHUTDOWN	Sensor is powered off	0x0
		ACS_SENSOR_POWERED	Sensor is powered on	0x1*
26	SENSOR_ISOLATE	ACS_SENSOR_NOT_ISOLATE	Sensor is not isolate	0x0*
		ACS_SENSOR_ISOLATE	Sensor is isolate	0x1
25	CCAO_PWR_EN	ACS_CCAO_SHUTDOWN	CryptoCell Always on is powered off	0x0
		ACS_CCAO_POWERED	CryptoCell Always on is powered on	0x1*
24	CCAO_ISOLATE	ACS_CCAO_NOT_ISOLATE	CryptoCell Always on is not isolate	0x0*
		ACS_CCAO_ISOLATE	CryptoCell Always on is isolate	0x1
23:0	POWER_KEY	ACS_PWR_KEY	Write 24-bit key to enable the write access to power controls	0x63412B

8.8 TIMING

The RSL15 system performs a number of steps, following resets and on transitions to or from various power modes, to ensure that when entering into Run Mode the user application can properly use and configure (or reconfigure) various system components needed by the application.

The following sub-sections discuss system timing for system startup (following a power-on reset), as well as for entering and exiting Sleep and Standby Modes.

8.8.1 Startup Timing

Startup timing starts when a supply voltage is provided to the battery supply voltage (VBAT) and Digital Output Supply Voltage (VDDO). For more information on these power supply inputs, see Section 1.1 “Power Supply Inputs ” on page 1.

A diagram showing the startup timing until the system reaches Run Mode is shown in the "Startup Timing" figure (Figure 42).

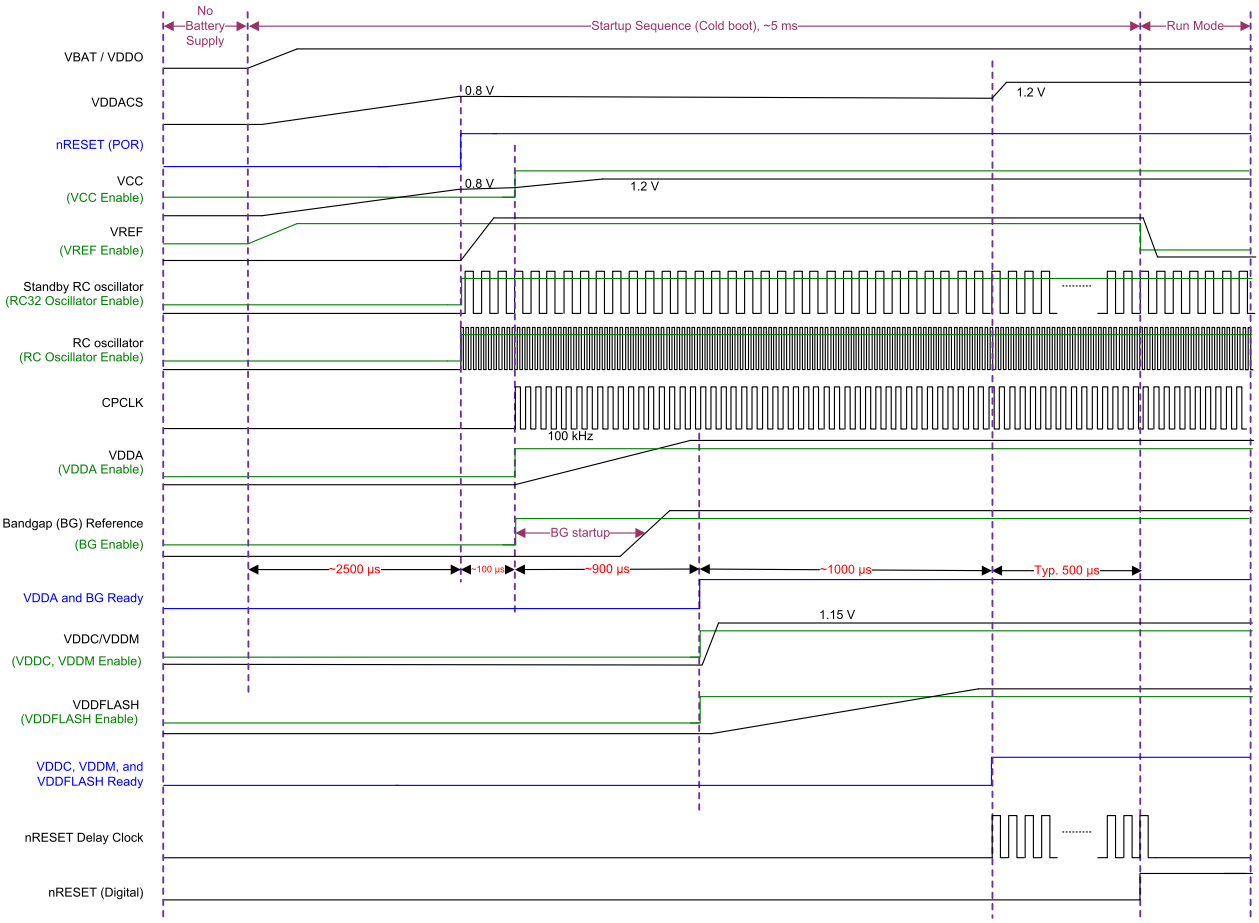


Figure 42. Startup Timing

Timing until a user application starts executes will differ from this diagram for a number of reasons, including:

- If the system is starting up following a full reset, delays awaiting the charging and rise of VBAT, VDDACS, and VREF can be omitted if the system power has remained stable leading up to the reset.

  NOTE: Charging of power supplies at startup can vary significantly, depending on the current and voltage of the supply provided and existing power supply state. To accommodate the worst case startup timing, the "Startup Timing" figure (Figure 42) shows the expected worst case timing until nRESET is released.

- For devices executing in a secure state, there is additional startup time in Run Mode for the execution of the Program ROM and potentially a secure bootloader. This additionally needs to account for:
  - Key certificates
  - Content certificates
  - Debug certificates
  - Application authentication, loading, and/or decryption.

For more information about operation in secure modes, see the *RSL15 Security User's Guide*. For more information on the example secure bootloader provided, see the *Secure Bootloader Usage* guide.

8.8.2 Sleep Power Mode Related Timing

A diagram providing the timing between when a device in Sleep Mode receives a wakeup event and when the system reaches Run Mode again is shown in Figure 43.

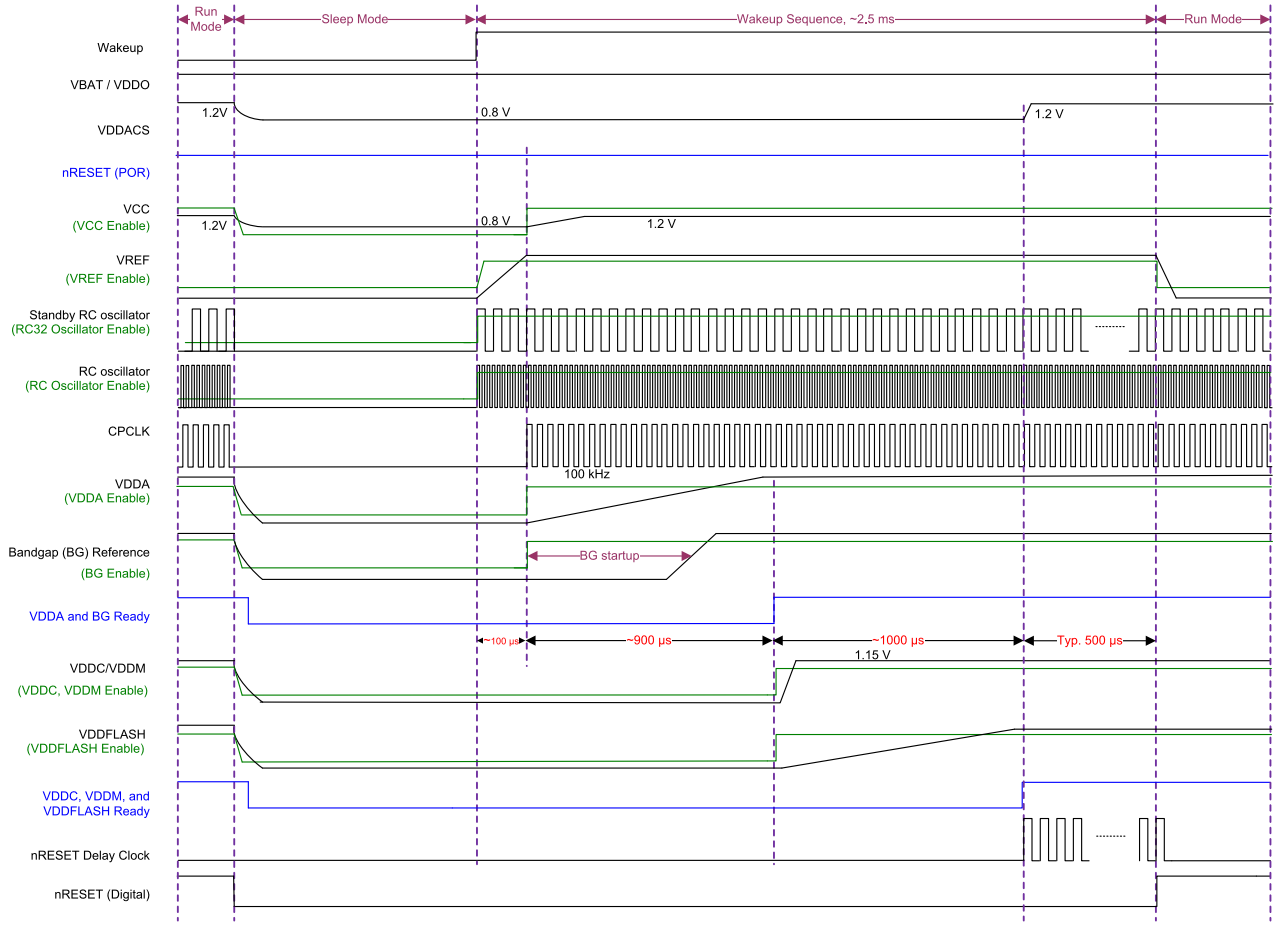


Figure 43. Sleep Mode and Wakeup From Sleep Mode Timing

The "Sleep Mode and Wakeup From Sleep Mode Timing" figure (Figure 43) includes all expected delays in the wakeup configuration for Sleep Mode. Wakeup timing can vary from the timing shown for a number of different reasons, including:

- Variations in Sleep Mode configuration, as discussed in Section 8.6 "Power Modes" on page 431. Elements that are not disabled in a specific Sleep Mode configuration do not require delays during wakeup.
- The desired wakeup delay as controlled by the nRESET delay clock, and configured in the `ACS_WAKEUP_CFG_DELAY` field of the `ACS_WAKEUP_CFG` register.

An example current capture from a sample application showing power consumption of the RSL15 device relative to the wakeup timing is provided in the "Example wakeup, Bluetooth advertisement, return to Sleep Mode" figure (Figure 44). This sample application shows a wakeup from Sleep Mode due to an event (at m1), transmits a Bluetooth Low Energy advertisement without receiving a response once in Run Mode, and returns to Sleep Mode (at m2).

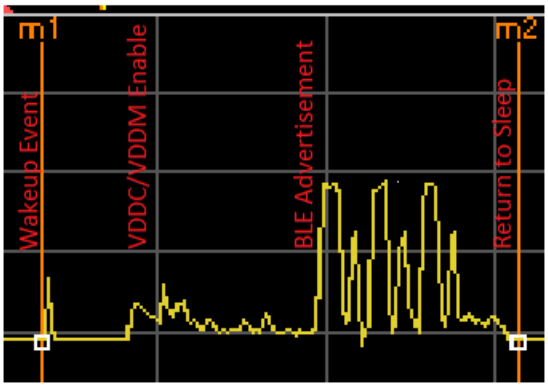


Figure 44. Example wakeup, Bluetooth advertisement, return to Sleep Mode

8.8.3 Standby Power Mode Related Timing

A diagram providing the timing between when a device in Standby Mode receives a wakeup event and when the system reaches Run Mode again is shown in the "Standby Mode and Wakeup From Standby Mode Timing" figure (Figure 45).

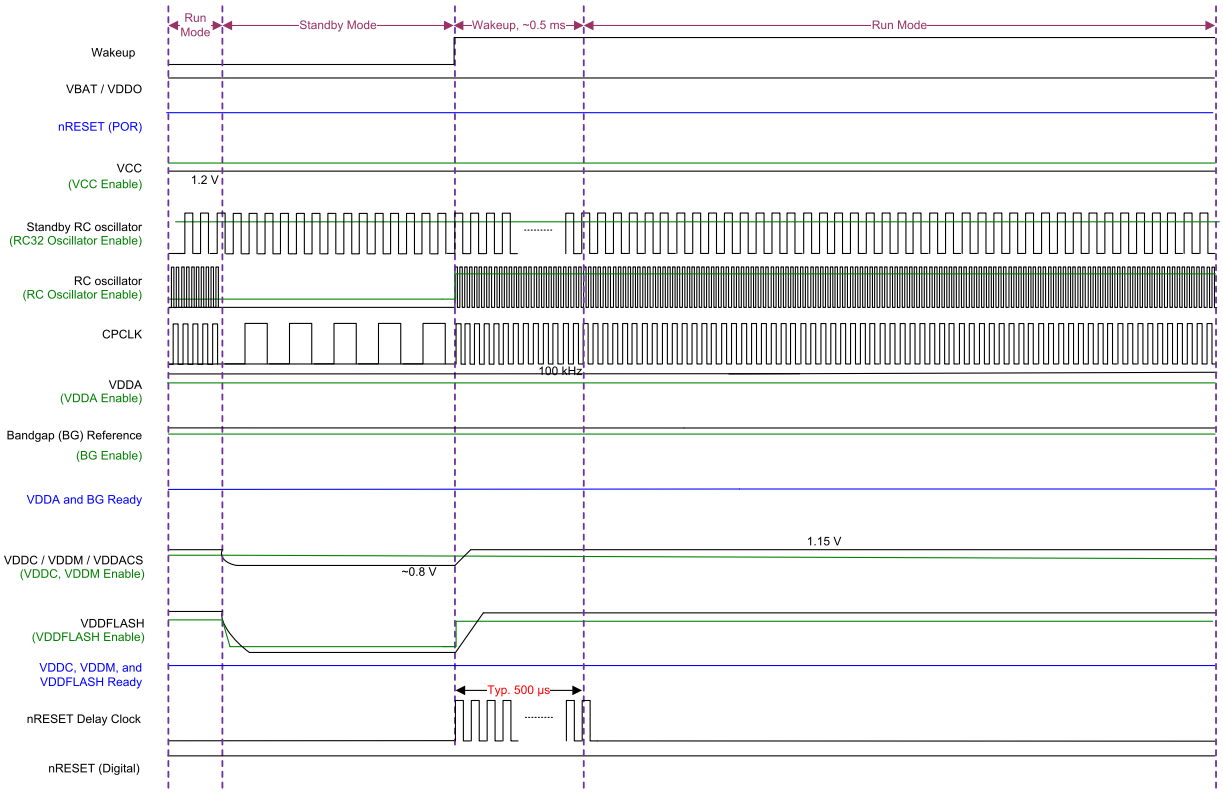


Figure 45. Standby Mode and Wakeup From Standby Mode Timing

The "Standby Mode and Wakeup From Standby Mode Timing" figure (Figure 45) includes all expected delays in the wakeup configuration for Standy Mode. Since most of the critical items remain active in Standby Mode, wakeup timing in Standby Mode is largely determined by the configured wakeup delay. This delay can by configured by adjusting the nRESET delay clock that controls the wakeup delay configuration using the `ACS_WAKEUP_CFG_DELAY` field of the `ACS_WAKEUP_CFG` register.

CHAPTER 9

Memory

The memory systems provided by the RSL15 system are constructed using a number of memories (memory instances), memory buses, memory controllers, and memory arbiters. These memories and other memory-mapped elements are addressable by the Arm Cortex-M33 processor and its peripherals.

9.1 ARCHITECTURE

The RSL15 system uses a memory architecture based on the pre-defined memory map of the Arm Cortex-M33 processor.

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-M33 possessor, although some interfaces and peripherals such as the DMA and Bluetooth baseband provide additional access paths to specific memory elements. The connections to the components that make up the memory for the RSL15 system are shown in the "System Memory Architecture" figure (Figure 46).

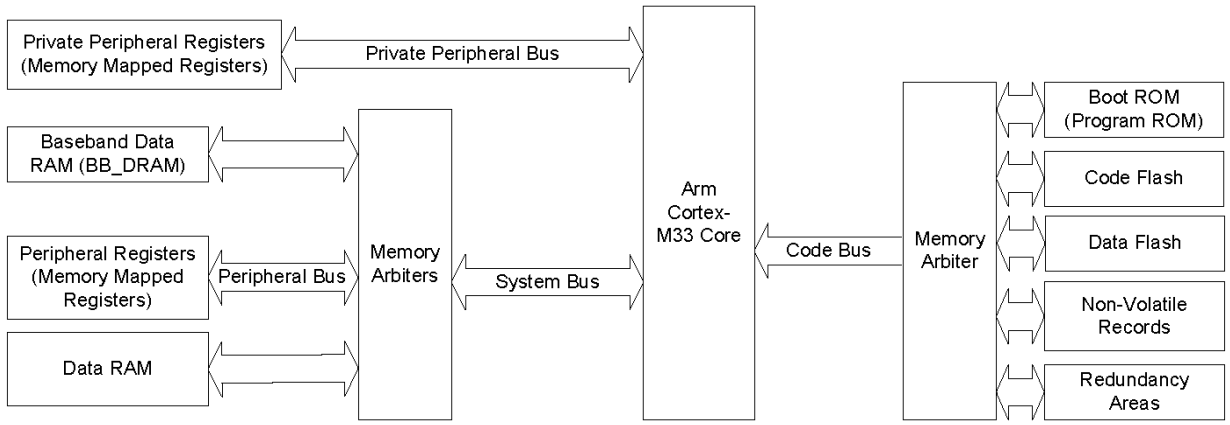


Figure 46. System Memory Architecture

9.1.1 Memory Instances

The memory architecture for RSL15, including the memory instances, registers, and other components, are accessible from the Arm Cortex-M33 processor through one or more of the processor's standard buses. All the memory instances are shown in the "RSL15 Memory Instances" table (Table 15).

NOTE: RSL15 is produced in two variants with different amounts of flash memory for each variant; the "RSL15 Memory Instances" table (Table 15) below lists memory sizes for each variant, marked as RSL15-512 and RSL15-284 respectively.

Table 15. RSL15 Memory Instances

Instance Name	Size	Type	Start Address
Boot ROM (Program ROM)	20 KB	ROM	0x000000000
Flash Code Redundancy	2 KB x 2	Flash	0x00060000, 0x00060800
Flash Data Redundancy	256 B x 2	Flash	0x00061000, 0x00061100
Flash Non-Volatile Records	256 B x 8	Flash	0x00080000, 0x00080100, 0x00080200, ..., 0x00080700
Flash Code	352 KB (RSL15-512)	Flash	0x00100000
	264 KB (RSL15-284)		
Flash MNVR	128 B + 128 B	Flash	00080800
Flash Data	160 KB (RSL15-512)	Flash	0x00158000
	20 KB (RSL15-284)		
Data RAM	8 KB x 8	RAM	0x20000000, 0x20002000, 0x20004000, ..., 0x2000E000
Baseband Data RAM	8 KB x 2	RAM	0x20010000, 0x20012000

The memory instances can be disabled when not in use, to reduce power consumption. The data RAM and baseband data RAM instances can also be placed into a retention mode, where the contents of the memories are maintained but cannot be accessed. In total, there are five memory power states:

Disabled

If the memory instance's corresponding bit in the `SYSCTRL_MEM_POWER_STARTUP` register is cleared, the memory is in disabled mode. In this mode, memory is not powered and memory isolation is enabled. Memory instances do not retain their state in disabled mode, and are inaccessible.

When a memory instance is in disabled mode, its corresponding bits in the `SYSCTRL_MEM_ACCESS_CFG` and `SYSCTRL_MEM_POWER_ENABLE` registers are cleared. To enable a memory instance in disabled mode, the instance must first be placed into the power up or wakeup modes.

Retention

If the data RAM or baseband data RAM memory instance's corresponding bits in the `SYSCTRL_MEM_POWER_STARTUP` and `SYSCTRL_MEM_RETENTION_CFG` registers are set, the memory is placed into retention mode when using low power modes. In this mode, the memories are held in a low-power state, and memory isolation is enabled. Memory instances retain their state in retention mode, but are inaccessible.

When a memory instance is in retention mode, the corresponding bits in the `SYSCTRL_MEM_POWER_ENABLE` and `SYSCTRL_MEM_ACCESS_CFG` registers are cleared. To enable a memory instance in retention mode, the instance must first be placed into the power up or wakeup modes.

Power up

If the memory instance's corresponding bit in the SYSCTRL_MEM_POWER_STARTUP register is set, and the corresponding bits in the SYSCTRL_MEM_POWER_ENABLE, SYSCTRL_MEM_ACCESS_CFG and SYSCTRL_MEM_RETENTION_CFG registers is cleared, the memory is in power up mode. In this mode, memories are powered one-by-one at 200 ns intervals and held in a normal power state with memory isolation once powered.

This mode is a transitional state between disabled or retention and enabled modes, where memory instances remain inaccessible. Once all memories are powered they are automatically held in the power up state for a minimum of 1.3 µs.

Wakeup

If the memory instance's corresponding bits in the SYSCTRL_MEM_POWER_STARTUP and SYSCTRL_MEM_POWER_ENABLE registers is set, and the corresponding bits in the SYSCTRL_MEM_ACCESS_CFG and SYSCTRL_MEM_RETENTION_CFG registers is cleared, the memory is in wakeup mode. In this mode, memory is powered and held in a normal power state with memory isolation.

This mode is a transitional state between disabled or retention and enabled modes, where memory instances remain inaccessible. Memories are held in the wakeup state for a minimum of 1.3 µs.

Enabled

If the memory instance's corresponding bits in SYSCTRL_MEM_POWER_STARTUP, SYSCTRL_MEM_POWER_ENABLE, and SYSCTRL_MEM_ACCESS_CFG registers are set, the memory is in Enabled mode. In this mode, memory is held in a normal power mode, with memory isolation disabled. Memories are fully functional in enabled mode.

Accesses outside of defined memory areas produce a memory fault or ACCESS_ERROR interrupt. Accesses to a memory that is not in enabled mode produce a bus fault or ACCESS_ERROR interrupt:

- If the failed memory access occurs on the Arm Cortex-M33 processor, a memory fault is generated for access outside of a defined memory area and a bus fault is generated for accesses to memories that are not enabled.
- If the failed memory access occurs on any other access, an ACCESS_ERROR has occurred. The SYSCTRL_ACCESS_ERROR register indicates if the error occurred on an access from:
    - The DMA (SYSCTRL_ACCESS_ERROR_DMA_MEM_ERROR)
    - The Bluetooth baseband (SYSCTRL_ACCESS_ERROR_BB_MEM_ERROR)
    - The flash copier (SYSCTRL_ACCESS_ERROR_FLASH_COPIER_MEM_ERROR)
    - The Arm CryptoCell-312 (SYSCTRL_ACCESS_ERROR_CC312_MEM_ERROR)

An ACCESS_ERROR interrupt can also be triggered when the DMA accesses a RAM or peripheral, while the TrustZone security state is not aligned with the DMA security state. If this occurs, the SYSCTRL_ACCESS_ERROR_DMA_PERIPH_ERROR bit in the SYSCTRL_ACCESS_ERROR register is set.

To clear the ACCESS_ERROR interrupt status bits, write SYSCTRL_ACCESS_ERROR_CLEAR to the SYSCTRL_ACCESS_ERROR_ACCESS_ERROR_CLEAR bit from the SYSCTRL_ACCESS_ERROR register.

IMPORTANT: To minimize power consumption, place all unused memories in the disabled state.

9.1.2 Memory Buses

Buses connected to the Arm Cortex-M33 processor implement the standard Arm Cortex-M33 core memory map. All buses share the same 32-bit Arm Cortex-M33 processor memory space. These buses can also be seen in Figure 46 on page 466, and are as follows:

Code Bus
: Allows the Arm Cortex-M33 processor to fetch instruction and data information from the Arm Cortex-M33 flash memory instances. Instructions can also be fetched from the ROM instance.

System Bus
: Allows the Arm Cortex-M33 processor to fetch instructions and data information from the Arm Cortex-M33 core's data memory instances. This bus also provides access to the Bluetooth baseband's data RAM and Peripheral Bus.

Peripheral Bus
: Allows the Arm Cortex-M33 processor to access memory-mapped peripherals

DMA Memory Bus
: Memory bus shared by the DMA, Arm CryptoCell-312, and flash copier peripherals

9.1.3 Memory Arbitration

In front of each memory instance, an arbiter manages the simultaneous accesses between the masters - the Arm Cortex-M33 processor (uC), the baseband controller (BB), and the DMA memory bus (DMA).

- Data RAM and peripheral bus accesses use a fixed arbitration scheme where accesses by the Arm Cortex-M33 processor have higher priority than accesses from the DMA (uC > DMA).
- Baseband Data RAM selects between two fixed arbitration schemes:
    - If the baseband is using a divided clock source, the priority order from highest to lowest is the Bluetooth baseband, Arm Cortex-M33 processor, and DMA (BB > uC > DMA).
    - Otherwise, the priority order from highest to lowest is Arm Cortex-M33 processor, DMA, and Bluetooth baseband (uC > DMA > BB).

The DMA Memory Bus also uses a fixed arbitration scheme for accesses, with the priority order of peripheral accesses to the bus being (from highest to lowest priority) the DMA controller, the Arm CryptoCell-312, and the flash copier.

9.2 Memory Map and Usage

The memory map provided for the RSL15 device follows the standard Arm Cortex-M33 processor memory layout. All memory instances and memory-mapped elements are accessible through this memory map, as shown in the "Default RSL15 Memory Map" figure (Figure 47).

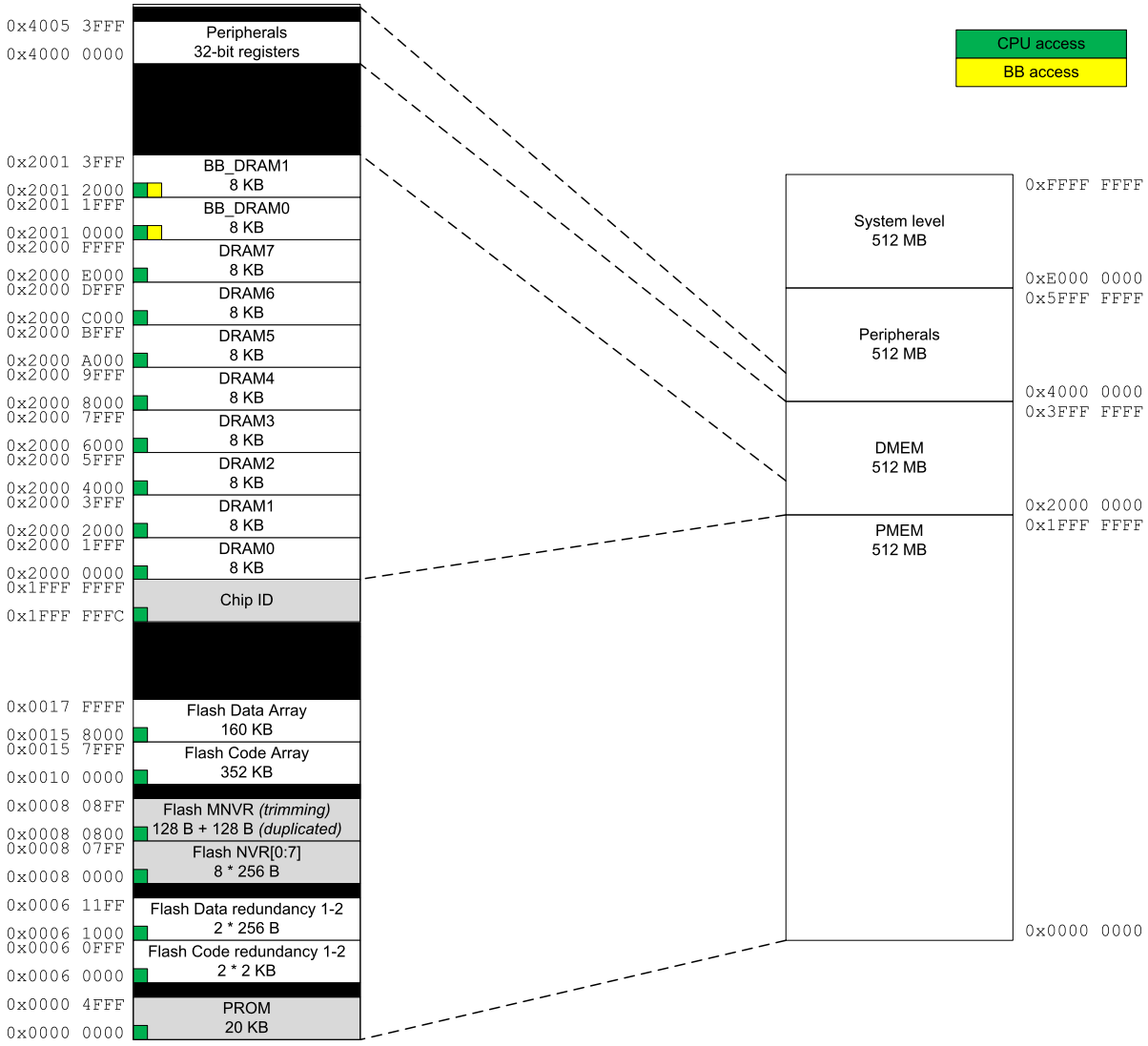


Figure 47. Default RSL15 Memory Map

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 *RSL15 Firmware Reference Manual*
2. Flash Memory
3. Data RAM
4. Other memory mapped areas

9.2.1 Flash Memory

RSL15 contains a flash memory instance that provides the following areas that can be written through the flash memory interface:

Code Flash
: An area of flash optimized for use as larger (2 KB) blocks of memory. This area is typically used to store program code.

Data Flash
: An area of flash optimized for use as smaller (256 B) blocks of memory. This area is typically used to store static data or additional program code, including memory reserved for the Arm CryptoCell-312 (DEU reserved), and Bluetooth bond records (bond info).

Non-Volatile Records
: A set of manufacturing-specific and other non-volatile records. These records are used to store elements used by the Arm CryptoCell-312 (NVR0 to NVR3), manufacturing and calibration records (NVR7, MNVR), and user records (NVR4 to NVR6).

Redundancy Sectors
: A set of four redundancy sectors that can be used to replace two code flash and two data flash sectors.

For more information about the flash memory, see Section 9.3 “Flash” on page 474.

9.2.1.1 Manually Configuring Memory Allocation

Two changes to particular defines allow for an increase or decrease in the number of sectors in the bondlist, in the files *rsl15_map.h* and *bondlist.h*:

Navigate to *PROJDIR\Includes\C:\Users\username\AppData\Local\Arm\Packs\ONSemiconductor\RSL15\1.4.187\firmware\include\rsl15_map.h*, and change the following define, as shown in the "Definition to Change in rsl_map.h" figure (Figure 50):

```
#define FLASH_BOND_INFO_SIZE
#define FLASH_BOND_INFO_TOP
```

Navigate to *PROJDIR\Includes\C:\Users/username\AppData\Local\Arm\Packs\ONSemiconductor\RSL15\1.4.187\firmware\source\lib\ble_abstraction\ble_common\include\bondlist.h*, and change the following define, as shown in the "Definition to Change in bondlist.h" figure (Figure 48):

```
#define BOND_INFO_FLASH_SECTORS_COUNT
```

rsl15_map.h

```
278 /*------------------------------------------------------------------------------*/
279 #define FLASH0_CODE_BASE                0x00100000
280 #define FLASH0_CODE_RSL15_284_TOP       0x00141FFF
281 #define FLASH0_CODE_RSL15_284_SIZE      (FLASH0_CODE_RSL15_284_TOP - FLASH0_CODE_BASE + 1)
282
283 #define FLASH0_CODE_TOP                 0x00157FFF
284 #define FLASH0_CODE_SIZE                (FLASH0_CODE_TOP - FLASH0_CODE_BASE + 1)
285
286 #define FLASH0_DATA_BASE                0x00158000
287 #define FLASH0_DATA_RSL15_284_TOP       0x0015CFFF
288 #define FLASH0_DATA_RSL15_284_SIZE      (FLASH0_DATA_RSL15_284_TOP - FLASH0_DATA_BASE + 1)
289
290 #define FLASH0_DATA_TOP                 0x0017FFFF
291 #define FLASH0_DATA_SIZE                (FLASH0_DATA_TOP - FLASH0_DATA_BASE + 1)
292
293 /* Memory reservations for security and Bluetooth use cases */
294 #define FLASH_DEU_RESERVED_BASE         FLASH0_DATA_BASE
295 #define FLASH_DEU_RESERVED_SIZE         0xC00
296 #define FLASH_DEU_RESERVED_TOP          (FLASH0_DATA_BASE + FLASH_DEU_RESERVED_SIZE - 1)
297
298 #define FLASH_BOND_INFO_BASE            (FLASH_DEU_RESERVED_TOP + 1)
299 #define FLASH_BOND_INFO_SIZE            0x800
300 #define FLASH_BOND_INFO_TOP             (FLASH_BOND_INFO_BASE + FLASH_BOND_INFO_SIZE - 1)
301
302 #define FLASH_MESH_INFO_BASE            (FLASH_BOND_INFO_TOP + 1)
303 #define FLASH_MESH_INFO_SIZE            0x1000
304 #define FLASH_MESH_INFO_TOP             (FLASH_MESH_INFO_BASE + FLASH_MESH_INFO_SIZE - 1)
305 /*------------------------------------------------------------------------------
306  * Device Information
307  *------------------------------------------------------------------------------*/
308 #define DEVICE_INFO_BASE                FLASH0_NVR5_BASE
309 #define DEVICE_INFO_BLUETOOTH_ADDR      (DEVICE_INFO_BASE + 0x00)
310 #define DEVICE_INFO_BLUETOOTH_IRK       (DEVICE_INFO_BASE + 0x10)
311 #define DEVICE_INFO_BLUETOOTH_CSRK      (DEVICE_INFO_BASE + 0x20)
312
313 /*------------------------------------------------------------------------------*/
314 /*------------------------------------------------------------------------------*/
315 /* System Bus Memory Structures                                                 */
316 /*------------------------------------------------------------------------------*/
317 /*------------------------------------------------------------------------------*/
318
319 /*------------------------------------------------------------------------------*/
```

bondlist.h

```
46  ····uint8_t addr_type;··········/**< Address type */
47  ····uint8_t irk_exchanged;·····/**< Non-zero if IRK has been exchanged */
48  ····uint8_t csrk[16];···········/**< Connection resolving signature key */
49  ····uint8_t irk[16];············/**< Identity resolving key */
50  ····uint8_t rand[8];············/**< Random number */
51  } BondInfo_t;·····················/**< 72 bytes */
52
53  /** Start address and size of the memory region where bond list is stored:
54   *  ····BOND_INFO_BASE: 3KB from base of data flash array
55   ····························(the first 3KB of data flash array is reserved to ROM use)
56   *  ····BOND_INFO_FLASH_SECTORS_COUNT: 8 sectors (2KB)
57   *  Notes: if needed, user application can re-define BOND_INFO_BASE and
58   *  BOND_INFO_FLASH_SECTORS_COUNT. Any increase to the number of sectors
59   *  will also need to be updated in the linker script.
60   */
61  #ifndef BOND_INFO_BASE
62  #define BOND_INFO_BASE··········FLASH_BOND_INFO_BASE····/**< Start of address for bond info */
63  #endif····/* ifndef BOND_INFO_BASE */
64
65  #ifndef BOND_INFO_FLASH_SECTORS_COUNT
66  #define BOND_INFO_FLASH_SECTORS_COUNT·····8·····/**< Number of sectors for bond info */
67  #endif····/* ifndef BOND_INFO_FLASH_SECTORS_COUNT */
68
69  #if BOND_INFO_FLASH_SECTORS_COUNT < 1
70  ····#error "The number of flash sectors should be greater than 1"
71  #endif····/* if BOND_INFO_FLASH_SECTORS_COUNT < 1 */
72
73  /** The maximum number of bond information records that can be stored in the memory region that is
74   *  ····reserved for the bondlist (Default is 2KB -> 8 sectors -> ~28 records)
75   */
76  #define BONDLIST_MAX_SIZE··············((DATA_SECTOR_LEN_BYTES * BOND_INFO_FLASH_SECTORS_COUNT) / \
77  ·····································sizeof(BondInfo_t))
```

Figure 48. Definition to Change in bondlist.h

Changes must also be made to the linker script (*sections.ld*). The length of the bondlist details the length of `FLASH_BOND_INFO_SIZE` in bytes.

The origin of `FLASH_DATA` is 1 byte larger than `FLASH_BOND_INFO_TOP`, and the length of `FLASH_DATA` is the remaining memory. This can be calculated as 160 KB minus the used memory. These areas to change are shown in the "Values to Change in sections.ld" figure (Figure 49)

```
/* The ROM requires the first 3K of data flash for DEU transfer space */
FLASH_RSVD (xrw)    : ORIGIN = 0x00158000, LENGTH = 3K

/* Reserve 2k for Bluetooth   information */
FLASH_BOND_RSVD (xrw)    : ORIGIN = 0x00158C00, LENGTH = 2K

/* The rest of the data flash is available for application use */
FLASH_DATA (xrw)    : ORIGIN = 0x00159400, LENGTH = 155K
```

Figure 49. Values to Change in sections.ld

9.2.2 Data Memory

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

- 64 KB of DRAM are shared between the Bluetooth stack and the user application. The DRAM is subdivided into 8 instances of 8192 bytes (2048 words) as DRAM0 - DRAM7. These instances are used to store any type of data needed for user applications.
- 16 KB of BB DRAM act as the exchange memory between the Arm Cortex-M33 processor and the baseband controller. The BB DRAM is subdivided into 2 instances of 8192 bytes (2048 words) as BB_DRAM0 and BB_DRAM1. These instances are directly accessible by the Arm Cortex-M33 processor and the DMA, parallel to the baseband controller.

9.2.3 Other Memory Mapped Areas

9.2.3.1 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.

9.2.3.2 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-M33 processor.

Figure 50. Definition to Change in rsl_map.h

9.3 FLASH

The RSL15 device includes a flash memory instance for use as non-volatile memory. The topics that follow describe its characteristics and use.

9.3.1 Flash Characteristics

The flash memory instance contains:

- A main flash area consisting of two flash arrays:
  - For RSL15-512 devices, this flash memory provides 512 KB of flash memory divided into a code flash array of 352 KB and a data flash array of 160 KB.
  - For RSL15-284 devices, this flash memory provides 284 KB of flash memory divided into a code flash array of 264 KB and a data flash array of 20 KB.
  - The code flash array is organized into 2048-byte sectors; the data flash array is organized into 256-byte sectors. Both arrays are accessed using the same memory buses, so it is equally efficient to execute code or access data stored to either array.
- Eight non-volatile records (NVR), each consisting of one flash sector of 256 bytes
- One manufacturing non-volatile record (MNVR), consisting of one flash sector that contains two copies of the 128 bytes of redundant data programmed during production
- Redundancy sectors, including:
  - Two sectors for production patching of the code flash array
  - Two sectors for production patching of the data flash array

NOTE: Each user redundancy sector can be set only once, but can be used repeatedly just like any other sector of memory.

Data in this memory is stored as 38-bit words, including six bits of parity data (for error correction coding) for every four bytes of user data.

The flash memory is supported by:

- Timing circuitry configured to ensure proper flash operation. For more information, see Section 9.3 “Flash”.
- Redundancy sectors, including both production sectors (as listed above) and user redundancy sectors, which can be configured as described in Section 1.0.1 “ Redundancy Sectors” on page 1.

  NOTE: User redundancy sector configuration is the only component of the MNVR record that can be updated outside of production, and each user redundancy sector can be programmed a maximum of one time.
- Error correcting support hardware, as described in Section 1.0.1 “Error-Correction Coding” on page 1.
- A flash copier that can be used to validate the flash contents. For more information, see Section 13.5 “Flash Copier” on page 705.

9.3.2 Flash Delays

The flash instance provided as part of the RSL15 system uses specific timing relative to the SYSCLK frequency to ensure proper flash operation. Relevant flash delays include:

- Read operation delays of 30 ns to code flash sectors and 40 ns to data flash sectors, implemented using wait states on flash read accesses if the clock period is known to be too short to ensure an appropriate delay
- Programming operation delays of 9.6 to 12 µs, including a 1.6 to 2 µs pre-program pulse and an 8 to 10 µs programming pulse
- Sector erase delays of 0.8 to 1 ms
- Mass erase delays of 8 to 10 ms

To ensure that the timing requirements of reading from flash are met, the `FLASH_DELAY_CTRL` register needs to be configured for a frequency that is at least as high as the highest possible clock frequency at all times. To ensure proper operation for flash programming and erase cycles, the `FLASH_DELAY_CTRL` register needs to be configured to match a known SYSCLK frequency that is sourced from the crystal oscillator or a calibrated RC clock (including temperature stabilization), as an error of greater than ±10% is unacceptable for flash write operations.

NOTE: The flash program and erase operations cannot be supported if the clock has more than 10% frequency error. Because the system RC oscillator has a greater than 10% variation over system operating conditions, we recommend use of the system XTAL oscillator for all program and erase operations.

NOTE: When executing from STANDBYCLK, the `FLASH_DELAY_CTRL` register needs to be configured to use the setting for a SYSCLK of 3 MHz (`FLASH_DELAY_FOR_SYSCLK_3MHZ`), and programming is not possible.

9.3.3 Flash Memory Operations

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 RSL15 program ROM, and can be used to write to the flash at any time. Refer to the*RSL15 Firmware Reference* for more information.

Commands are issued explicitly to the flash interface by writes to the `FLASH_CMD_CTRL_COMMAND` bit-field of the `FLASH_CMD_CTRL` register, or as side effects of writes to the `FLASH_IF_CTRL` register that change the value of the `FLASH_IF_CTRL_VREAD1_MODE`, `FLASH_IF_CTRL_RECALL`, or `FLASH_IF_CTRL_LP_MODE` bits. Commands that can

be executed on the flash are listed in the "Flash" table (9.3). All commands return to the idle state, except the sequential programming (CMD_PROGRAM_SEQ) and the deep power down (CMD_DPD) commands. If the flash is already busy, the command is ignored along with writes to all other flash registers, except:

- FLASH_DATA* when new data is requested for a CMD_PROGRAM_SEQ command
- FLASH_CMD_CTRL_CMD_END from FLASH_CMD_CTRL, which can be used to terminate the current command
- FLASH_DELAY_CTRL, which updates the flash timing as described in Section 9.3 “Flash”
- FLASH_ECC_CTRL, which updates the use of the error correction coding as described in Section 1.0.1 “Error-Correction Coding”

CAUTION: While changing FLASH_ECC_CTRL is possible while executing a flash command, changing the FLASH_ECC_CTRL_CMD_ECC_CTRL bit while executing a read or write command results in undefined behavior for the flash operation. Since this can result in an unknown value written to or read from flash, this bit-field must not be changed while writing or reading from flash.

NOTE: While any flash operations are in progress, the flash memory instance cannot be accessed from the Arm Cortex-M33 processor's code bus. This includes any case where a flash command is issued through a write to the FLASH_CMD_CTRL register or when a low-level flash command is triggered by writing to the FLASH_IF_CTRL register. Access to the flash while a flash operation is being processed results in a bus fault.

Table 16. Flash Low-Level Commands

Command	Description
CMD_IDLE	No flash command is active or pending.
CMD_WAKE_UP	This power-up sequence starts automatically when the flash isolation is removed through the SYSCTRL_MEM_ACCESS_CFG register.
CMD_LOAD_TRIM	Transfer from the MNVR sector to the PATCH_ADDR[7:0] registers and to the flash internal configuration registers. The status bit TRIMMED_STATUS is updated at the end of the command, indicating if there has been an uncorrectable ECC error or a word pair containing all 1s or 0s when reading MNVR, in which case the default trim values are used to guarantee the proper flash read functionality.
CMD_SET_CONF_REG	Copy from FLASH_DATA[1:0] registers to the flash internal configuration registers specified by the FLASH_ADDR register (only used for Production test).
CMD_DPD	This command that does not complete by itself, but requires CMD_END for this. While this command is active, the BUSY bit of the FLASH_IF_STATUS register is set and any CBus access to the flash results in a bus fault.

Table 16. Flash Low-Level Commands (Continued)

Command	Description
CMD_READ	Execute a read access, mainly for test purposes. If ECC is enabled, a pair of 32-bit words is read from the flash. If ECC is disabled, a single 38-bit word is read from the flash.
CMD_PROGRAM_NOS EQ	Execute a non-sequential programming access. This command is only accepted when the sector addressed by FLASH_ADDR is in an unlocked flash zone. A single word is written in the flash. The RETRY field of FLASH_IF_CTRL must be set to FLASH_RETRY_4.
CMD_PROGRAM_SEQ	Initiates a sequential programming access. This command is only accepted when the sector addressed by FLASH_ADDR is in an unlocked flash zone. Up to 128 words can be written in the same row within a Code sector and up to 32 words can be written in the same row within a Data sector. The RETRY field of FLASH_IF_CTRL must be set to FLASH_RETRY_4. This command that does not complete by itself, but requires CMD_END for this. While this command is active, the BUSY bit of the FLASH_IF_STATUS register is set and any CBus access to the flash results in a bus fault.

Table 16. Flash Low-Level Commands (Continued)

Command	Description
CMD_PRE_PROGRAM_NOS EQ	Execute a non-sequential programming access with a pre-programming step. The pre-program is needed if an endurance higher than 10 k is required. This command is only accepted when the sector addressed by FLASH_ADDR is in an unlocked flash zone. A single word is written in the flash. The RETRY field of FLASH_IF_CTRL must be set to FLASH_RETRY_4. This command must be follow by the CMD_ PROGRAM_NOSEQ command with the same address and the same data written with the CMD_ PRE_PROGRAM_NOSEQ command.
CMD_PRE_PROGRAM_SEQ	Initiates a sequential programming access with a pre-programming step. The pre-program is needed if an endurance higher than 10 k is required. This command is only accepted when the sector addressed by FLASH_ADDR is in an unlocked flash zone. Up to 128 words can be written in the same row within a Code sector and up to 32 word can be written in the same row within a Data sector. The RETRY field of the FLASH_IF_CTRL register must be set to FLASH_RETRY_4. This command that does not complete by itself, but requires CMD_END for this. While this command is active, the BUSY bit of the FLASH_IF_STATUS register is set and any CBus access to the flash results in a bus fault. This command must be followed by the CMD_ PROGRAM_SEQ command with the same addresses and the data written with the CMD_ PRE_PROGRAM_SEQ command.

Table 16. Flash Low-Level Commands (Continued)

Command	Description
CMD_END	Terminates a sequential programming access (with or without pre-program) or exits deep power down mode. This command is only accepted when the PROG_SEQ_DATA_REQ bit of the FLASH_IF_STATUS register is set, or while in deep power down mode. Note that it is not an actual flash command, but a write action bit.
CMD_SECTOR_ERASE	This sector erase command is only accepted when the sector addressed by FLASH_ADDR is in an unlocked Flash zone. The RETRY field of FLASH_IF_CTRL determines the erase strength; see Section 9.4 “Flash Memory Registers” on page 483 for more details.
CMD_MASS_ERASE	This mass erase command is only accepted when all MAIN zones are unlocked, and NVR zones are locked. All MAIN sectors (code & data arrays), including the redundancy sectors, are erased except when NVR0 through NVR7 are unlocked. The RETRY field of FLASH_IF_CTRL must be set to FLASH_RETRY_4. The write cycle endurance of the Flash is impacted by performing a mass erase. It is therefore recommended to only use sector erase in the application.
CMD_SET_LOW_POWER	Sets the LPWR Flash pin respecting the hold & setup time. This command is called automatically when the LP_MODE bit in the FLASH_IF_CTRL register is changed from 0 to 1.
CMD_UNSET_LOW_POWER	Unsets the LPWR Flash pin respecting the hold & setup time. This command is called automatically when the LP_MODE bit in theFLASH_IF_CTRL register is changed from 1 to 0.
CMD_SET_RECALL	Sets the RECALL Flash pin respecting the hold & setup time. This command is called automatically when the RECALL bit in theFLASH_IF_CTRL register is changed from 0 to 1.

Table 16. Flash Low-Level Commands (Continued)

Command	Description
CMD_UNSET_RECALL	Unsets the RECALL Flash pin respecting the hold & setup time. This command is called automatically when the RECALL bit in the FLASH_IF_CTRL register is changed from 1 to 0.
CMD_SET_VREAD1	Sets the VREAD1 Flash pin respecting the hold & setup time. This command is called automatically when the VREAD1_MODE bit in the FLASH_IF_CTRL register is changed from 0 to 1
CMD_UNSET_VREAD1	Sets the VREAD1 Flash pin respecting the hold & setup time. This command is called automatically when the VREAD1_MODE bit in the FLASH_IF_CTRL register is changed from 1 to 0.
CMD_WRITE_USER_RED	Writes FLASH_DATA[0] into MNVR. FLASH_DATA[1] indicates the redundancy sector.

9.3.4 Redundancy Sectors

The RSL15 flash provides four redundancy sectors, including:

- Two sectors that can be used in production to patch any code array sectors
- Two sectors that can be used in production to patch any data array sectors

The flash also supports the use of four sectors that can be used as user redundancy sectors, including:

- The last two flash code sectors that can be used by the user application to patch any other code array sectors
- The last two flash data sectors that can be used by the user application to patch any other data array sectors

The addresses of the sectors to be patched are stored in MNVR, and loaded during flash initialization to the FLASH_PATCH_ADDR* registers. If a flash redundancy sector is not used, the FLASH_PATCH_ADDR_PATCH_NOT_VALID bit from its FLASH_PATCH_ADDR* register is set.

Information on the available flash redundancy sectors is provided in the "Flash" table (9.3).

Table 17. Flash Redundancy Sectors

Redundancy Sector	Flash Sector Type	Nominal Address	Patch Register
Code redundancy 1 (RED1)	Code	0x00060000	FLASH_PATCH_ADDR0
Code redundancy 2 (RED2)	Code	0x00060800	FLASH_PATCH_ADDR1
Data redundancy 1 (RED3)	Data	0x00061000	FLASH_PATCH_ADDR2
Data redundancy 2 (RED4)	Data	0x00061100	FLASH_PATCH_ADDR3
User code redundancy (USER_RED1)	Code	0x00157000	FLASH_PATCH_ADDR4
User code redundancy (USER_RED2)	Code	0x00157800	FLASH_PATCH_ADDR5
User data redundancy (USER_RED3)	Data	0x001D7E00	FLASH_PATCH_ADDR6
User data redundancy (USER_RED4)	Data	0x001D7F00	FLASH_PATCH_ADDR7

IMPORTANT: Each of the user redundancy sectors can be configured once for a device, using the `CMD_WRITE_USER_RED` flash command. Before configuring a user redundancy sector, a user application must ensure that the sector is not already in use. When running this command, the address of the sector to be patched must be written to the `FLASH_DATA[0]` register, and the user redundancy index (between 1 and 4) must be written to `FLASH_DATA[1]`, as described in Section 1.0.1 “Flash Memory Operations” on page 1.

NOTE: If multiple flash redundancy sectors are applied to the same address, the following priority is observed (listed from highest to lowest priority): USER_RED2, USER_RED1, RED2, RED1 for flash code array sectors, and USER_RED4, USER_RED3, RED4, RED3 for flash data array sectors.

NOTE: User redundancy sectors can only be patched by production redundancy sectors.

9.3.5 Error-Correction Coding

To prevent possible issues inherent in flash technology, an Error Correcting Code (ECC) ensures the integrity of the flash content, as follows:

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

The algorithm relies on the (38, 32) extended Hamming code, where 32 data bits are extended by 6 parity bits to form a 38-bit word. This is a Single Error Correcting (SEC) code, which allows correction of a single bit error.

The flash ECC bit generation and the error detection and correction are performed on the fly by a dedicated hardware block with no incurred latency on the read/write operation, as shown in the "Flash" figure (9.3).

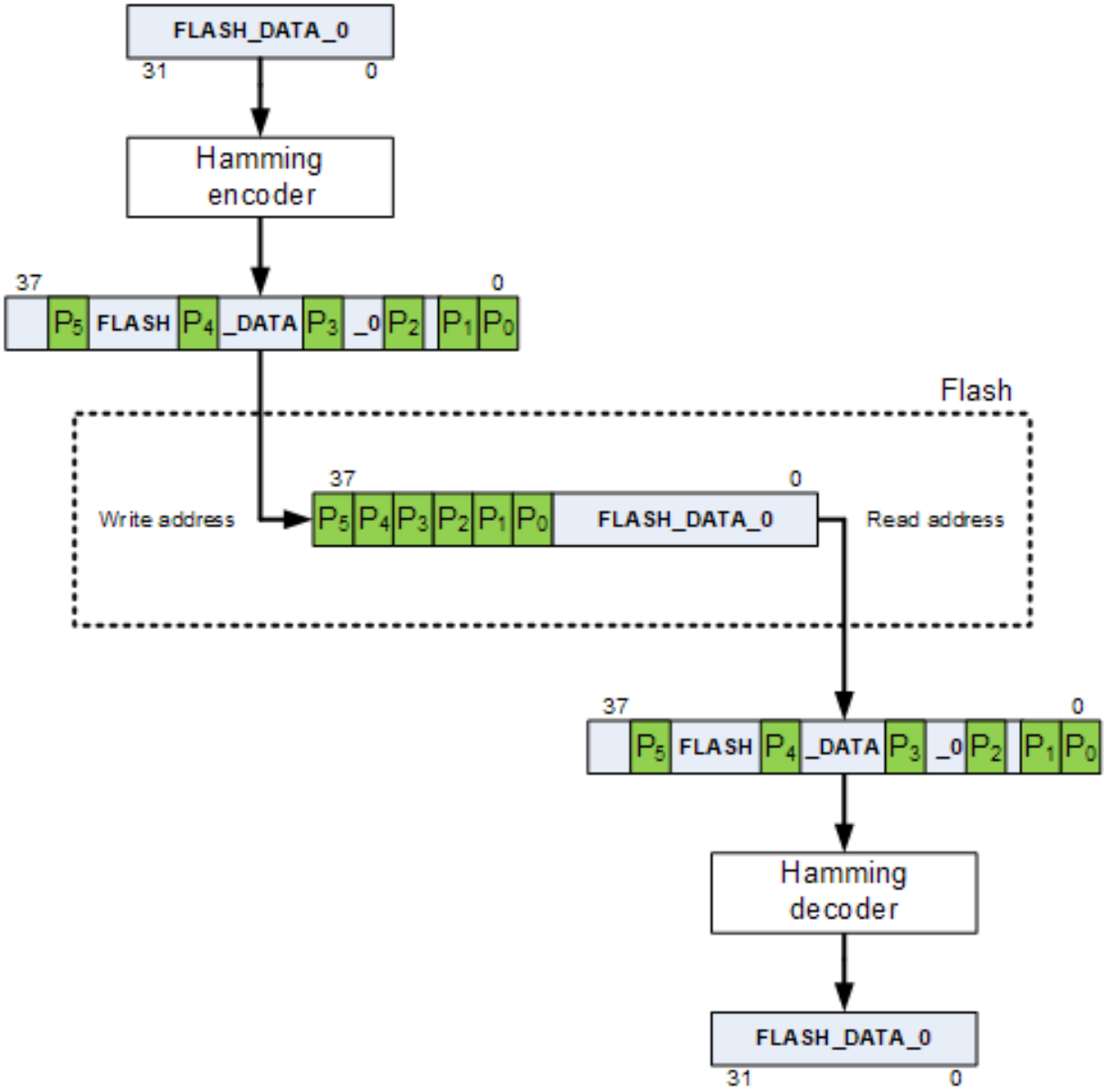


Figure 51. Flash Write and Read Operations Sequence with ECC Enabled

When writing to the flash memory with ECC enabled (the `FLASH_ECC_CTRL_CMD_ECC_CTRL` field in the `FLASH_ECC_CTRL` register), the 6 parity bits are first computed over the 32-bit `FLASH_DATA[0]` registers using the extended Hamming encoding scheme. The 32-bit data from the register is left-extended with the resulting 6 parity bits, and written into the flash as a 38-bit word.

When reading from the flash memory with ECC enabled, the 38-bit word is read from the flash address, and the ECC bits are computed over the 38-bit data read from the flash using the extended Hamming encoding scheme. The computed ECC bits are compared to the six parity bits and are then read from the flash.

No errors

If no difference is observed, the status register FLASH_ECC_STATUS is not updated.

One error

If the bit to be corrected is inside the 32-bit data, one error has been detected. The data is corrected before being used, the FLASH_ECC_STATUS_ECC_COR_ERROR_CNT_STATUS bit in the FLASH_ECC_STATUS register is set, the corrupted flash address is saved in the FLASH_ECC_ERROR_ADDR register, and the value of FLASH_ECC_COR_ERROR_CNT counter is incremented by one. When the counter becomes equal to or greater than FLASH_ECC_CTRL_ECC_COR_CNT_INT_THRESHOLD, a FLASH_ECC_INT interrupt is generated.

More than one error

If the corrected bit is outside the 32-bit data, the detected ECC error cannot be corrected. The FLASH_ECC_STATUS_ECC_UNCOR_ERROR_CNT_STATUS bit in the FLASH_ECC_STATUS register is set, the corrupted flash address is saved in the FLASH_ECC_ERROR_ADDR register, the value of the FLASH_ECC_UNCOR_ERROR_CNT counter is incremented by one, and a FLASH_ECC_INT interrupt is generated.

The ECC status flags and FLASH_ECC_ERROR_ADDR register can be cleared using the FLASH_ECC_STATUS_ECC_COR_ERROR_CNT_CLEAR, FLASH_ECC_STATUS_ECC_UNCOR_ERROR_CNT_CLEAR, and FLASH_ECC_STATUS_ECC_ERROR_ADDR_CLEAR bits from the FLASH_ECC_STATUS register.

The total number of detected errors is reported in the FLASH_ECC_UNCOR_ERROR_CNT register. The counter saturates at 255, and can be reset by setting the FLASH_ECC_STATUS_ECC_UNCOR_ERROR_CNT_CLEAR bit in the FLASH_ECC_STATUS register.

The total number of corrected errors is reported in the FLASH_ECC_COR_ERROR_CNT register. The counter saturates at 255, and can be reset by setting the FLASH_ECC_STATUS_ECC_COR_ERROR_CNT_CLEAR bit in the FLASH_ECC_STATUS register.

9.4 Flash Memory Registers

Register Name	Register Description	Address
FLASH0_IF_CTRL	Flash Interface Control Register	0x40000800
FLASH0_MAIN_WRITE_UNLOCK	Flash Main Write Unlock Register	0x40000804
FLASH0_MAIN_CTRL	Flash Main Write Control Register	0x40000808
FLASH0_DELAY_CTRL	Flash, Memory and RF Power-Up Delay Configuration	0x4000080C
FLASH0_CMD_CTRL	Flash Command Control Register	0x40000830
FLASH0_IF_STATUS	Flash Interface Status Register	0x40000834
FLASH0_ADDR	Flash Address Register	0x40000838
FLASH0_DATA	Flash Read/Write Data Register	0x4000083C-0x40000840
FLASH0_NVR_WRITE_UNLOCK	Flash NVR Write Unlock Register	0x40000844
FLASH0_NVR_CTRL	Flash NVR Control Register	0x40000848

Register Name	Register Description	Address
FLASH0_PATCH_ADDR	Flash Patch Address Register	0x40000864-0x40000880
FLASH0_ECC_CTRL	Flash ECC Control Register	0x400008A8
FLASH0_ECC_STATUS	Flash ECC Status Register	0x400008AC
FLASH0_ECC_ERROR_ADDR	Flash Address of the Latest Detected Error	0x400008B0
FLASH0_ECC_UNCOR_ERROR_CNT	Flash ECC Uncorrected Error Counter	0x400008B4
FLASH0_ECC_COR_ERROR_CNT	Flash ECC Corrected Error Counter	0x400008B8
FLASH0_NVM_STATUS	Flash NVM Status (only available when NVR_FOR_CC312 in ID_NUM register is 1)	0x400008BC
FLASH0_MAIN_MASK	Flash Main Mask	0x400008C0
FLASH0_IF_ID_NUM	Flash Interface ID number	0x400008FC

9.4.0.1 FLASH_IF_CTRL

Bit Field	Read/Write	Field Name	Description
16	RW	NOT_LOAD_AUTO	Do not automatically load the configuration registers and the patch information from MNVR sector after the command WAKEUP is completed.
12	RW	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.
10	RW	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	RW	RETRY	Configures the erase retry iteration. This impacts the Flash endurance time. Also used by Flash programming.
0	RW	LP_MODE	Set the low power mode. Changing this bit will execute the CMD_SET_LOW_POWER or CMD_UNSET_LOW_POWER command.

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
16	NOT_LOAD_AUTO	FLASH_LOAD_AUTO_ENABLE	No automatic load done after the WAKEUP command	0x0*
		FLASH_LOAD_AUTO_DISABLE	The CMD_WAKEUP includes the loading of internal registers and patch information.	0x1
12	VREAD1_MODE	FLASH_VREAD1_DISABLE	After erase, read data with a normal condition	0x0*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		FLASH_VREAD1_ENABLE	After erase, read data with a more stringent condition	0x1
10	RECALL	FLASH_RECALL_DISABLE	RECALL pin low during read command	0x0*
		FLASH_RECALL_ENABLE	RECALL pin high during read command	0x1
9:8	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
0	LP_MODE	FLASH_LOW_POWER_DISABLE	Disable the Flash low power mode	0x0*
		FLASH_LOW_POWER_ENABLE	Enable the Flash low power mode	0x1

9.4.0.2 FLASH_MAIN_WRITE_UNLOCK

Bit Field	Read/Write	Field Name	Description
31:0	W	UNLOCK_KEY	32-bit key to allow for write accesses into the Flash MAIN Block

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:0	UNLOCK_KEY	FLASH_MAIN_KEY	32-bit key to allow for Read and Write accesses into the Flash MAIN Block	0xDBC8264E

9.4.0.3 FLASH_MAIN_CTRL

Bit Field	Read/Write	Field Name	Description
11	RW	DATA_A_35K_TO_40K_W_EN	Authorize the write access to the Flash MAIN Data array from 35K to 40K word block through the FLASH_IF registers.
10	RW	DATA_A_30K_TO_35K_W_EN	Authorize the write access to the Flash MAIN Data array from 30K to 35K word block through the FLASH_IF registers.
9	RW	DATA_A_25K_TO_30K_W_EN	Authorize the write access to the Flash MAIN Data array from 25K to 30K word block through the FLASH_IF registers.
8	RW	DATA_A_20K_TO_25K_W_EN	Authorize the write access to the Flash MAIN Data array from 20K to 25K word block through the FLASH_IF registers.
7	RW	DATA_A_15K_TO_20K_W_EN	Authorize the write access to the Flash MAIN Data array from 15K to 20K word block through the FLASH_IF registers.
6	RW	DATA_A_10K_TO_15K_W_EN	Authorize the write access to the Flash MAIN Data array from 10K to 15K word block through the FLASH_IF registers.

Bit Field	Read/Write	Field Name	Description
5	RW	DATA_A_5K_TO_10K_W_EN	Authorize the write access to the Flash MAIN Data array from 5K to 10K word block through the FLASH_IF registers.
4	RW	DATA_A_0K_TO_5K_W_EN	Authorize the write access to the Flash MAIN Data array from 0 to 5K word block through the FLASH_IF registers.
3	RW	CODE_A_66K_TO_88K_W_EN	Authorize the write access to the Flash MAIN Code array from 66K to 88K word block through the FLASH_IF registers.
2	RW	CODE_A_44K_TO_66K_W_EN	Authorize the write access to the Flash MAIN Code array from 44K to 66K word block through the FLASH_IF registers.
1	RW	CODE_A_22K_TO_44K_W_EN	Authorize the write access to the Flash MAIN Code array from 22K to 44K word block through the FLASH_IF registers.
0	RW	CODE_A_0K_TO_22K_W_EN	Authorize the write access to the Flash MAIN Code array from 0 to 22K word block through the FLASH_IF registers.

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
11	DATA_A_35K_TO_40K_W_EN	DATA_A_35K_TO_40K_W_DISABLE	The part K to K of the Flash MAIN Data array is protected against write access	0x0*
		DATA_A_35K_TO_40K_W_ENABLE	The part K to K of the Flash MAIN Data array can be written	0x1
10	DATA_A_30K_TO_35K_W_EN	DATA_A_30K_TO_35K_W_DISABLE	The part K to K of the Flash MAIN Data array is protected against write access	0x0*
		DATA_A_30K_TO_35K_W_ENABLE	The part K to K of the Flash MAIN Data array can be written	0x1
9	DATA_A_25K_TO_30K_W_EN	DATA_A_25K_TO_30K_W_DISABLE	The part K to K of the Flash MAIN Data array is protected against write access	0x0*
		DATA_A_25K_TO_30K_W_ENABLE	The part K to K of the Flash MAIN Data array can be written	0x1
8	DATA_A_20K_TO_25K_W_EN	DATA_A_20K_TO_25K_W_DISABLE	The part K to K of the Flash MAIN Data array is protected against write access	0x0*
		DATA_A_20K_TO_25K_W_ENABLE	The part K to K of the Flash MAIN Data array can be written	0x1
7	DATA_A_15K_TO_20K_W_EN	DATA_A_15K_TO_20K_W_DISABLE	The part K to K of the Flash MAIN Data array is protected against write access	0x0*
		DATA_A_15K_TO_20K_W_ENABLE	The part K to K of the Flash MAIN Data array can be written	0x1
6	DATA_A_10K_TO_15K_W_EN	DATA_A_10K_TO_15K_W_DISABLE	The part K to K of the Flash MAIN Data array is protected against write access	0x0*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		DATA_A_10K_TO_15K_W_ENABLE	The part K to K of the Flash MAIN Data array can be written	0x1
5	DATA_A_5K_TO_10K_W_EN	DATA_A_5K_TO_10K_W_DISABLE	The part K to K of the Flash MAIN Data array is protected against write access	0x0*
		DATA_A_5K_TO_10K_W_ENABLE	The part K to K of the Flash MAIN Data array can be written	0x1
4	DATA_A_0K_TO_5K_W_EN	DATA_A_0K_TO_5K_W_DISABLE	The part K to K of the Flash MAIN Data array is protected against write access	0x0*
		DATA_A_0K_TO_5K_W_ENABLE	The part K to K of the Flash MAIN Data array can be written	0x1
3	CODE_A_66K_TO_88K_W_EN	CODE_A_66K_TO_88K_W_DISABLE	The part K to K of the Flash MAIN Code array is protected against write access	0x0*
		CODE_A_66K_TO_88K_W_ENABLE	The part K to K of the Flash MAIN Code array can be written	0x1
2	CODE_A_44K_TO_66K_W_EN	CODE_A_44K_TO_66K_W_DISABLE	The part K to K of the Flash MAIN Code array is protected against write access	0x0*
		CODE_A_44K_TO_66K_W_ENABLE	The part K to K of the Flash MAIN Code array can be written	0x1
1	CODE_A_22K_TO_44K_W_EN	CODE_A_22K_TO_44K_W_DISABLE	The part K to K of the Flash MAIN Code array is protected against write access	0x0*
		CODE_A_22K_TO_44K_W_ENABLE	The part K to K of the Flash MAIN Code array can be written	0x1
0	CODE_A_0K_TO_22K_W_EN	CODE_A_0K_TO_22K_W_DISABLE	The part K to K of the Flash MAIN Code array is protected against write access	0x0*
		CODE_A_0K_TO_22K_W_ENABLE	The part K to K of the Flash MAIN Code array can be written	0x1

9.4.0.4 FLASH_DELAY_CTRL

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

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
7	READ_MARGIN	DEFAULT_READ_MARGIN	Use default read margins	0x0*
		FAST_READ_MARGIN	Use fast read margins	0x1
3:0	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

9.4.0.5 FLASH_CMD_CTRL

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

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
5	CMD_END	CMD_END	Terminates an active Flash command if possible	0x1
4:0	COMMAND	CMD_IDLE	Idle command	0x0*
		CMD_WAKE_UP	Wake up the Flash	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		CMD_LOAD_TRIM	Load patch and trimming values from MNVR	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_WRITE_USER_RED	Write FLASH_DATA to PATCH4 of M NVR, FLASH_DATA[1] indicates which redundancy sector, FLASH_DATA[0] the data to write	0x10
		CMD_PRE_PROGRAM_NOSEQ	Execute a non-sequential pre-programming cycle	0x11
		CMD_PRE_PROGRAM_SEQ	Starts a sequential pre-programming sequence	0x12

9.4.0.6 FLASH_IF_STATUS

Bit Field	Read/Write	Field Name	Description
31	R	TRIMMED_STATUS	Flash trimming status
30	R	ISOLATE_STATUS	Flash isolate status
29	R	PROG_SEQ_DATA_REQ	Request new data while in sequential program mode
28	R	BUSY	Flash interface busy status bit
27	R	DATA_RED2_W_UNLOCK	Flash Data array RED2 write unlock status bit
26	R	DATA_RED1_W_UNLOCK	Flash Data array RED1 write unlock status bit
25	R	CODE_RED2_W_UNLOCK	Flash Code array RED2 write unlock status bit
24	R	CODE_RED1_W_UNLOCK	Flash Code array RED1 write unlock status bit

Bit Field	Read/Write	Field Name	Description
22	R	NVR7_W_UNLOCK	Flash NVR7 write unlock status bit
21	R	NVR6_W_UNLOCK	Flash NVR6 write unlock status bit
20	R	NVR5_W_UNLOCK	Flash NVR5 write unlock status bit
19	R	NVR4_W_UNLOCK	Flash NVR4 write unlock status bit
18	R	NVR3_W_UNLOCK	Flash NVR3 write unlock status bit
17	R	NVR2_W_UNLOCK	Flash NVR2 write unlock status bit
16	R	NVR1_W_UNLOCK	Flash NVR1 write unlock status bit
15	R	NVR0_W_UNLOCK	Flash NVR0 write unlock status bit
11	R	DATA_A_35K_TO_40K_W_UNLOCK	Write unlock status bit of the part 35K to 40K of the Flash MAIN Data array
10	R	DATA_A_30K_TO_35K_W_UNLOCK	Write unlock status bit of the part 30K to 35K of the Flash MAIN Data array
9	R	DATA_A_25K_TO_30K_W_UNLOCK	Write unlock status bit of the part 25K to 30K of the Flash MAIN Data array
8	R	DATA_A_20K_TO_25K_W_UNLOCK	Write unlock status bit of the part 20K to 25K of the Flash MAIN Data array
7	R	DATA_A_15K_TO_20K_W_UNLOCK	Write unlock status bit of the part 15K to 20K of the Flash MAIN Data array
6	R	DATA_A_10K_TO_15K_W_UNLOCK	Write unlock status bit of the part 10K to 15K of the Flash MAIN Data array
5	R	DATA_A_5K_TO_10K_W_UNLOCK	Write unlock status bit of the part 5K to 10K of the Flash MAIN Data array
4	R	DATA_A_0K_TO_5K_W_UNLOCK	Write unlock status bit of the part 0K to 5K of the Flash MAIN Data array
3	R	CODE_A_66K_TO_88K_W_UNLOCK	Write unlock status bit of the part 66K to 88K of the Flash MAIN Code array
2	R	CODE_A_44K_TO_66K_W_UNLOCK	Write unlock status bit of the part 44K to 66K of the Flash MAIN Code array
1	R	CODE_A_22K_TO_44K_W_UNLOCK	Write unlock status bit of the part 22K to 44K of the Flash MAIN Code array
0	R	CODE_A_0K_TO_22K_W_UNLOCK	Write unlock status bit of the part 0K to 22K of the Flash MAIN Code array

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31	TRIMMED_STATUS	FLASH_UNTRIMMED	All MNVR CBD0-CDB7 contents are equal to 0xFFFF. Flash untrimmed.	0x0*
		FLASH_TRIMMED	Some registers CBD0-CBD7 contents are not equal to 0xFFFF. Flash trimmed.	0x1
30	ISOLATE_STATUS	FLASH_ACCESSIBLE	Flash can be accessed (isolation inactive)	0x0
		FLASH_ISOLATE	Flash cannot be accessed (isolation active)	0x1*
29	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
28	BUSY	FLASH_IF_IDLE	Indicates that the Flash interface is ready.	0x0*
		FLASH_IF_BUSY	Indicates that the Flash interface is busy.	0x1
27	DATA_RED2_W_UNLOCK	FLASH_DATA_RED2_W_LOCKED	Indicates that the Flash Data array RED2 sector is protected against write accesses by the Flash interface	0x0*
		FLASH_DATA_RED2_W_UNLOCKED	Indicates that the Flash Data array RED2 sector can be write accessed by the Flash interface	0x1
26	DATA_RED1_W_UNLOCK	FLASH_DATA_RED1_W_LOCKED	Indicates that the Flash Data array RED1 sector is protected against write accesses by the Flash interface	0x0*
		FLASH_DATA_RED1_W_UNLOCKED	Indicates that the Flash Data array RED1 sector can be write accessed by the Flash interface	0x1
25	CODE_RED2_W_UNLOCK	FLASH_CODE_RED2_W_LOCKED	Indicates that the Flash Code array RED2 sector is protected against write accesses by the Flash interface	0x0*
		FLASH_CODE_RED2_W_UNLOCKED	Indicates that the Flash Code array RED2 sector can be write accessed by the Flash interface	0x1
24	CODE_RED1_W_UNLOCK	FLASH_CODE_RED1_W_LOCKED	Indicates that the Flash Code array RED1 sector is protected against write accesses by the Flash interface	0x0*
		FLASH_CODE_RED1_W_UNLOCKED	Indicates that the Flash Code array RED1 sector can be write accessed by the Flash interface	0x1
22	NVR7_W_UNLOCK	FLASH_NVR7_W_LOCKED	Indicates that the Flash NVR7 sector is	0x0*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
			protected against write accesses by the Flash interface	
		FLASH_NVR7_W_UNLOCKED	Indicates that the Flash NVR7 sector can be write accessed by the Flash interface	0x1
21	NVR6_W_UNLOCK	FLASH_NVR6_W_LOCKED	Indicates that the Flash NVR6 sector is protected against write accesses by the Flash interface	0x0*
		FLASH_NVR6_W_UNLOCKED	Indicates that the Flash NVR6 sector can be write accessed by the Flash interface	0x1
20	NVR5_W_UNLOCK	FLASH_NVR5_W_LOCKED	Indicates that the Flash NVR5 sector is protected against write accesses by the Flash interface	0x0*
		FLASH_NVR5_W_UNLOCKED	Indicates that the Flash NVR5 sector can be write accessed by the Flash interface	0x1
19	NVR4_W_UNLOCK	FLASH_NVR4_W_LOCKED	Indicates that the Flash NVR4 sector is protected against write accesses by the Flash interface	0x0*
		FLASH_NVR4_W_UNLOCKED	Indicates that the Flash NVR4 sector can be write accessed by the Flash interface	0x1
18	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
17	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
16	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
15	NVR0_W_UNLOCK	FLASH_NVR0_W_LOCKED	Indicates that the Flash NVR0 sector is protected against write accesses by the Flash interface	0x0*
		FLASH_NVR0_W_UNLOCKED	Indicates that the Flash NVR0 sector can	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
			be write accessed by the Flash interface	
11	DATA_A_35K_TO_40K_W_UNLOCK	DATA_A_35K_TO_40K_W_LOCKED	Indicates that the part 35K to 40K of the Flash MAIN Data array is protected against write accesses by the Flash interface	0x0*
		DATA_A_35K_TO_40K_W_UNLOCKED	Indicates that the part 35K to 40K of the Flash MAIN Data array can be write accessed by the Flash interface	0x1
10	DATA_A_30K_TO_35K_W_UNLOCK	DATA_A_30K_TO_35K_W_LOCKED	Indicates that the part 30K to 35K of the Flash MAIN Data array is protected against write accesses by the Flash interface	0x0*
		DATA_A_30K_TO_35K_W_UNLOCKED	Indicates that the part 30K to 35K of the Flash MAIN Data array can be write accessed by the Flash interface	0x1
9	DATA_A_25K_TO_30K_W_UNLOCK	DATA_A_25K_TO_30K_W_LOCKED	Indicates that the part 25K to 30K of the Flash MAIN Data array is protected against write accesses by the Flash interface	0x0*
		DATA_A_25K_TO_30K_W_UNLOCKED	Indicates that the part 25K to 30K of the Flash MAIN Data array can be write accessed by the Flash interface	0x1
8	DATA_A_20K_TO_25K_W_UNLOCK	DATA_A_20K_TO_25K_W_LOCKED	Indicates that the part 20K to 25K of the Flash MAIN Data array is protected against write accesses by the Flash interface	0x0*
		DATA_A_20K_TO_25K_W_UNLOCKED	Indicates that the part 20K to 25K of the Flash MAIN Data array can be write accessed by the Flash interface	0x1
7	DATA_A_15K_TO_20K_W_UNLOCK	DATA_A_15K_TO_20K_W_LOCKED	Indicates that the part 15K to 20K of the Flash MAIN Data array is protected against write accesses by the Flash interface	0x0*
		DATA_A_15K_TO_20K_W_UNLOCKED	Indicates that the part 15K to 20K of the Flash MAIN Data array can be write accessed by the Flash interface	0x1
6	DATA_A_10K_TO_15K_W_UNLOCK	DATA_A_10K_TO_15K_W_LOCKED	Indicates that the part 10K to 15K of the Flash MAIN Data array is protected against write accesses by the Flash interface	0x0*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		DATA_A_10K_TO_15K_W_UNLOCKED	Indicates that the part 10K to 15K of the Flash MAIN Data array can be write accessed by the Flash interface	0x1
5	DATA_A_5K_TO_10K_W_UNLOCK	DATA_A_5K_TO_10K_W_LOCKED	Indicates that the part 5K to 10K of the Flash MAIN Data array is protected against write accesses by the Flash interface	0x0*
		DATA_A_5K_TO_10K_W_UNLOCKED	Indicates that the part 5K to 10K of the Flash MAIN Data array can be write accessed by the Flash interface	0x1
4	DATA_A_0K_TO_5K_W_UNLOCK	DATA_A_0K_TO_5K_W_LOCKED	Indicates that the part 0K to 5K of the Flash MAIN Data array is protected against write accesses by the Flash interface	0x0*
		DATA_A_0K_TO_5K_W_UNLOCKED	Indicates that the part 0K to 5K of the Flash MAIN Data array can be write accessed by the Flash interface	0x1
3	CODE_A_66K_TO_88K_W_UNLOCK	CODE_A_66K_TO_88K_W_LOCKED	Indicates that the part 66K to 88K of the Flash MAIN Code array is protected against write accesses by the Flash interface	0x0*
		CODE_A_66K_TO_88K_W_UNLOCKED	Indicates that the part 66K to 88K of the Flash MAIN Code array can be write accessed by the Flash interface	0x1
2	CODE_A_44K_TO_66K_W_UNLOCK	CODE_A_44K_TO_66K_W_LOCKED	Indicates that the part 44K to 66K of the Flash MAIN Code array is protected against write accesses by the Flash interface	0x0*
		CODE_A_44K_TO_66K_W_UNLOCKED	Indicates that the part 44K to 66K of the Flash MAIN Code array can be write accessed by the Flash interface	0x1
1	CODE_A_22K_TO_44K_W_UNLOCK	CODE_A_22K_TO_44K_W_LOCKED	Indicates that the part 22K to 44K of the Flash MAIN Code array is protected against write accesses by the Flash interface	0x0*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		CODE_A_22K_TO_44K_W_UNLOCKED	Indicates that the part 22K to 44K of the Flash MAIN Code array can be write accessed by the Flash interface	0x1
0	CODE_A_0K_TO_22K_W_UNLOCK	CODE_A_0K_TO_22K_W_LOCKED	Indicates that the part 0K to 22K of the Flash MAIN Code array is protected against write accesses by the Flash interface	0x0*
		CODE_A_0K_TO_22K_W_UNLOCKED	Indicates that the part 0K to 22K of the Flash MAIN Code array can be write accessed by the Flash interface	0x1

9.4.0.7 FLASH_ADDR

Bit Field	Read/Write	Field Name	Description
20:2	RW	FLASH_ADDR	Flash Byte Address

9.4.0.8 FLASH_DATA

Bit Field	Read/Write	Field Name	Description
31:0	RW	DATA	32-bit Flash Data

9.4.0.9 FLASH_NVR_WRITE_UNLOCK

Bit Field	Read/Write	Field Name	Description
31:0	W	UNLOCK_KEY	32-bit key to allow for write access to NVR sectors of the Flash

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:0	UNLOCK_KEY	FLASH_NVR_KEY	32-bit key to allow for write access to the Flash NVR sector	0x71B371F5

9.4.0.10 FLASH_NVR_CTRL

Bit Field	Read/Write	Field Name	Description
7	RW	NVR7_W_EN	Authorize write access to the Flash NVR7 sector through the FLASH_IF registers.
6	RW	NVR6_W_EN	Authorize write access to the Flash NVR6 sector through the FLASH_IF registers.
5	RW	NVR5_W_EN	Authorize write access to the Flash NVR5 sector through the FLASH_IF registers.

Bit Field	Read/Write	Field Name	Description
4	RW	NVR4_W_EN	Authorize write access to the Flash NVR4 sector through the FLASH_IF registers.
3	RW	NVR3_W_EN	Authorize write access to the Flash NVR3 sector through the FLASH_IF registers.
2	RW	NVR2_W_EN	Authorize write access to the Flash NVR2 sector through the FLASH_IF registers.
1	RW	NVR1_W_EN	Authorize write access to the Flash NVR1 sector through the FLASH_IF registers.
0	RW	NVR0_W_EN	Authorize write access to the Flash NVR0 sector through the FLASH_IF registers.

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
7	NVR7_W_EN	NVR7_WRITE_DISABLE	The Flash NVR7 block is protected against write access.	0x0*
		NVR7_WRITE_ENABLE	The Flash NVR7 block can be written.	0x1
6	NVR6_W_EN	NVR6_WRITE_DISABLE	The Flash NVR6 block is protected against write access.	0x0*
		NVR6_WRITE_ENABLE	The Flash NVR6 block can be written.	0x1
5	NVR5_W_EN	NVR5_WRITE_DISABLE	The Flash NVR5 block is protected against write access.	0x0*
		NVR5_WRITE_ENABLE	The Flash NVR5 block can be written.	0x1
4	NVR4_W_EN	NVR4_WRITE_DISABLE	The Flash NVR4 block is protected against write access.	0x0*
		NVR4_WRITE_ENABLE	The Flash NVR4 block can be written.	0x1
3	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
2	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
1	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
0	NVR0_W_EN	NVR0_WRITE_DISABLE	The Flash NVR0 block is protected against write access.	0x0*
		NVR0_WRITE_ENABLE	The Flash NVR0 block can be written.	0x1

9.4.0.11 FLASH_PATCH_ADDR

Bit Field	Read/Write	Field Name	Description
31	RW	PATCH_NOT_VALID	Indicates if the patch address is valid
20:8	RW	PATCH_ADDR	

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31	PATCH_NOT_VALID	PATCH_VALID	This bit must be cleared in order for the patching to be taken into consideration.	0x0
		PATCH_NOT_VALID		0x1*

9.4.0.12 FLASH_COPY_CFG

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

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
18	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
17	COMP_ADDR_DIR	COMP_ADDR_DOWN	FLASH_COPIER address count-down	0x0
		COMP_ADDR_UP	FLASH_COPIER address count-up	0x1*
16	COMP_MODE	COMP_MODE_CONSTANT	FLASH_DATA[1:0] compare with Flash DOUT	0x0*
		COMP_MODE_CHBK	Odd address compare with FLASH_ DATA[1:0], even address compare with inverse FLASH_DATA[1:0]	0x1
9	COPY_DEST	COPY_TO_MEM	Copy Flash to memory	0x0*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		COPY_TO_CRC	Copy Flash to CRC	0x1
8	COPY_MODE	COPY_TO_32BIT	Copy Flash to 32-bit memory	0x0*
1	PRIORITY	FLASH_CM33_PRIORITY	CM33 has priority over Flash Copier	0x0*
		FLASH_COPY_PRIORITY	Flash Copier has priority over CM33	0x1
0	MODE	COPY_MODE	Flash copier mode	0x0*
		COMPARATOR_MODE	Flash comparator mode	0x1

9.4.0.13 FLASH_COPY_CTRL

Bit Field	Read/Write	Field Name	Description
3	R	ERROR	Error status
2	W	STOP	Stop the transfer
1	W	START	Start the transfer
0	R	BUSY	Busy status

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

9.4.0.14 FLASH_COPY_SRC_ADDR_PTR

Bit Field	Read/Write	Field Name	Description
20:0	RW	COPY_SRC_ADDR_PTR	Source address pointer

9.4.0.15 FLASH_COPY_DST_ADDR_PTR

Bit Field	Read/Write	Field Name	Description
31:2	RW	COPY_DST_ADDR_PTR	Destination address pointer

9.4.0.16 FLASH_COPY_WORD_CNT

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

9.4.0.17 FLASH_ECC_CTRL

Bit Field	Read/Write	Field Name	Description
15:8	RW	ECC_COR_CNT_INT_ THRESHOLD	Select the number of corrected errors before sending a CM33 interrupt
3	RW	COPIER_ECC_CTRL	
2	RW	CMD_ECC_CTRL	
0	RW	CBUS_ECC_CTRL	Select the operating mode of the Flash ECC

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
15:8	ECC_COR_CNT_INT_ THRESHOLD	FLASH_ECC_COR_INT_ THRESHOLD_DISABLED	Interrupt is disabled	0x0
		FLASH_ECC_COR_INT_ THRESHOLD_1	Send a CM33 interrupt when one or more correctable errors are detected.	0x1*
		FLASH_ECC_COR_INT_ THRESHOLD_255	Send a CM33 interrupt when 255 or more correctable errors are detected.	0xFF
3	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*
2	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*
0	CBUS_ECC_CTRL	FLASH_CBUS_ECC_DISABLE	Disables ECC when reading Flash through CBus	0x0
		FLASH_CBUS_ECC_ENABLE	Enables ECC when reading Flash through CBus	0x1*

9.4.0.18 FLASH_ECC_STATUS

Bit Field	Read/Write	Field Name	Description
6	W	ECC_COR_ERROR_CNT_CLEAR	Reset the Flash corrected errors counter
5	W	ECC_UNCOR_ERROR_CNT_CLEAR	Reset the Flash uncorrected errors counter
4	W	ECC_ERROR_ADDR_CLEAR	Reset the Flash address of the last detected error
1	R	ECC_COR_ERROR_CNT_STATUS	FLASH_ECC_ERROR_COR_CNT status
0	R	ECC_UNCOR_ERROR_CNT_STATUS	FLASH_ECC_ERROR_UNCOR_CNT status

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
6	ECC_COR_ERROR_CNT_CLEAR	FLASH_ECC_COR_ERROR_CNT_CLEAR	Reset the Flash corrected errors counter	0x1
5	ECC_UNCOR_ERROR_CNT_CLEAR	FLASH_ECC_UNCOR_ERROR_CNT_CLEAR	Reset the Flash uncorrected errors counter	0x1
4	ECC_ERROR_ADDR_CLEAR	FLASH_ECC_ERROR_ADDR_CLEAR	Reset the Flash address of the latest detected error	0x1
1	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
0	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

9.4.0.19 FLASH_ECC_ERROR_ADDR

Bit Field	Read/Write	Field Name	Description
20:2	R	ECC_ERROR_ADDR	Store the Flash address of the latest Flash ECC error

9.4.0.20 FLASH_ECC_UNCOR_ERROR_CNT

Bit Field	Read/Write	Field Name	Description
7:0	R	ECC_UNCOR_ERROR_CNT	Flash ECC uncorrected error counter

9.4.0.21 FLASH_ECC_COR_ERROR_CNT

Bit Field	Read/Write	Field Name	Description
7:0	R	ECC_COR_ERROR_CNT	Flash ECC corrected error counter

9.4.0.22 FLASH_NVM_STATUS

Bit Field	Read/Write	Field Name	Description
16	W	CLEAR_NVM_STATUS	Clear all the NVM status bits
8	R	NVM_BIT_FAILURE	Indicates if a bit has failed in an address from the CC-NVM layout used by the CryptoCell
5:0	R	FAILED_NVM_ADDRESS	Last failing word address in CC-NVM layout (0x00 to 0x3F)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
16	CLEAR_NVM_STATUS	CLEAR_NVM_STATUS	Clear all the NVM status bits	0x1
8	NVM_BIT_FAILURE	NVM_OK	No bit failure detection in the CC-NVM layout	0x0*
		NVM_BIT_FAILED	At least a bit has failed in the CC-NVM layout	0x1

9.4.0.23 FLASH_MAIN_MASK

Bit Field	Read/Write	Field Name	Description
11	R	DATA_A_35K_TO_40K_W_MASK	Authorize the access to the Flash MAIN Data array from 35K to 40K word block through the FLASH_IF registers.
10	R	DATA_A_30K_TO_35K_W_MASK	Authorize the access to the Flash MAIN Data array from 30K to 35K word block through the FLASH_IF registers.
9	R	DATA_A_25K_TO_30K_W_MASK	Authorize the access to the Flash MAIN Data array from 25K to 30K word block through the FLASH_IF registers.
8	R	DATA_A_20K_TO_25K_W_MASK	Authorize the access to the Flash MAIN Data array from 20K to 25K word block through the FLASH_IF registers.
7	R	DATA_A_15K_TO_20K_W_MASK	Authorize the access to the Flash MAIN Data array from 15K to 20K word block through the FLASH_IF registers.
6	R	DATA_A_10K_TO_15K_W_MASK	Authorize the access to the Flash MAIN Data array from 10K to 15K word block through the FLASH_IF registers.
5	R	DATA_A_5K_TO_10K_W_MASK	Authorize the access to the Flash MAIN Data array from 5K to 10K word block through the FLASH_IF registers.
4	R	DATA_A_0K_TO_5K_W_MASK	Authorize the access to the Flash MAIN Data array from 0 to 5K word block through the FLASH_IF registers.

Bit Field	Read/Write	Field Name	Description
3	R	CODE_A_66K_TO_88K_W_MASK	Authorize the access to the Flash MAIN Code array from 66K to 88K word block through the FLASH_IF registers.
2	R	CODE_A_44K_TO_66K_W_MASK	Authorize the access to the Flash MAIN Code array from 44K to 66K word block through the FLASH_IF registers.
1	R	CODE_A_22K_TO_44K_W_MASK	Authorize the access to the Flash MAIN Code array from 22K to 44K word block through the FLASH_IF registers.
0	R	CODE_A_0K_TO_22K_W_MASK	Authorize the access to the Flash MAIN Code array from 0 to 22K word block through the FLASH_IF registers.

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
11	DATA_A_35K_TO_40K_W_MASK	DATA_A_35K_TO_40K_W_NOT_ACCESSIBLE	The part K to K of the Flash MAIN Data array is not accessible.	0x0
		DATA_A_35K_TO_40K_W_ACCESSIBLE	The part K to K of the Flash MAIN Data array is accessible.	0x1*
10	DATA_A_30K_TO_35K_W_MASK	DATA_A_30K_TO_35K_W_NOT_ACCESSIBLE	The part K to K of the Flash MAIN Data array is not accessible.	0x0
		DATA_A_30K_TO_35K_W_ACCESSIBLE	The part K to K of the Flash MAIN Data array is accessible.	0x1*
9	DATA_A_25K_TO_30K_W_MASK	DATA_A_25K_TO_30K_W_NOT_ACCESSIBLE	The part K to K of the Flash MAIN Data array is not accessible.	0x0
		DATA_A_25K_TO_30K_W_ACCESSIBLE	The part K to K of the Flash MAIN Data array is accessible.	0x1*
8	DATA_A_20K_TO_25K_W_MASK	DATA_A_20K_TO_25K_W_NOT_ACCESSIBLE	The part K to K of the Flash MAIN Data array is not accessible.	0x0
		DATA_A_20K_TO_25K_W_ACCESSIBLE	The part K to K of the Flash MAIN Data array is accessible.	0x1*
7	DATA_A_15K_TO_20K_W_MASK	DATA_A_15K_TO_20K_W_NOT_ACCESSIBLE	The part K to K of the Flash MAIN Data array is not accessible.	0x0
		DATA_A_15K_TO_20K_W_ACCESSIBLE	The part K to K of the Flash MAIN Data array is accessible.	0x1*
6	DATA_A_10K_TO_15K_W_MASK	DATA_A_10K_TO_15K_W_NOT_ACCESSIBLE	The part K to K of the Flash MAIN Data array is not accessible.	0x0
		DATA_A_10K_TO_15K_W_ACCESSIBLE	The part K to K of the Flash MAIN Data array is accessible.	0x1*
5	DATA_A_5K_TO_10K_W_MASK	DATA_A_5K_TO_10K_W_NOT_ACCESSIBLE	The part K to K of the Flash MAIN Data array is not accessible.	0x0

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		DATA_A_5K_TO_10K_W_ACCESSIBLE	The part K to K of the Flash MAIN Data array is accessible.	0x1*
4	DATA_A_0K_TO_5K_W_MASK	DATA_A_0K_TO_5K_W_NOT_ACCESSIBLE	The part K to K of the Flash MAIN Data array is not accessible.	0x0
		DATA_A_0K_TO_5K_W_ACCESSIBLE	The part K to K of the Flash MAIN Data array is accessible.	0x1*
3	CODE_A_66K_TO_88K_W_MASK	CODE_A_66K_TO_88K_W_NOT_ACCESSIBLE	The part K to K of the Flash MAIN Code array is not accessible.	0x0
		CODE_A_66K_TO_88K_W_ACCESSIBLE	The part K to K of the Flash MAIN Code array is accessible.	0x1*
2	CODE_A_44K_TO_66K_W_MASK	CODE_A_44K_TO_66K_W_NOT_ACCESSIBLE	The part K to K of the Flash MAIN Code array is not accessible.	0x0
		CODE_A_44K_TO_66K_W_ACCESSIBLE	The part K to K of the Flash MAIN Code array is accessible.	0x1*
1	CODE_A_22K_TO_44K_W_MASK	CODE_A_22K_TO_44K_W_NOT_ACCESSIBLE	The part K to K of the Flash MAIN Code array is not accessible.	0x0
		CODE_A_22K_TO_44K_W_ACCESSIBLE	The part K to K of the Flash MAIN Code array is accessible.	0x1*
0	CODE_A_0K_TO_22K_W_MASK	CODE_A_0K_TO_22K_W_NOT_ACCESSIBLE	The part K to K of the Flash MAIN Code array is not accessible.	0x0
		CODE_A_0K_TO_22K_W_ACCESSIBLE	The part K to K of the Flash MAIN Code array is accessible.	0x1*

9.4.0.24 FLASH_IF_ID_NUM

Bit Field	Read/Write	Field Name	Description
20	R	FLASH_IF_NVR_FOR_CC312	NVR sectors 0 to 3 are used for the CryptoCell
19:16	R	FLASH_IF_NUMBER	FLASH_IF Instance number
15:8	R	FLASH_IF_MAJOR_REVISION	FLASH_IF Major Revision number
7:0	R	FLASH_IF_MINOR_REVISION	FLASH_IF Minor Revision number

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
20	FLASH_IF_NVR_FOR_CC312	NVR_SECTORS_AVAILABLE	NVR sectors 0 to 3 are available to the user	0x0*
		NVR_SECTORS_USED_BY_CRYPTOCELL	NVR sectors 0 to 3 are used by the CryptoCell and are not available to the user	0x1
15:8	FLASH_IF_MAJOR_REVISION	FLASH_IF_MAJOR_REVISION	Flash interface revision 1.0	0x1*
7:0	FLASH_IF_MINOR_REVISION	FLASH_IF_MINOR_REVISION	Flash interface revision 1.0	0x0*

9.5 Memory Registers

Register Name	Register Description	Address
SYSCTRL_CM33_LOOP_CACHE_CFG	CM33 Loop Cache Configuration	0x40000000
SYSCTRL_ACCESS_ERROR	Memory and Peripheral Access Error Flags	0x40000004
SYSCTRL_MEM_POWER_STARTUP	Memory Power Configuration	0x40000008
SYSCTRL_MEM_POWER_ENABLE	Memory Power Configuration	0x4000000C
SYSCTRL_MEM_ACCESS_CFG	Memory Access Configuration and wake-up Restore Address in packed 4-bit format	0x40000018
SYSCTRL_WAKEUP_ADDR	Wake-up Restore Address in Unpacked 32-bit Format	0x4000001C
SYSCTRL_MEM_RETENTION_CFG	Memory Retention Configuration	0x40000020
SYSCTRL_MEM_TIMING_CFG	Memory Timing Configuration	0x40000024

9.5.0.1 SYSCTRL_CM33_LOOP_CACHE_CFG

Bit Field	Read/Write	Field Name	Description
0	RW	CM33_LOOP_CACHE_ENABLE	CM33 loop cache enable

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
0	CM33_LOOP_CACHE_ENABLE	CM33_LOOP_CACHE_DISABLE	CM33 loop cache disabled	0x0*
		CM33_LOOP_CACHE_ENABLE	CM33 loop cache enabled	0x1

9.5.0.2 SYSCTRL_ACCESS_ERROR

Bit Field	Read/Write	Field Name	Description
16	W	ACCESS_ERROR_CLEAR	Write a 1 to clear the access error flags
12	R	CC312_MEM_ERROR	CC312 memory error flag
10	R	DMA_PERIPH_ERROR	DMA peripheral access error flag
9	R	DMA_MEM_ERROR	DMA memory error flag
8	R	BB_MEM_ERROR	Baseband memory error flag
0	R	FLASH_COPIER_MEM_ERROR	FLASH[0:0] copier memory error flag

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
16	ACCESS_ERROR_ CLEAR	SYSCTRL_ACCESS_ERROR_CLEAR	Clear the access error flags	0x1
12	CC312_MEM_ERROR	CC312_MEM_NO_ERROR_ DETECTED	No CC312 memory error detected	0x0*
		CC312_MEM_ERROR_DETECTED	CC312 has accessed an isolated memory	0x1
10	DMA_PERIPH_ERROR	DMA_PERIPH_NO_ERROR_ DETECTED	No DMA peripheral access error detected	0x0*
		DMA_PERIPH_ERROR_DETECTED	DMA received a peripheral access error	0x1
9	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
8	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
0	FLASH_COPIER_MEM_ ERROR	FLASH0_COPIER_MEM_NO_ ERROR_DETECTED	No FLASH copier memory error detected	0x0*
		FLASH0_COPIER_MEM_ERROR_ DETECTED	FLASH0 copier has accessed an isolated memory	0x1

9.5.0.3 SYSCTRL_MEM_POWER_STARTUP

Bit Field	Read/Write	Field Name	Description
17:16	RW	BB_DRAM_STARTUP	Baseband DRAM[1:0] power startup control
15:8	RW	DRAM_STARTUP	DRAM[7:0] power startup control
1	RW	FLASH_STARTUP	Flash[0:0] power startup control
0	RW	PROM_STARTUP	PROM power startup control

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
17:16	BB_DRAM_STARTUP	BB_DRAM0_POWER_OFF	Baseband DRAM0 startup disabled	0x0
		BB_DRAM1_POWER_OFF	Baseband DRAM1 startup disabled	0x0
		BB_DRAM0_POWER_STARTUP	Baseband DRAM0 startup enabled	0x1
		BB_DRAM1_POWER_STARTUP	Baseband DRAM1 startup enabled	0x2
15:8	DRAM_STARTUP	DRAM0_POWER_OFF	DRAM0 startup disabled	0x0
		DRAM1_POWER_OFF	DRAM1 startup disabled	0x0
		DRAM2_POWER_OFF	DRAM2 startup disabled	0x0
		DRAM3_POWER_OFF	DRAM3 startup disabled	0x0
		DRAM4_POWER_OFF	DRAM4 startup disabled	0x0
		DRAM5_POWER_OFF	DRAM5 startup disabled	0x0
		DRAM6_POWER_OFF	DRAM6 startup disabled	0x0
		DRAM7_POWER_OFF	DRAM7 startup disabled	0x0
		DRAM0_POWER_STARTUP	DRAM0 startup enabled	0x1
		DRAM1_POWER_STARTUP	DRAM1 startup enabled	0x2
		DRAM2_POWER_STARTUP	DRAM2 startup enabled	0x4
		DRAM3_POWER_STARTUP	DRAM3 startup enabled	0x8
		DRAM4_POWER_STARTUP	DRAM4 startup enabled	0x10
		DRAM5_POWER_STARTUP	DRAM5 startup enabled	0x20
		DRAM6_POWER_STARTUP	DRAM6 startup enabled	0x40
		DRAM7_POWER_STARTUP	DRAM7 startup enabled	0x80
1	FLASH_STARTUP	FLASH0_POWER_OFF	FLASH0 startup disabled	0x0*
		FLASH0_POWER_STARTUP	FLASH0 startup enabled	0x1
0	PROM_STARTUP	PROM_POWER_OFF	PROM startup disabled	0x0
		PROM_POWER_STARTUP	PROM startup enabled	0x1*

9.5.0.4 SYSCTRL_MEM_POWER_ENABLE

Bit Field	Read/Write	Field Name	Description
17:16	RW	BB_DRAM_ENABLE	Baseband DRAM[1:0] power enable control
15:8	RW	DRAM_ENABLE	DRAM[7:0] power enable control
1	RW	FLASH_ENABLE	Flash[0:0] power enable control
0	RW	PROM_ENABLE	PROM power enable control

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
17:16	BB_DRAM_ENABLE	BB_DRAM0_POWER_DISABLE	Baseband DRAM0 disabled	0x0*
		BB_DRAM1_POWER_DISABLE	Baseband DRAM1 disabled	0x0*
		BB_DRAM0_POWER_ENABLE	Baseband DRAM0 enabled	0x1
		BB_DRAM1_POWER_ENABLE	Baseband DRAM1 enabled	0x2
15:8	DRAM_ENABLE	DRAM0_POWER_DISABLE	DRAM0 disabled	0x0
		DRAM1_POWER_DISABLE	DRAM1 disabled	0x0
		DRAM2_POWER_DISABLE	DRAM2 disabled	0x0
		DRAM3_POWER_DISABLE	DRAM3 disabled	0x0
		DRAM4_POWER_DISABLE	DRAM4 disabled	0x0
		DRAM5_POWER_DISABLE	DRAM5 disabled	0x0
		DRAM6_POWER_DISABLE	DRAM6 disabled	0x0
		DRAM7_POWER_DISABLE	DRAM7 disabled	0x0
		DRAM0_POWER_ENABLE	DRAM0 enabled	0x1*
		DRAM1_POWER_ENABLE	DRAM1 enabled	0x2
		DRAM2_POWER_ENABLE	DRAM2 enabled	0x4
		DRAM3_POWER_ENABLE	DRAM3 enabled	0x8
		DRAM4_POWER_ENABLE	DRAM4 enabled	0x10
		DRAM5_POWER_ENABLE	DRAM5 enabled	0x20
		DRAM6_POWER_ENABLE	DRAM6 enabled	0x40
		DRAM7_POWER_ENABLE	DRAM7 enabled	0x80
1	FLASH_ENABLE	FLASH0_POWER_DISABLE	FLASH0 disabled	0x0*
		FLASH0_POWER_ENABLE	FLASH0 enabled	0x1
0	PROM_ENABLE	PROM_POWER_DISABLE	PROM disabled	0x0
		PROM_POWER_ENABLE	PROM enabled	0x1*

9.5.0.5 SYSCTRL_MEM_ACCESS_CFG

Bit Field	Read/Write	Field Name	Description
29:24	RW	WAKEUP_ADDR_PACKED	Wakeup restore address in packed 6-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.
17:16	RW	BB_DRAM_ACCESS	Baseband DRAM[1:0] access configuration
15:8	RW	DRAM_ACCESS	DRAM[7:0] access configuration

Bit Field	Read/Write	Field Name	Description
1	RW	FLASH_ACCESS	FLASH[0:0] access configuration
0	RW	PROM_ACCESS	PROM access configuration

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
17:16	BB_DRAM_ACCESS	BB_DRAM0_ACCESS_DISABLE	Baseband DRAM0 access disabled	0x0*
		BB_DRAM1_ACCESS_DISABLE	Baseband DRAM1 access disabled	0x0*
		BB_DRAM0_ACCESS_ENABLE	Baseband DRAM0 access enabled	0x1
		BB_DRAM1_ACCESS_ENABLE	Baseband DRAM1 access enabled	0x2
15:8	DRAM_ACCESS	DRAM0_ACCESS_DISABLE	DRAM0 access disabled	0x0
		DRAM1_ACCESS_DISABLE	DRAM1 access disabled	0x0
		DRAM2_ACCESS_DISABLE	DRAM2 access disabled	0x0
		DRAM3_ACCESS_DISABLE	DRAM3 access disabled	0x0
		DRAM4_ACCESS_DISABLE	DRAM4 access disabled	0x0
		DRAM5_ACCESS_DISABLE	DRAM5 access disabled	0x0
		DRAM6_ACCESS_DISABLE	DRAM6 access disabled	0x0
		DRAM7_ACCESS_DISABLE	DRAM7 access disabled	0x0
		DRAM0_ACCESS_ENABLE	DRAM0 access enabled	0x1*
		DRAM1_ACCESS_ENABLE	DRAM1 access enabled	0x2
		DRAM2_ACCESS_ENABLE	DRAM2 access enabled	0x4
		DRAM3_ACCESS_ENABLE	DRAM3 access enabled	0x8
		DRAM4_ACCESS_ENABLE	DRAM4 access enabled	0x10
		DRAM5_ACCESS_ENABLE	DRAM5 access enabled	0x20
		DRAM6_ACCESS_ENABLE	DRAM6 access enabled	0x40
		DRAM7_ACCESS_ENABLE	DRAM7 access enabled	0x80
1	FLASH_ACCESS	FLASH0_ACCESS_DISABLE	FLASH0 access disabled	0x0*
		FLASH0_ACCESS_ENABLE	FLASH0 access enabled	0x1
0	PROM_ACCESS	PROM_ACCESS_DISABLE	PROM access disabled	0x0
		PROM_ACCESS_ENABLE	PROM access enabled	0x1*

9.5.0.6 SYSCTRL_WAKEUP_ADDR

Bit Field	Read/Write	Field Name	Description
31:0	RW	WAKEUP_ADDR	Wake-up restore address in unpacked 32-bit format.

9.5.0.7 SYSCTRL_MEM_RETENTION_CFG

Bit Field	Read/Write	Field Name	Description
17:16	RW	BB_DRAM_RETENTION	Baseband DRAM[1:0] retention configuration
15:8	RW	DRAM_RETENTION	DRAM[7:0] retention configuration

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
17:16	BB_DRAM_RETENTION	BB_DRAM0_NORMAL_MODE	Baseband DRAM0 normal mode	0x0
		BB_DRAM1_NORMAL_MODE	Baseband DRAM1 normal mode	0x0
		BB_DRAM0_RETENTION_MODE	Baseband DRAM0 retention mode	0x1
		BB_DRAM1_RETENTION_MODE	Baseband DRAM1 retention mode	0x2
15:8	DRAM_RETENTION	DRAM0_NORMAL_MODE	DRAM0 normal mode	0x0
		DRAM1_NORMAL_MODE	DRAM1 normal mode	0x0
		DRAM2_NORMAL_MODE	DRAM2 normal mode	0x0
		DRAM3_NORMAL_MODE	DRAM3 normal mode	0x0
		DRAM4_NORMAL_MODE	DRAM4 normal mode	0x0
		DRAM5_NORMAL_MODE	DRAM5 normal mode	0x0
		DRAM6_NORMAL_MODE	DRAM6 normal mode	0x0
		DRAM7_NORMAL_MODE	DRAM7 normal mode	0x0
		DRAM0_RETENTION_MODE	DRAM0 retention mode	0x1
		DRAM1_RETENTION_MODE	DRAM1 retention mode	0x2
		DRAM2_RETENTION_MODE	DRAM2 retention mode	0x4
		DRAM3_RETENTION_MODE	DRAM3 retention mode	0x8
		DRAM4_RETENTION_MODE	DRAM4 retention mode	0x10
		DRAM5_RETENTION_MODE	DRAM5 retention mode	0x20
		DRAM6_RETENTION_MODE	DRAM6 retention mode	0x40
		DRAM7_RETENTION_MODE	DRAM7 retention mode	0x80

9.5.0.8 SYSCTRL_MEM_TIMING_CFG

Bit Field	Read/Write	Field Name	Description
22:20	RW	RAM_PKA_EMA	RAM extra margin configuration for PKA RAM
17:16	RW	RAM_PKA_EMAW	RAM write extra margin configuration for PKA RAM
6:4	RW	PROM_EMA	PROM extra margin configuration
0	RW	PROM_KEN	PROM bit lines keeper configuration

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
22:20	RAM_PKA_EMA	RAM_PKA_EMA_MIN	RAM minimum extra margin	0x0
		RAM_PKA_EMA_DEFAULT	RAM default extra margin	0x2*
		RAM_PKA_EMA_MAX	RAM maximum extra margin	0x7
17:16	RAM_PKA_EMAW	RAM_PKA_EMAW_DEFAULT	RAM default/minimum extra write margin	0x0*
		RAM_PKA_EMAW_MAX	RAM maximum extra write margin	0x3
6:4	PROM_EMA	PROM_EMA_MIN	PROM minimum extra margin	0x0
		PROM_EMA_DEFAULT	PROM default extra margin	0x2*
		PROM_EMA_MAX	PROM maximum extra margin	0x7
0	PROM_KEN	PROM_KEN_ENABLED	PROM bit lines keeper enabled	0x0
		PROM_KEN_DISABLED	PROM bit lines keeper disabled	0x1*

CHAPTER 10

General Purpose Input/Output

This topic covers the configuration, use, and control of the RSL15‘s 16 GPIO pads.

IMPORTANT: Some RSL15 package variants do not have access to all 16 GPIO pads. For more information, please refer to the data sheet for your selected RSL15 package.

10.1 Overview

The RSL15 system contains 16 general purpose input/output (GPIO) pads that can be configured:

- To support the sensor and communications interfaces, output clocks, and other I/Os
- As general-purpose I/Os controllable from the Arm Cortex-M33 core

The RSL15 system includes dedicated I/O pads for the following functionalities:

- The SWDCLK and SWDIO (also used as JTCK and JTMS) pads for the standard SWJ-DP debug port included with the Arm Cortex-M33 core — for more information, see Section 3.2 “Debug Port” on page 55.
- A reset pad (NRESET) — for more information, see Section 8.4 “Resets” on page 427.
- An antenna pad (RF) — for more information, see Chapter 5 "RF Front-End" on page 140.

All other input and output functionality is routed through 16 configurable GPIOs. For more information about the functional configuration of GPIO pads, see Section 10.2 “Functional Configuration” on page 512.

The GPIO pads support a variety of physical configuration parameters that can be used to properly interface with external components. These configuration parameters include:

- Pull-up and pull-down resistors
- Low-pass input filtering
- Configurable output drive strength

A complete description of the physical configuration of GPIOs can be found in Section 10.3 “Physical Configuration” on page 516.

The "GPIO Pad and Multiplexing" figure (Figure 52) shows a simplified pad drawing for a GPIO. This figure includes the physical configuration, functional multiplexing, and TrustZone security gating of input and output signals.

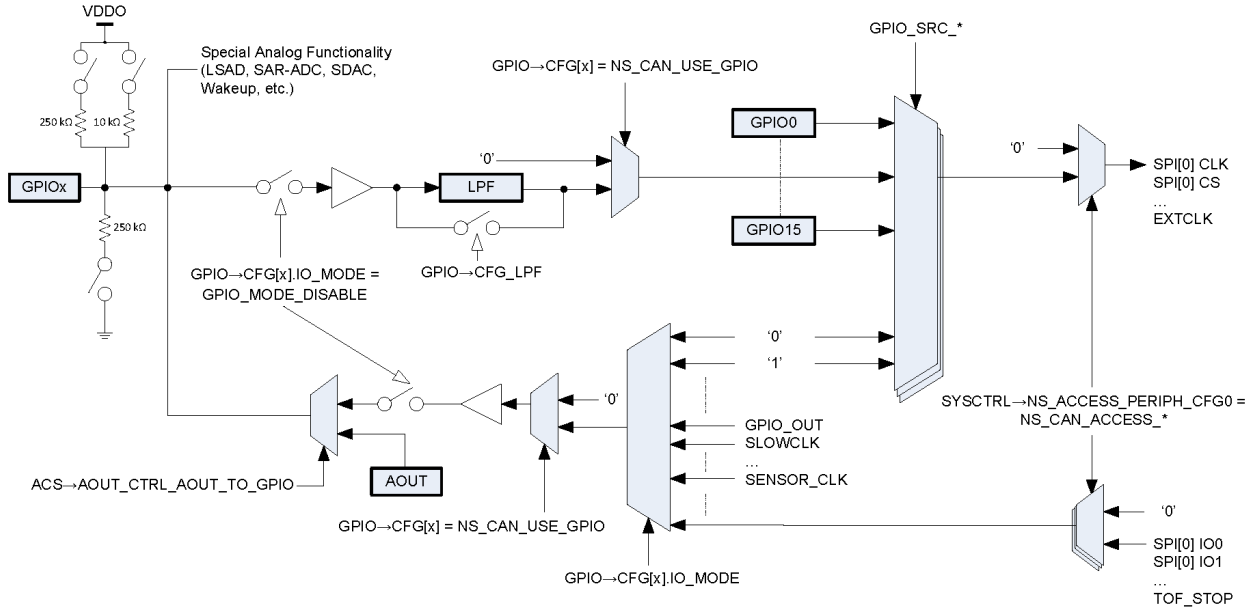


Figure 52. GPIO Pad and Multiplexing

A description of the direct use of GPIO pads as general-purpose inputs/outputs, rather than as part of an interface, is described in Section 10.4 “Direct Control” on page 518.

All of the GPIO pads are powered from the VDDO power supply. For more information about this power supply and its configuration, see Section 8.2 “Power Supply Overview” on page 406.

Access to the GPIOs and the input/output functionalities they provide can be restricted in non-secure modes by clearing the `GPIO_CFG_NS_ACCESS_GPIO` bits in the `GPIO_CFG` registers, and configuring the `SYSCTRL_NS_ACCESS_PERIPH_CFG0` register, respectively.

10.2 Functional Configuration

The GPIO pads can be configured using the `GPIO_CFG_IO_MODE` bit field from the `GPIO_CFG_*` registers:

- For a variety of digital output modes
- For a general-purpose digital input mode, with the input function configured by the `GPIO_SRC_*` registers

The "GPIO Use Case Multiplexing" table (Table 18) contains a list of the multiplexed functional modes for which GPIOs can be configured.

In addition to standard digital functional configuration, certain GPIOs can be configured for operation in both run and low power modes, such as Sleep Mode. Specifically:

- Wakeup functionality is only available on GPIO[0:3]; see Section 8.6 “Power Modes” on page 431 for more information.
- The RTC clock can be output on GPIO0; see Section 7.2 “Clock Generation” on page 383 for more information.

- The PWM (always-on) can be output on GPIO4; see "PWM" on page 598 for more information.
- The simple DAC can be output on GPIO7; see "Simple Low-Power DAC" on page 655 for more information.

These special modes require the GPIO to be placed in a high impedance mode, with its digital input and output driver disabled (`GPIO_MODE_DISABLE`), and all pull-up or pull-down resisters disabled.

IMPORTANT: To prevent cross-talk with the SWD debugging interface, which can potentially crash the IDE, we recommend that you do not run high frequency signals (greater then 1 MHz) on GPIO2 while debugging with the RSL15 Evaluation and Development Board.

CAUTION: While a GPIO can be configured to be both an output and an input, it is the user application's responsibility to ensure that the GPIO is not driving an output to a pad that is also being driven externally. If a GPIO pad signal has two drivers, the physical values and inputs that are read from this pad are considered undefined.

Table 18. GPIO Use Case Multiplexing

GPIO	Mode	Description and Notes
0-3	RTC_CLK_INPUT	External RTC clock source; requires the GPIO to be in high impedance mode. For more information, see Section 7.4 “Clock Distribution” on page 390.
0	RTC_CLK_OUTPUT	RTC clock output. For more information, see Section 7.4 “Clock Distribution” on page 390.
0-3	WAKEUP_SOURCE	Wakeup source from low power modes. For more information on wakeup configuration, see Section 8.6 “Power Modes” on page 431.
2	JTAG_TDO	JTAG Test Data Out. Configuration for use in JTAG mode overrides other uses of GPIO 2, as described in Section 10.2.1 “JTAG I/O Functional Configuration”.
3	JTAG_TDI	JTAG Test Data In. Configuration for use in JTAG mode overrides other uses of GPIO 3, as described in Section 10.2.1 “JTAG I/O Functional Configuration”.
4	JTAG_TRST	JTAG Test Reset. Configuration for use in a 5-wire JTAG configuration overrides other uses of GPIO 4, as described in Section 10.2.1 “JTAG I/O Functional Configuration”.
4	AO-PWM	Always On PWM. This PWM is enabled using the `ACS_PWM_AO_CTRL` register. This PWM signal can be output in sleep modes if the GPIO is configured for high-impedance mode. For more information, see Section 11.4 “PWM” on page 598
7	SDAC_OUTPUT	SDAC output; requires the GPIO to be in high impedance mode. The SDAC is enabled using the `ACS_SDAC_CFG` register. For more information, see Section 12.5 “Simple Low-Power DAC” on page 655
9	SAR_ADC SUPPLY	SAR ADC voltage supply and reference (as configured using `SENSOR_SAR_CFG`). For more information, see Section 12.2 “SAR-ADC ” on page 646.

Table 18. GPIO Use Case Multiplexing (Continued)

GPIO	Mode	Description and Notes
0-15	SAR_ADC_INPUT	Successive Approximation ADC (SAR-ADC) analog input; requires the GPIO to be configured for high-impedance mode. For more information, see Section 12.2 “SAR-ADC ” on page 646.
	LSAD_INPUT	Low Speed ADC (LSAD) analog input; requires the GPIO to be configured for high-impedance mode. For more information, see Section 12.4 “LSAD” on page 651.
	CURRENT_SOURCE_OUTPUT	Current source analog output; requires the GPIO to be configured for high-impedance mode. For more information, see Section 12.6 “Current Source” on page 656
	ACOMP_INPUT	Analog comparator analog input; requires the GPIO to be configured for high-impedance mode. For more information, see Section 12.5 “Simple Low-Power DAC” on page 655.

Table 18. GPIO Use Case Multiplexing (Continued)

GPIO	Mode	Description and Notes
	SLOWCLK (output) SYSCLK (output) USRCLK (output) RCCLK (output) SWCLK (output) EXTCLK (output) RFCLK (output) STANDBYCLK (output) SENSORCLK (output)	Clocking. For more information, see Chapter 7 "Clock Components" on page 381.
	UART0_RX UART0_TX SPI0_MOSI/DATA0 SPI0_MISO/DATA1 SPI0_DATA2 SPI0_DATA3 SPI0_CS SPI0_CLK SPI1_MOSI/DATA0 SPI1_MISO/DATA1 SPI1_DATA2 SPI1_DATA3 SPI1_CS SPI1_CLK I2C0_SCL I2C0_SDA PWM0 PWM1 PWM2 PWM3 PWM4 PWM0_INV PWM1_INV PWM2_INV PWM3_INV PWM4_INV PCM_SERI PCM_SERO PCM_FRAME PCM_CLK	Interfaces. For more information, see Chapter 11 "Communication Interfaces" on page 575.
	AOUT	Analog test output; requires the GPIO to be configured for high-impedance mode. For more information, see Section 10.2.2 “Analog Test Output”

10.2.1 JTAG I/O Functional Configuration

When configured for serial wire (SW) mode, only the reserved SWCLK and SWDIO pads are used by the SWJ-DP interface. When configured for JTAG mode, the SWJ-DP interface uses the following additional pads along with the reserved pads to form a 4- or 5-wire JTAG interface:

- GPIO 2 configured as TDO
- GPIO 3 configured as TDI
- GPIO 4 configured as TRST (5-wire JTAG interface only)

The GPIO_CFG_IO_MODE bit field in the GPIO_CFG registers is overridden for GPIO 4, when CM33_JTAG_TRST_ENABLED is set using the GPIO_JTAG_SW_PAD_CFG_CM33_JTAG_TRST_EN bit from the GPIO_JTAG_SW_PAD_CFG register. The GPIO_CFG_IO_MODE bit field in the GPIO_CFG registers is overridden for GPIOs 2 and 3, when CM33_JTAG_DATA_ENABLED is set using the CM33_JTAG_DATA_EN bit from the GPIO_JTAG_SW_PAD_CFG register.

10.2.2 Analog Test Output

The analog test output (AOUT) can be used to provide access to internal analog and digital signals for test and debug purposes.

IMPORTANT: All signals that are output using AOUT are available for test measurement and debug only. Power supply signals routed to AOUT must not be used to supply external components.

The AOUT signal is configured using the ACS_AOUT_CTRL register, which has the following configuration options:

Test Signal Selection

The ACS_AOUT_CTRL_TEST_AOUT bit-field is used to select the test signal that is output on AOUT. The ACS_AOUT_CTRL_AOUT_IOUT_SEL_TO_GPIO selects between the output of an AOUT test voltage and a current source output based on the PTAT (proportional to absolute temperature) current source used by the power management unit. This must be configured to match the ACS_AOUT_CTRL_TEST_AOUT configuration selecting SEL_IOUT_TO_GPIO for any current source output, and SEL_AOUT_TO_GPIO for all other outputs.

GPIO

The ACS_AOUT_CTRL_AOUT_TO_GPIO bit-field is used to select which GPIO the AOUT test signal is routed to. If a GPIO is configured to output the AOUT test signal, the GPIO output buffer and pull resistors are automatically disabled to ensure that analog circuitry connected through AOUT is not damaged.

IMPORTANT: The ACS_AOUT_CTRL register is also used to configure the RTC output on GPIO 0. Care must be taken when using this register for both use cases, to avoid zeroing out the AOUT configuration when configuring for RTC output and vice versa.

10.3 Physical Configuration

The RSL15 system includes physical configuration parameters for each GPIO. These parameters are set using configuration bits from the appropriate GPIO_CFG_* registers.

If the GPIO is configured as an input pad, the GPIO_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 (10 kΩ) pull-up resistor

In the reset state — for example, when the NRESET pad is driven low — the GPIO output drive is disabled and a weak (250 kΩ) pull-up resistor is enabled. After reset is released, the default GPIO configuration is applied.

The GPIO_CFG_LPF bits in the GPIO_CFG registers enables or disables a low-pass filter that can be used to provide a cleaner input signal by filtering out high frequency noise.

IMPORTANT: For optimal noise performance, we recommend enabling the low-pass filters provided for GPIO inputs when using input signals received at 1 MHz or less. For signals that use a frequency exceeding 1 MHz, the GPIO low-pass filters need to be disabled.

NOTE: When switching from disabled to GPIO output mode:

- From an input mode, the initial output value matches the value that has most recently been read from the input.
- From another output mode, the initial output value matches the current output value.
- The initial output value matches the value shown in the GPIO_OUTPUT_DATA register.

IMPORTANT: When a GPIO is configured as output, its value is based on its previous GPIO data register. Therefore, before configuring a GPIO to produce a specific output level, the application must set the GPIO register first and then configure the GPIO as an output.

If the GPIO is configured as an output pad, the GPIO_CFG_DRIVE bit allows you to select the drive strength for the GPIO output, with settings for 2x, 3x, 5x, and 6x the normal drive strength.

10.3.1 nRESET Pull Resistor Configuration

In the reset state, the nRESET signal has a 100 kΩ pull-up resistor. When the ACS is not in reset, the nRESET signal has a 200 kΩ pull-up resistor. The physical characteristics for the nRESET pad are not otherwise configurable.

10.3.2 JTAG I/O Pad Configuration

Physical configuration of the JTAG TDO, TDI, and TRSET signals are as per their corresponding GPIO physical configuration. The dedicated SWJ-DP pads, used as the SWCLK (JTAG JTCK) and SWDIO (JTAG JTMS) have separate physical configurations, controlled by the GPIO_JTAG_SW_PAD_CFG register. This includes:

- The GPIO_JTAG_SW_PAD_CFG_SWCLK_PULL and GPIO_JTAG_SW_PAD_CFG_SWDIO_PULL bits are used for configuring each pad to use a pull-up or pull-down resistor. These pads have the same pull-resistor configuration options as the GPIO pads, allowing for selection of no pull resistor, a weak or strong pull-up resistor, or a weak pull-down resistor.
- The GPIO_JTAG_SW_PAD_CFG_SWCLK_LPF and GPIO_JTAG_SW_PAD_CFG_SWDIO_LPF bits are used to enable or disable the low-pass filters, which can be used to provide a cleaner input signal by filtering out high frequency noise.

- The GPIO_JTAG_SW_PAD_CFG_SWDIO_DRIVE bit is used to select the drive strength for the SWDIO output, with settings for 2x, 3x, 5x, and 6x the normal drive strength.

10.4 Direct Control

RSL15 can configure any of the general-purpose input/output (GPIO) pads directly, rather than using them through an interface. The function of these GPIO pads is defined by a user application, which can use them for any general-purpose input or output.

The GPIO_MODE register indicates which GPIO pads have been configured for direct application control of the GPIO functionality; bits in this register are set (for example, GPIO*_IS_GPIO_MODE) if they are configured as GPIOs, and cleared otherwise (for example, GPIO_IS_NOT_GPIO_MODE). For any pads that are defined as being in a direct control GPIO mode, the GPIO_DIR register can be written to set the input/output direction for these pads, or read to query the direction of the pads.

The value observed at the digital input pads can be read from the GPIO_INPUT_DATA register. This value is read as 0 for all pads configured as LSAD inputs (see Section 12.4 “LSAD” on page 651), 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 driven to the digital output pads can also be set or queried using the GPIO_OUTPUT_DATA register for any pads that are configured as application-controlled GPIO outputs. Additionally, GPIO outputs can be set or cleared using the GPIO_OUTPUT_DATA_SET and GPIO_OUTPUT_DATA_CLR registers, without changing any other GPIO output values.

IMPORTANT: If any GPIOs are unprotected in Trustzone non-secure code (that is, NS_CAN_USE_GPIO **has been used to set the** GPIO_CFG_NS_ACCESS_GPIO **bit in any** GPIO_CFG **register), use of the** GPIO_OUTPUT_DATA_CLR**,** GPIO_OUTPUT_DATA_SET **and** GPIO_DIR **registers is not supported. In this configuration, all GPIO direction configuration must use the** GPIO_CFG **registers and all direct access to GPIO output data must use the** GPIO_OUTPUT_DATA **register.**

10.4.1 GPIO Interrupts

The GPIOs are supported by a set of four general-purpose configurable interrupts which, when enabled, signal the occurrence of an event or a condition on a specified GPIO pad. See Section 3.3.2 “Nested Vector Interrupt Controller (NVIC)” on page 58 for information regarding interrupt configuration and handling.

Each of the GPIO interrupts can use any of the 16 GPIOs as the triggering input source for the interrupt. 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 bit fields in the GPIO_INT_CFG registers. For each of the four interrupts:

- To select the GPIO pad to use as a trigger for the interrupt, set the GPIO_INT_CFG_SRC bit field.
- To select the event to use as a trigger for the interrupt, use the GPIO_INT_CFG_EVENT bit field. A list of the possible triggering events (including a description of each event) is listed in the "GPIO Interrupt Triggering Events" table (Table 19).

Table 19. GPIO Interrupt Triggering 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.

Each of the GPIO interrupts can be configured to use a debounce filter to eliminate extraneous interrupt triggers; to enable the debounce filter, set the GPIO_INT_CFG_DEBOUNCE_ENABLE bits from the GPIO_INT_CFG registers. If enabled, the debounce filters:

- Transition their output to the new value for the GPIO as soon as detected.
- Hold this new output value for a period of time as defined by the GPIO_INT_DEBOUNCE register; its GPIO_INT_DEBOUNCE_DEBOUNCE_CLK bit selects the SLOWCLK divider used for the debounce clock (selecting between 32 and 1024 cycles), and the GPIO_INT_DEBOUNCE_DEBOUNCE_COUNT bit-field, also in the GPIO_INT_DEBOUNCE register, sets the number of these divided clock cycles to use for the hold period.
- After the hold period has expired, the debounce filter transitions the filters' output to the input value and begins monitoring for new transitions.
- The hold time is reset if the debounce filter configuration is changed, the GPIO interrupt source configuration is changed, or the GPIO interrupt is disabled.

Each GPIO interrupt can also be configured to be accessible or inaccessible in non-secure execution modes using the GPIO_INT_CFG_NS_ACCESS bit in the appropriate GPIO_INT_CFG register. For more information about non-secure application execution, see Chapter 4 "Arm TrustZone CryptoCell-312 Security IP" on page 117.

10.5 GPIO Registers

Register Name	Register Description	Address
GPIO_CFG	Digital IO Configuration. NS-application can access it if NS_ACCESS_GPIO field is set.	0x40000900-0x4000093C
GPIO_INPUT_DATA	Digital IOs Input Data State	0x40000940
GPIO_OUTPUT_DATA	GPIO Output Data Register	0x40000944
GPIO_OUTPUT_DATA_SET	GPIO Output Data Set	0x40000948
GPIO_OUTPUT_DATA_CLR	GPIO Output Data Clear	0x4000094C
GPIO_DIR	Digital IOs Direction State	0x40000950
GPIO_MODE	Digital IOs Mode State	0x40000954
GPIO_INT_CFG	GPIO Interrupt Configuration. NS-application can access it if	0x40000958-

Register Name	Register Description	Address
	the NS_ACCESS field is set.	0x40000964
GPIO_INT_STATUS_S	GPIO Interrupt Status	0x40000968
GPIO_INT_STATUS	GPIO Interrupt Status. NS-application can access it if the GPIO_INT_CFG[3:0].NS_ACCESS field is set.	0x4000096C-0x40000978
GPIO_INT_DEBOUNCE	GPIO Interrupt Button Debounce Filter Time Configuration	0x4000097C
GPIO_JTAG_SW_PAD_CFG	JTAG / SW Pad Configuration Register	0x40000980
GPIO_SRC_SPI	SPI Interface Input Selection	0x40000A00-0x40000A04
GPIO_SRC_SPI_IO	SPI Interface IO Selection	0x40000A08-0x40000A0C
GPIO_SRC_UART	UART Interface Input Selection	0x40000A10
GPIO_SRC_I2C	I2C Input Selection	0x40000A14-0x40000A18
GPIO_SRC_PCM	PCM Input Selection	0x40000A1C
GPIO_SRC_LIN	LIN Input Selection	0x40000A20
GPIO_SRC_NMI	NMI Input Selection	0x40000A24
GPIO_SRC_BB_RX	Baseband controller RX data and clock input selection	0x40000A28
GPIO_SRC_BB_SPI	Baseband controller SPI input selection	0x40000A2C
GPIO_SRC_BB_COEX	Baseband controller WLAN coexistence input selection	0x40000A30
GPIO_SRC_BB_IQ_DATA	Baseband controller IQ data input selection	0x40000A34
GPIO_SRC_BB_IQ_DATA_P	Baseband controller IQ data pulse input selection	0x40000A38
GPIO_SRC_RF_SPI	RF front-end SPI input selection	0x40000A3C
GPIO_SRC_RF_GPIO03	RF front-end GPIOs 0-3 input selection	0x40000A40
GPIO_SRC_RF_GPIO47	RF front-end GPIOs 4-7 input selection	0x40000A44
GPIO_SRC_RF_GPIO89	RF front-end GPIOs 8-9 input selection	0x40000A48
GPIO_SRC_RF_CTE	RF front-end CTE input selection	0x40000A4C
GPIO_SRC_ASCC	ASCC Interface Input Selection	0x40000A50
GPIO_SRC_EXTCLK	EXTCLK Input Selection	0x40000A54
ACS_AOUT_CTRL	Analog Output Configuration Register	0x40001B7C

10.5.0.1 GPIO_CFG

Bit Field	Read/Write	Field Name	Description
13:12	RW	DRIVE	Drive strength configuration
10	RW	LPF	Low Pass Filter enable
9:8	RW	PULL_CTRL	Pull selection
7	RW	NS_ACCESS_GPIO	Non-Secure code can use GPIO (can only be written by a secure code)
6:0	RW	IO_MODE	IO mode selection

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
13:12	DRIVE	GPIO_2X_DRIVE	2x drive strength	0x0
		GPIO_3X_DRIVE	3x drive strength	0x1
		GPIO_5X_DRIVE	5x drive strength	0x2
		GPIO_6X_DRIVE	6x drive strength	0x3*
10	LPF	GPIO_LPF_DISABLE	Disable low pass filter	0x0*
		GPIO_LPF_ENABLE	Enable low pass filter	0x1
9:8	PULL_CTRL	GPIO_NO_PULL	No pull selected	0x0
		GPIO_WEAK_PULL_UP	Weak pull-up selected	0x1*
		GPIO_WEAK_PULL_DOWN	Weak pull-down selected	0x2
		GPIO_STRONG_PULL_UP	Strong pull-up selected	0x3
7	NS_ACCESS_GPIO	NS_CANNOT_USE_GPIO	Non-Secure code cannot use this GPIO	0x0*
		NS_CAN_USE_GPIO	Non-Secure code can use this GPIO	0x1
6:0	IO_MODE	GPIO_MODE_DISABLE	Disable	0x0*
		GPIO_MODE_INPUT	Input mode	0x1
		GPIO_MODE_GPIO_IN	GPIO input mode	0x2
		GPIO_MODE_GPIO_OUT	GPIO output mode	0x3
		GPIO_MODE_SLOWCLK	Output SLOWCLK (slow clock) signal	0x4
		GPIO_MODE_SYSCLK	Output SYSCLK (system clock) signal	0x5
		GPIO_MODE_USRCLK	Output USRCLK (user clock) signal	0x6
		GPIO_MODE_RCCLK	Output RCCLK signal	0x7
		GPIO_MODE_SWCLK_DIV	Output of SWCLK divider signal	0x8
		GPIO_MODE_EXTCLK_DIV	Output of EXTCLK divider signal	0x9

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		GPIO_MODE_RFCLK	Output RFCLK signal	0xA
		GPIO_MODE_STANDBYCLK	Output STANDBYCLK signal	0xB
		GPIO_MODE_SENSORCLK	Output SENSORCLK signal	0xC
		GPIO_MODE_SPI0_IO0	Output SPI0_IO0 interface signal (master SERO)	0xD
		GPIO_MODE_SPI0_IO1	Output SPI0_IO1 interface signal (slave SERO)	0xE
		GPIO_MODE_SPI0_IO2	Output SPI0_IO2 interface signal	0xF
		GPIO_MODE_SPI0_IO3	Output SPI0_IO3 interface signal	0x10
		GPIO_MODE_SPI0_CS	Output SPI0_CS interface signal	0x11
		GPIO_MODE_SPI0_CLK	Output SPI0_CLK interface signal	0x12
		GPIO_MODE_SPI1_IO0	Output SPI1_IO0 interface signal (master SERO)	0x13
		GPIO_MODE_SPI1_IO1	Output SPI1_IO1 interface signal (slave SERO)	0x14
		GPIO_MODE_SPI1_IO2	Output SPI1_IO2 interface signal	0x15
		GPIO_MODE_SPI1_IO3	Output SPI1_IO3 interface signal	0x16
		GPIO_MODE_SPI1_CS	Output SPI1_CS interface signal	0x17
		GPIO_MODE_SPI1_CLK	Output SPI1_CLK interface signal	0x18
		GPIO_MODE_UART0_TX	Output UART0_TX interface signal	0x19
		GPIO_MODE_I2C0_SCL	Output I2C0 SCL interface signal (open collector)	0x1A
		GPIO_MODE_I2C0_SDA	Output I2C0 SDA interface signal (open collector)	0x1B
		GPIO_MODE_I2C1_SCL	Output I2C1 SCL interface signal (open collector)	0x1C
		GPIO_MODE_I2C1_SDA	Output I2C1 SDA interface signal (open collector)	0x1D
		GPIO_MODE_PCM0_SERO	Output PCM0_SERO interface signal	0x1E
		GPIO_MODE_PCM0_FRAME	Output PCM0_FRAME interface signal	0x1F
		GPIO_MODE_PWM0	Output PWM0 interface signal	0x20
		GPIO_MODE_PWM1	Output PWM1 interface signal	0x21
		GPIO_MODE_PWM2	Output PWM2 interface signal	0x22

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		GPIO_MODE_PWM3	Output PWM3 interface signal	0x23
		GPIO_MODE_PWM4	Output PWM4 interface signal	0x24
		GPIO_MODE_PWM0_INV	Output PWM0 interface signal inverted	0x25
		GPIO_MODE_PWM1_INV	Output PWM1 interface signal inverted	0x26
		GPIO_MODE_PWM2_INV	Output PWM2 interface signal inverted	0x27
		GPIO_MODE_PWM3_INV	Output PWM3 interface signal inverted	0x28
		GPIO_MODE_PWM4_INV	Output PWM4 interface signal inverted	0x29
		GPIO_MODE_LIN0_TX	Output LIN0_TX interface signal	0x2A
		GPIO_MODE_BB_TX_DATA	Output baseband controller TX data signal	0x2B
		GPIO_MODE_BB_TX_DATA_VALID	Output baseband controller TX data valid signal	0x2C
		GPIO_MODE_BB_SPI_CSN	Output baseband controller SPI_CSN signal	0x2D
		GPIO_MODE_BB_SPI_CLK	Output baseband controller SPI_CLK signal	0x2E
		GPIO_MODE_BB_SPI_MOSI	Output baseband controller SPI_MOSI signal	0x2F
		GPIO_MODE_BB_DBG_0	Output baseband controller debug port (bit 0) signal	0x30
		GPIO_MODE_BB_DBG_1	Output baseband controller debug port (bit 1) signal	0x31
		GPIO_MODE_BB_DBG_2	Output baseband controller debug port (bit 2) signal	0x32
		GPIO_MODE_BB_DBG_3	Output baseband controller debug port (bit 3) signal	0x33
		GPIO_MODE_BB_DBG_4	Output baseband controller debug port (bit 4) signal	0x34
		GPIO_MODE_BB_DBG_5	Output baseband controller debug port (bit 5) signal	0x35
		GPIO_MODE_BB_DBG_6	Output baseband controller debug port (bit 6) signal	0x36
		GPIO_MODE_BB_DBG_7	Output baseband controller debug port (bit 7) signal	0x37
		GPIO_MODE_BB_BLE_SYNC	Output baseband controller wlan coex	0x38

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
			signal sync	
		GPIO_MODE_BB_BLE_IN_PROCESS	Output baseband controller wlan coex signal in_process	0x39
		GPIO_MODE_BB_BLE_TX	Output baseband controller wlan coex signal Tx	0x3A
		GPIO_MODE_BB_BLE_RX	Output baseband controller wlan coex signal Rx	0x3B
		GPIO_MODE_BB_BLE_PTI_0	Output baseband controller wlan coex signal PTI (bit 0)	0x3C
		GPIO_MODE_BB_BLE_PTI_1	Output baseband controller wlan coex signal PTI (bit 1)	0x3D
		GPIO_MODE_BB_BLE_PTI_2	Output baseband controller wlan coex signal PTI (bit 2)	0x3E
		GPIO_MODE_BB_BLE_PTI_3	Output baseband controller wlan coex signal PTI (bit 3)	0x3F
		GPIO_MODE_BB_ANT_SW_EN	Output baseband controller antenna switch enable	0x40
		GPIO_MODE_BB_ANT_SW_0	Output baseband controller antenna switch (bit 0)	0x41
		GPIO_MODE_BB_ANT_SW_1	Output baseband controller antenna switch (bit 1)	0x42
		GPIO_MODE_BB_ANT_SW_2	Output baseband controller antenna switch (bit 2)	0x43
		GPIO_MODE_BB_ANT_SW_3	Output baseband controller antenna switch (bit 3)	0x44
		GPIO_MODE_BB_ANT_SW_4	Output baseband controller antenna switch (bit 4)	0x45
		GPIO_MODE_BB_ANT_SW_5	Output baseband controller antenna switch (bit 5)	0x46
		GPIO_MODE_BB_ANT_SW_6	Output baseband controller antenna switch (bit 6)	0x47
		GPIO_MODE_BB_CTE_MODE	Output baseband controller CTE mode	0x48
		GPIO_MODE_BB_CTE_SAMPLE_P	Output baseband controller CTE sample pulse	0x49
		GPIO_MODE_RF_SPI_MISO	Output RF front-end SPI_MISO interface signal	0x4A

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		GPIO_MODE_RF_GPIO0	Output RF front-end GPIO0 output signal (RX_DATA)	0x4B
		GPIO_MODE_RF_GPIO1	Output RF front-end GPIO1 output signal (RX_CLK)	0x4C
		GPIO_MODE_RF_GPIO2	Output RF front-end GPIO2 output signal (SYNC_P)	0x4D
		GPIO_MODE_RF_GPIO3	Output RF front-end GPIO3 output signal	0x4E
		GPIO_MODE_RF_GPIO4	Output RF front-end GPIO4 output signal	0x4F
		GPIO_MODE_RF_GPIO5	Output RF front-end GPIO5 output signal	0x50
		GPIO_MODE_RF_GPIO6	Output RF front-end GPIO6 output signal	0x51
		GPIO_MODE_RF_GPIO7	Output RF front-end GPIO7 output signal	0x52
		GPIO_MODE_RF_GPIO8	Output RF front-end GPIO8 output signal	0x53
		GPIO_MODE_RF_GPIO9	Output RF front-end GPIO9 output signal	0x54
		GPIO_MODE_RF_IQ_DATA_P	Output RF front-end IQ data pulse signal	0x55
		GPIO_MODE_RF_I_DATA_0	Output RF front-end I data (bit 0)	0x56
		GPIO_MODE_RF_I_DATA_1	Output RF front-end I data (bit 1)	0x57
		GPIO_MODE_RF_I_DATA_2	Output RF front-end I data (bit 2)	0x58
		GPIO_MODE_RF_I_DATA_3	Output RF front-end I data (bit 3)	0x59
		GPIO_MODE_RF_I_DATA_4	Output RF front-end I data (bit 4)	0x5A
		GPIO_MODE_RF_I_DATA_5	Output RF front-end I data (bit 5)	0x5B
		GPIO_MODE_RF_I_DATA_6	Output RF front-end I data (bit 6)	0x5C
		GPIO_MODE_RF_I_DATA_7	Output RF front-end I data (bit 7)	0x5D
		GPIO_MODE_RF_Q_DATA_0	Output RF front-end Q data (bit 0)	0x5E
		GPIO_MODE_RF_Q_DATA_1	Output RF front-end Q data (bit 1)	0x5F
		GPIO_MODE_RF_Q_DATA_2	Output RF front-end Q data (bit 2)	0x60
		GPIO_MODE_RF_Q_DATA_3	Output RF front-end Q data (bit 3)	0x61

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		GPIO_MODE_RF_Q_DATA_4	Output RF front-end Q data (bit 4)	0x62
		GPIO_MODE_RF_Q_DATA_5	Output RF front-end Q data (bit 5)	0x63
		GPIO_MODE_RF_Q_DATA_6	Output RF front-end Q data (bit 6)	0x64
		GPIO_MODE_RF_Q_DATA_7	Output RF front-end Q data (bit 7)	0x65
		GPIO_MODE_RF_ANT_SW_0	Output RF front-end antenna switch (bit 0)	0x66
		GPIO_MODE_RF_ANT_SW_1	Output RF front-end antenna switch (bit 1)	0x67
		GPIO_MODE_RF_ANT_SW_2	Output RF front-end antenna switch (bit 2)	0x68
		GPIO_MODE_RF_ANT_SW_3	Output RF front-end antenna switch (bit 3)	0x69
		GPIO_MODE_TOF_START	Output Time Of Flight timer start trigger	0x6A
		GPIO_MODE_TOF_STOP	Output Time Of Flight timer stop trigger	0x6B

10.5.0.2 GPIO_INPUT_DATA

Bit Field	Read/Write	Field Name	Description
15:0	R	DATA	GPIO[15:0] read data

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
15:0	DATA	GPIO0_IN_LOW	GPIO pad is low	0x0*
		GPIO1_IN_LOW	GPIO pad is low	0x0*
		GPIO2_IN_LOW	GPIO pad is low	0x0*
		GPIO3_IN_LOW	GPIO pad is low	0x0*
		GPIO4_IN_LOW	GPIO pad is low	0x0*
		GPIO5_IN_LOW	GPIO pad is low	0x0*
		GPIO6_IN_LOW	GPIO pad is low	0x0*
		GPIO7_IN_LOW	GPIO pad is low	0x0*
		GPIO8_IN_LOW	GPIO pad is low	0x0*
		GPIO9_IN_LOW	GPIO pad is low	0x0*
		GPIO10_IN_LOW	GPIO pad is low	0x0*
		GPIO11_IN_LOW	GPIO pad is low	0x0*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		GPIO12_IN_LOW	GPIO pad is low	0x0*
		GPIO13_IN_LOW	GPIO pad is low	0x0*
		GPIO14_IN_LOW	GPIO pad is low	0x0*
		GPIO15_IN_LOW	GPIO pad is low	0x0*
		GPIO0_IN_HIGH	GPIO pad is high	0x1
		GPIO1_IN_HIGH	GPIO pad is high	0x2
		GPIO2_IN_HIGH	GPIO pad is high	0x4
		GPIO3_IN_HIGH	GPIO pad is high	0x8
		GPIO4_IN_HIGH	GPIO pad is high	0x10
		GPIO5_IN_HIGH	GPIO pad is high	0x20
		GPIO6_IN_HIGH	GPIO pad is high	0x40
		GPIO7_IN_HIGH	GPIO pad is high	0x80
		GPIO8_IN_HIGH	GPIO pad is high	0x100
		GPIO9_IN_HIGH	GPIO pad is high	0x200
		GPIO10_IN_HIGH	GPIO pad is high	0x400
		GPIO11_IN_HIGH	GPIO pad is high	0x800
		GPIO12_IN_HIGH	GPIO pad is high	0x1000
		GPIO13_IN_HIGH	GPIO pad is high	0x2000
		GPIO14_IN_HIGH	GPIO pad is high	0x4000
		GPIO15_IN_HIGH	GPIO pad is high	0x8000

10.5.0.3 GPIO_OUTPUT_DATA

Bit Field	Read/Write	Field Name	Description
15:0	RW	DATA	GPIO[15:0] output data

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
15:0	DATA	GPIO0_OUT_LOW	Set the GPIO pad to low if IO_MODE is GPIO	0x0*
		GPIO1_OUT_LOW	Set the GPIO pad to low if IO_MODE is GPIO	0x0*
		GPIO2_OUT_LOW	Set the GPIO pad to low if IO_MODE is GPIO	0x0*
		GPIO3_OUT_LOW	Set the GPIO pad to low if IO_MODE is	0x0*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
			GPIO	
		GPIO4_OUT_LOW	Set the GPIO pad to low if IO_MODE is GPIO	0x0*
		GPIO5_OUT_LOW	Set the GPIO pad to low if IO_MODE is GPIO	0x0*
		GPIO6_OUT_LOW	Set the GPIO pad to low if IO_MODE is GPIO	0x0*
		GPIO7_OUT_LOW	Set the GPIO pad to low if IO_MODE is GPIO	0x0*
		GPIO8_OUT_LOW	Set the GPIO pad to low if IO_MODE is GPIO	0x0*
		GPIO9_OUT_LOW	Set the GPIO pad to low if IO_MODE is GPIO	0x0*
		GPIO10_OUT_LOW	Set the GPIO pad to low if IO_MODE is GPIO	0x0*
		GPIO11_OUT_LOW	Set the GPIO pad to low if IO_MODE is GPIO	0x0*
		GPIO12_OUT_LOW	Set the GPIO pad to low if IO_MODE is GPIO	0x0*
		GPIO13_OUT_LOW	Set the GPIO pad to low if IO_MODE is GPIO	0x0*
		GPIO14_OUT_LOW	Set the GPIO pad to low if IO_MODE is GPIO	0x0*
		GPIO15_OUT_LOW	Set the GPIO pad to low if IO_MODE is GPIO	0x0*
		GPIO0_OUT_HIGH	Set the GPIO pad to high if IO_MODE is GPIO	0x1
		GPIO1_OUT_HIGH	Set the GPIO pad to high if IO_MODE is GPIO	0x2
		GPIO2_OUT_HIGH	Set the GPIO pad to high if IO_MODE is GPIO	0x4
		GPIO3_OUT_HIGH	Set the GPIO pad to high if IO_MODE is GPIO	0x8
		GPIO4_OUT_HIGH	Set the GPIO pad to high if IO_MODE is GPIO	0x10
		GPIO5_OUT_HIGH	Set the GPIO pad to high if IO_MODE is GPIO	0x20

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		GPIO6_OUT_HIGH	Set the GPIO pad to high if IO_MODE is GPIO	0x40
		GPIO7_OUT_HIGH	Set the GPIO pad to high if IO_MODE is GPIO	0x80
		GPIO8_OUT_HIGH	Set the GPIO pad to high if IO_MODE is GPIO	0x100
		GPIO9_OUT_HIGH	Set the GPIO pad to high if IO_MODE is GPIO	0x200
		GPIO10_OUT_HIGH	Set the GPIO pad to high if IO_MODE is GPIO	0x400
		GPIO11_OUT_HIGH	Set the GPIO pad to high if IO_MODE is GPIO	0x800
		GPIO12_OUT_HIGH	Set the GPIO pad to high if IO_MODE is GPIO	0x1000
		GPIO13_OUT_HIGH	Set the GPIO pad to high if IO_MODE is GPIO	0x2000
		GPIO14_OUT_HIGH	Set the GPIO pad to high if IO_MODE is GPIO	0x4000
		GPIO15_OUT_HIGH	Set the GPIO pad to high if IO_MODE is GPIO	0x8000

10.5.0.4 GPIO_OUTPUT_DATA_SET

Bit Field	Read/Write	Field Name	Description
15:0	W	GPIO	GPIO[15:0] output data set

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
15:0	GPIO	GPIO0_SET	Set the GPIO output data high	0x1
		GPIO1_SET	Set the GPIO output data high	0x2
		GPIO2_SET	Set the GPIO output data high	0x4
		GPIO3_SET	Set the GPIO output data high	0x8
		GPIO4_SET	Set the GPIO output data high	0x10
		GPIO5_SET	Set the GPIO output data high	0x20
		GPIO6_SET	Set the GPIO output data high	0x40
		GPIO7_SET	Set the GPIO output data high	0x80

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		GPIO8_SET	Set the GPIO output data high	0x100
		GPIO9_SET	Set the GPIO output data high	0x200
		GPIO10_SET	Set the GPIO output data high	0x400
		GPIO11_SET	Set the GPIO output data high	0x800
		GPIO12_SET	Set the GPIO output data high	0x1000
		GPIO13_SET	Set the GPIO output data high	0x2000
		GPIO14_SET	Set the GPIO output data high	0x4000
		GPIO15_SET	Set the GPIO output data high	0x8000

10.5.0.5 GPIO_OUTPUT_DATA_CLR

Bit Field	Read/Write	Field Name	Description
15:0	W	GPIO	GPIO[15:0] output data clear

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
15:0	GPIO	GPIO0_CLR	Set the GPIO output data low	0x1
		GPIO1_CLR	Set the GPIO output data low	0x2
		GPIO2_CLR	Set the GPIO output data low	0x4
		GPIO3_CLR	Set the GPIO output data low	0x8
		GPIO4_CLR	Set the GPIO output data low	0x10
		GPIO5_CLR	Set the GPIO output data low	0x20
		GPIO6_CLR	Set the GPIO output data low	0x40
		GPIO7_CLR	Set the GPIO output data low	0x80
		GPIO8_CLR	Set the GPIO output data low	0x100
		GPIO9_CLR	Set the GPIO output data low	0x200
		GPIO10_CLR	Set the GPIO output data low	0x400
		GPIO11_CLR	Set the GPIO output data low	0x800
		GPIO12_CLR	Set the GPIO output data low	0x1000
		GPIO13_CLR	Set the GPIO output data low	0x2000
		GPIO14_CLR	Set the GPIO output data low	0x4000
		GPIO15_CLR	Set the GPIO output data low	0x8000

10.5.0.6 GPIO_DIR

Bit Field	Read/Write	Field Name	Description
15:0	R	GPIO	Get GPIO[15:0] direction
15:0	W	GPIO	Set GPIO[15:0] GPIO direction (only in GPIO_MODE_GPIO_x)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
15:0	GPIO	GPIO0_STATUS_IN	GPIO is an input	0x0*
		GPIO1_STATUS_IN	GPIO is an input	0x0*
		GPIO2_STATUS_IN	GPIO is an input	0x0*
		GPIO3_STATUS_IN	GPIO is an input	0x0*
		GPIO4_STATUS_IN	GPIO is an input	0x0*
		GPIO5_STATUS_IN	GPIO is an input	0x0*
		GPIO6_STATUS_IN	GPIO is an input	0x0*
		GPIO7_STATUS_IN	GPIO is an input	0x0*
		GPIO8_STATUS_IN	GPIO is an input	0x0*
		GPIO9_STATUS_IN	GPIO is an input	0x0*
		GPIO10_STATUS_IN	GPIO is an input	0x0*
		GPIO11_STATUS_IN	GPIO is an input	0x0*
		GPIO12_STATUS_IN	GPIO is an input	0x0*
		GPIO13_STATUS_IN	GPIO is an input	0x0*
		GPIO14_STATUS_IN	GPIO is an input	0x0*
		GPIO15_STATUS_IN	GPIO is an input	0x0*
		GPIO0_STATUS_OUT	GPIO is an output	0x1
		GPIO1_STATUS_OUT	GPIO is an output	0x2
		GPIO2_STATUS_OUT	GPIO is an output	0x4
		GPIO3_STATUS_OUT	GPIO is an output	0x8
		GPIO4_STATUS_OUT	GPIO is an output	0x10
		GPIO5_STATUS_OUT	GPIO is an output	0x20
		GPIO6_STATUS_OUT	GPIO is an output	0x40
		GPIO7_STATUS_OUT	GPIO is an output	0x80
		GPIO8_STATUS_OUT	GPIO is an output	0x100
		GPIO9_STATUS_OUT	GPIO is an output	0x200
		GPIO10_STATUS_OUT	GPIO is an output	0x400

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		GPIO11_STATUS_OUT	GPIO is an output	0x800
		GPIO12_STATUS_OUT	GPIO is an output	0x1000
		GPIO13_STATUS_OUT	GPIO is an output	0x2000
		GPIO14_STATUS_OUT	GPIO is an output	0x4000
		GPIO15_STATUS_OUT	GPIO is an output	0x8000
15:0	GPIO	GPIO0_DIR_IN	Set GPIO to input if GPIO_MODE_GPIO_IN_x	0x0
		GPIO1_DIR_IN	Set GPIO to input if GPIO_MODE_GPIO_IN_x	0x0
		GPIO2_DIR_IN	Set GPIO to input if GPIO_MODE_GPIO_IN_x	0x0
		GPIO3_DIR_IN	Set GPIO to input if GPIO_MODE_GPIO_IN_x	0x0
		GPIO4_DIR_IN	Set GPIO to input if GPIO_MODE_GPIO_IN_x	0x0
		GPIO5_DIR_IN	Set GPIO to input if GPIO_MODE_GPIO_IN_x	0x0
		GPIO6_DIR_IN	Set GPIO to input if GPIO_MODE_GPIO_IN_x	0x0
		GPIO7_DIR_IN	Set GPIO to input if GPIO_MODE_GPIO_IN_x	0x0
		GPIO8_DIR_IN	Set GPIO to input if GPIO_MODE_GPIO_IN_x	0x0
		GPIO9_DIR_IN	Set GPIO to input if GPIO_MODE_GPIO_IN_x	0x0
		GPIO10_DIR_IN	Set GPIO to input if GPIO_MODE_GPIO_IN_x	0x0
		GPIO11_DIR_IN	Set GPIO to input if GPIO_MODE_GPIO_IN_x	0x0
		GPIO12_DIR_IN	Set GPIO to input if GPIO_MODE_GPIO_IN_x	0x0
		GPIO13_DIR_IN	Set GPIO to input if GPIO_MODE_GPIO_IN_x	0x0
		GPIO14_DIR_IN	Set GPIO to input if GPIO_MODE_GPIO_IN_x	0x0
		GPIO15_DIR_IN	Set GPIO to input if GPIO_MODE_GPIO_IN_x	0x0

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		GPIO0_DIR_OUT	Set GPIO to output if GPIO_MODE_GPIO_OUT_x	0x1
		GPIO1_DIR_OUT	Set GPIO to output if GPIO_MODE_GPIO_OUT_x	0x2
		GPIO2_DIR_OUT	Set GPIO to output if GPIO_MODE_GPIO_OUT_x	0x4
		GPIO3_DIR_OUT	Set GPIO to output if GPIO_MODE_GPIO_OUT_x	0x8
		GPIO4_DIR_OUT	Set GPIO to output if GPIO_MODE_GPIO_OUT_x	0x10
		GPIO5_DIR_OUT	Set GPIO to output if GPIO_MODE_GPIO_OUT_x	0x20
		GPIO6_DIR_OUT	Set GPIO to output if GPIO_MODE_GPIO_OUT_x	0x40
		GPIO7_DIR_OUT	Set GPIO to output if GPIO_MODE_GPIO_OUT_x	0x80
		GPIO8_DIR_OUT	Set GPIO to output if GPIO_MODE_GPIO_OUT_x	0x100
		GPIO9_DIR_OUT	Set GPIO to output if GPIO_MODE_GPIO_OUT_x	0x200
		GPIO10_DIR_OUT	Set GPIO to output if GPIO_MODE_GPIO_OUT_x	0x400
		GPIO11_DIR_OUT	Set GPIO to output if GPIO_MODE_GPIO_OUT_x	0x800
		GPIO12_DIR_OUT	Set GPIO to output if GPIO_MODE_GPIO_OUT_x	0x1000
		GPIO13_DIR_OUT	Set GPIO to output if GPIO_MODE_GPIO_OUT_x	0x2000
		GPIO14_DIR_OUT	Set GPIO to output if GPIO_MODE_GPIO_OUT_x	0x4000
		GPIO15_DIR_OUT	Set GPIO to output if GPIO_MODE_GPIO_OUT_x	0x8000

10.5.0.7 GPIO_MODE

Bit Field	Read/Write	Field Name	Description
15:0	R	GPIO	GPIO[15:0] mode

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
15:0	GPIO	GPIO0_IS_NOT_GPIO_MODE	This GPIO is not configured as a CM33 controlled GPIO	0x0*
		GPIO1_IS_NOT_GPIO_MODE	This GPIO is not configured as a CM33 controlled GPIO	0x0*
		GPIO2_IS_NOT_GPIO_MODE	This GPIO is not configured as a CM33 controlled GPIO	0x0*
		GPIO3_IS_NOT_GPIO_MODE	This GPIO is not configured as a CM33 controlled GPIO	0x0*
		GPIO4_IS_NOT_GPIO_MODE	This GPIO is not configured as a CM33 controlled GPIO	0x0*
		GPIO5_IS_NOT_GPIO_MODE	This GPIO is not configured as a CM33 controlled GPIO	0x0*
		GPIO6_IS_NOT_GPIO_MODE	This GPIO is not configured as a CM33 controlled GPIO	0x0*
		GPIO7_IS_NOT_GPIO_MODE	This GPIO is not configured as a CM33 controlled GPIO	0x0*
		GPIO8_IS_NOT_GPIO_MODE	This GPIO is not configured as a CM33 controlled GPIO	0x0*
		GPIO9_IS_NOT_GPIO_MODE	This GPIO is not configured as a CM33 controlled GPIO	0x0*
		GPIO10_IS_NOT_GPIO_MODE	This GPIO is not configured as a CM33 controlled GPIO	0x0*
		GPIO11_IS_NOT_GPIO_MODE	This GPIO is not configured as a CM33 controlled GPIO	0x0*
		GPIO12_IS_NOT_GPIO_MODE	This GPIO is not configured as a CM33 controlled GPIO	0x0*
		GPIO13_IS_NOT_GPIO_MODE	This GPIO is not configured as a CM33 controlled GPIO	0x0*
		GPIO14_IS_NOT_GPIO_MODE	This GPIO is not configured as a CM33 controlled GPIO	0x0*
		GPIO15_IS_NOT_GPIO_MODE	This GPIO is not configured as a CM33 controlled GPIO	0x0*
		GPIO0_IS_GPIO_MODE	This GPIO is configured as a CM33 controlled GPIO	0x1
		GPIO1_IS_GPIO_MODE	This GPIO is configured as a CM33 controlled GPIO	0x2
		GPIO2_IS_GPIO_MODE	This GPIO is configured as a CM33 controlled GPIO	0x4

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		GPIO3_IS_GPIO_MODE	This GPIO is configured as a CM33 controlled GPIO	0x8
		GPIO4_IS_GPIO_MODE	This GPIO is configured as a CM33 controlled GPIO	0x10
		GPIO5_IS_GPIO_MODE	This GPIO is configured as a CM33 controlled GPIO	0x20
		GPIO6_IS_GPIO_MODE	This GPIO is configured as a CM33 controlled GPIO	0x40
		GPIO7_IS_GPIO_MODE	This GPIO is configured as a CM33 controlled GPIO	0x80
		GPIO8_IS_GPIO_MODE	This GPIO is configured as a CM33 controlled GPIO	0x100
		GPIO9_IS_GPIO_MODE	This GPIO is configured as a CM33 controlled GPIO	0x200
		GPIO10_IS_GPIO_MODE	This GPIO is configured as a CM33 controlled GPIO	0x400
		GPIO11_IS_GPIO_MODE	This GPIO is configured as a CM33 controlled GPIO	0x800
		GPIO12_IS_GPIO_MODE	This GPIO is configured as a CM33 controlled GPIO	0x1000
		GPIO13_IS_GPIO_MODE	This GPIO is configured as a CM33 controlled GPIO	0x2000
		GPIO14_IS_GPIO_MODE	This GPIO is configured as a CM33 controlled GPIO	0x4000
		GPIO15_IS_GPIO_MODE	This GPIO is configured as a CM33 controlled GPIO	0x8000

10.5.0.8 GPIO_INT_CFG

Bit Field	Read/Write	Field Name	Description
12	RW	NS_ACCESS	Non-Secure code can access this GPIO_INT (can only be written by a secure code)
11	RW	DEBOUNCE_ENABLE	Interrupt button debounce filter enable/disable
10:8	RW	EVENT	Interrupt event configuration
4:0	RW	SRC	Interrupt input selection

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
12	NS_ACCESS	NS_CANNOT_ACCESS_GPIO_INT	Non-Secure code cannot access the GPIO_INT	0x0*
		NS_CAN_ACCESS_GPIO_INT	Non-Secure code can access the GPIO_INT	0x1
11	DEBOUNCE_ENABLE	GPIO_DEBOUNCE_DISABLE	Button debounce filter disabled	0x0*
		GPIO_DEBOUNCE_ENABLE	Button debounce filter enabled	0x1
10:8	EVENT	GPIO_EVENT_NONE	Interrupt not triggered	0x0*
		GPIO_EVENT_HIGH_LEVEL	Interrupt triggered on high state	0x1
		GPIO_EVENT_LOW_LEVEL	Interrupt triggered on low state	0x2
		GPIO_EVENT_RISING_EDGE	Interrupt triggered on rising edge	0x3
		GPIO_EVENT_FALLING_EDGE	Interrupt triggered on falling edge	0x4
		GPIO_EVENT_TRANSITION	Interrupt triggered on any edge	0x5
4:0	SRC	GPIO_SRC_GPIO_0	Select GPIO[0] as source	0x0*
		GPIO_SRC_GPIO_1	Select GPIO[1] as source	0x1
		GPIO_SRC_GPIO_2	Select GPIO[2] as source	0x2
		GPIO_SRC_GPIO_3	Select GPIO[3] as source	0x3
		GPIO_SRC_GPIO_4	Select GPIO[4] as source	0x4
		GPIO_SRC_GPIO_5	Select GPIO[5] as source	0x5
		GPIO_SRC_GPIO_6	Select GPIO[6] as source	0x6
		GPIO_SRC_GPIO_7	Select GPIO[7] as source	0x7
		GPIO_SRC_GPIO_8	Select GPIO[8] as source	0x8
		GPIO_SRC_GPIO_9	Select GPIO[9] as source	0x9
		GPIO_SRC_GPIO_10	Select GPIO[10] as source	0xA
		GPIO_SRC_GPIO_11	Select GPIO[11] as source	0xB
		GPIO_SRC_GPIO_12	Select GPIO[12] as source	0xC
		GPIO_SRC_GPIO_13	Select GPIO[13] as source	0xD
		GPIO_SRC_GPIO_14	Select GPIO[14] as source	0xE
		GPIO_SRC_GPIO_15	Select GPIO[15] as source	0xF

10.5.0.9 GPIO_INT_STATUS_S

Bit Field	Read/Write	Field Name	Description
7	R	GPIO_INT3_STATUS	GPIO interrupt 3 status
6	R	GPIO_INT2_STATUS	GPIO interrupt 2 status
5	R	GPIO_INT1_STATUS	GPIO interrupt 1 status
4	R	GPIO_INT0_STATUS	GPIO interrupt 0 status
3	W	GPIO_INT3_CLEAR	GPIO interrupt 3 clear
2	W	GPIO_INT2_CLEAR	GPIO interrupt 2 clear
1	W	GPIO_INT1_CLEAR	GPIO interrupt 1 clear
0	W	GPIO_INT0_CLEAR	GPIO interrupt 0 clear

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
7	GPIO_INT3_STATUS	GPIO_INT3_FALSE	Interrupt 3 not triggered	0x0*
		GPIO_INT3_TRUE	Interrupt 3 triggered	0x1
6	GPIO_INT2_STATUS	GPIO_INT2_FALSE	Interrupt 2 not triggered	0x0*
		GPIO_INT2_TRUE	Interrupt 2 triggered	0x1
5	GPIO_INT1_STATUS	GPIO_INT1_FALSE	Interrupt 1 not triggered	0x0*
		GPIO_INT1_TRUE	Interrupt 1 triggered	0x1
4	GPIO_INT0_STATUS	GPIO_INT0_FALSE	Interrupt 0 not triggered	0x0*
		GPIO_INT0_TRUE	Interrupt 0 triggered	0x1
3	GPIO_INT3_CLEAR	GPIO_INT3_CLEAR	Interrupt 3 clear	0x1
2	GPIO_INT2_CLEAR	GPIO_INT2_CLEAR	Interrupt 2 clear	0x1
1	GPIO_INT1_CLEAR	GPIO_INT1_CLEAR	Interrupt 1 clear	0x1
0	GPIO_INT0_CLEAR	GPIO_INT0_CLEAR	Interrupt 0 clear	0x1

10.5.0.10 GPIO_INT_STATUS

Bit Field	Read/Write	Field Name	Description
4	R	GPIO_INT_STATUS	GPIO interrupt status
0	W	GPIO_INT_CLEAR	GPIO interrupt clear

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
4	GPIO_INT_STATUS	GPIO_INT_FALSE	Interrupt not triggered	0x0*
		GPIO_INT_TRUE	Interrupt triggered	0x1
0	GPIO_INT_CLEAR	GPIO_INT_CLEAR	Interrupt clear	0x1

10.5.0.11 GPIO_INT_DEBOUNCE

Bit Field	Read/Write	Field Name	Description
8	RW	DEBOUNCE_CLK	Interrupt button debounce filter clock
7:0	RW	DEBOUNCE_COUNT	Interrupt button debounce filter count

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
8	DEBOUNCE_CLK	GPIO_DEBOUNCE_SLOWCLK_DIV32	Button debounce filter runs on SLOWCLK divided by 32	0x0*
		GPIO_DEBOUNCE_SLOWCLK_DIV1024	Button debounce filter runs on SLOWCLK divided by 1024	0x1

10.5.0.12 GPIO_JTAG_SW_PAD_CFG

Bit Field	Read/Write	Field Name	Description
9	RW	SWCLK_LPF	SWCLK Low-Pass-Filter enable / disable
8	RW	SWDIO_LPF	SWDIO Low-Pass-Filter enable / disable
7	RW	CM33_JTAG_DATA_EN	CM33 JTAG data (TDI and TDO) on GPIO[3:2]
6	RW	CM33_JTAG_TRST_EN	CM33 JTAG TRST on GPIO4
5:4	RW	SWCLK_PULL	SWCLK pull-up enable / disable
3:2	RW	SWDIO_DRIVE	SWDIO drive strength
1:0	RW	SWDIO_PULL	SWDIO pull-up enable / disable

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
9	SWCLK_LPF	SWCLK_LPF_DISABLED	Disable low pass filter	0x0*
		SWCLK_LPF_ENABLED	Enable low pass filter	0x1
8	SWDIO_LPF	SWDIO_LPF_DISABLED	Disable low pass filter	0x0*
		SWDIO_LPF_ENABLED	Enable low pass filter	0x1
7	CM33_JTAG_DATA_EN	CM33_JTAG_DATA_DISABLED	CM33 JTAG data (TDI and TDO) not available on GPIO[3:2]	0x0
		CM33_JTAG_DATA_ENABLED	CM33 JTAG data (TDI and TDO) connected through GPIO[3:2]	0x1*
6	CM33_JTAG_TRST_EN	CM33_JTAG_TRST_DISABLED	CM33 JTAG TRST not available on GPIO4	0x0
		CM33_JTAG_TRST_ENABLED	CM33 JTAG TRST connected through	0x1*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
			GPIO4	
5:4	SWCLK_PULL	SWCLK_NO_PULL	No pull selected	0x0
		SWCLK_WEAK_PULL_UP	Weak pull-up selected	0x1*
		SWCLK_WEAK_PULL_DOWN	Weak pull-down selected	0x2
		SWCLK_STRONG_PULL_UP	Strong pull-up selected	0x3
3:2	SWDIO_DRIVE	SWDIO_2X_DRIVE	2x drive strength	0x0
		SWDIO_3X_DRIVE	3x drive strength	0x1
		SWDIO_5X_DRIVE	5x drive strength	0x2
		SWDIO_6X_DRIVE	6x drive strength	0x3*
1:0	SWDIO_PULL	SWDIO_NO_PULL	No pull selected	0x0
		SWDIO_WEAK_PULL_UP	Weak pull-up selected	0x1*
		SWDIO_WEAK_PULL_DOWN	Weak pull-down selected	0x2
		SWDIO_STRONG_PULL_UP	Strong pull-up selected	0x3

10.5.0.13 GPIO_SRC_SPI

Bit Field	Read/Write	Field Name	Description
12:8	RW	CS	SPI_CS input selection
4:0	RW	CLK	SPI_CLK input selection

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
12:8	CS	SPI_CS_SRC_GPIO_0	Select GPIO[0] as source	0x0
		SPI_CS_SRC_GPIO_1	Select GPIO[1] as source	0x1
		SPI_CS_SRC_GPIO_2	Select GPIO[2] as source	0x2
		SPI_CS_SRC_GPIO_3	Select GPIO[3] as source	0x3
		SPI_CS_SRC_GPIO_4	Select GPIO[4] as source	0x4
		SPI_CS_SRC_GPIO_5	Select GPIO[5] as source	0x5
		SPI_CS_SRC_GPIO_6	Select GPIO[6] as source	0x6
		SPI_CS_SRC_GPIO_7	Select GPIO[7] as source	0x7
		SPI_CS_SRC_GPIO_8	Select GPIO[8] as source	0x8
		SPI_CS_SRC_GPIO_9	Select GPIO[9] as source	0x9

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		SPI_CS_SRC_GPIO_10	Select GPIO[10] as source	0xA
		SPI_CS_SRC_GPIO_11	Select GPIO[11] as source	0xB
		SPI_CS_SRC_GPIO_12	Select GPIO[12] as source	0xC
		SPI_CS_SRC_GPIO_13	Select GPIO[13] as source	0xD
		SPI_CS_SRC_GPIO_14	Select GPIO[14] as source	0xE
		SPI_CS_SRC_GPIO_15	Select GPIO[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*
4:0	CLK	SPI_CLK_SRC_GPIO_0	Select GPIO[0] as source	0x0
		SPI_CLK_SRC_GPIO_1	Select GPIO[1] as source	0x1
		SPI_CLK_SRC_GPIO_2	Select GPIO[2] as source	0x2
		SPI_CLK_SRC_GPIO_3	Select GPIO[3] as source	0x3
		SPI_CLK_SRC_GPIO_4	Select GPIO[4] as source	0x4
		SPI_CLK_SRC_GPIO_5	Select GPIO[5] as source	0x5
		SPI_CLK_SRC_GPIO_6	Select GPIO[6] as source	0x6
		SPI_CLK_SRC_GPIO_7	Select GPIO[7] as source	0x7
		SPI_CLK_SRC_GPIO_8	Select GPIO[8] as source	0x8
		SPI_CLK_SRC_GPIO_9	Select GPIO[9] as source	0x9
		SPI_CLK_SRC_GPIO_10	Select GPIO[10] as source	0xA
		SPI_CLK_SRC_GPIO_11	Select GPIO[11] as source	0xB
		SPI_CLK_SRC_GPIO_12	Select GPIO[12] as source	0xC
		SPI_CLK_SRC_GPIO_13	Select GPIO[13] as source	0xD
		SPI_CLK_SRC_GPIO_14	Select GPIO[14] as source	0xE
		SPI_CLK_SRC_GPIO_15	Select GPIO[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*

10.5.0.14 GPIO_SRC_SPI_IO

Bit Field	Read/Write	Field Name	Description
28:24	RW	IO3	SPI_IO3 input selection
20:16	RW	IO2	SPI_IO2 input selection
12:8	RW	IO1	SPI_IO1 input selection (master SERI)

Bit Field	Read/Write	Field Name	Description
4:0	RW	IO0	SPI_IO0 input selection (slave SERI)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
28:24	IO3	SPI_IO3_SRC_GPIO_0	Select GPIO[0] as source	0x0
		SPI_IO3_SRC_GPIO_1	Select GPIO[1] as source	0x1
		SPI_IO3_SRC_GPIO_2	Select GPIO[2] as source	0x2
		SPI_IO3_SRC_GPIO_3	Select GPIO[3] as source	0x3
		SPI_IO3_SRC_GPIO_4	Select GPIO[4] as source	0x4
		SPI_IO3_SRC_GPIO_5	Select GPIO[5] as source	0x5
		SPI_IO3_SRC_GPIO_6	Select GPIO[6] as source	0x6
		SPI_IO3_SRC_GPIO_7	Select GPIO[7] as source	0x7
		SPI_IO3_SRC_GPIO_8	Select GPIO[8] as source	0x8
		SPI_IO3_SRC_GPIO_9	Select GPIO[9] as source	0x9
		SPI_IO3_SRC_GPIO_10	Select GPIO[10] as source	0xA
		SPI_IO3_SRC_GPIO_11	Select GPIO[11] as source	0xB
		SPI_IO3_SRC_GPIO_12	Select GPIO[12] as source	0xC
		SPI_IO3_SRC_GPIO_13	Select GPIO[13] as source	0xD
		SPI_IO3_SRC_GPIO_14	Select GPIO[14] as source	0xE
		SPI_IO3_SRC_GPIO_15	Select GPIO[15] as source	0xF
		SPI_IO3_SRC_CONST_LOW	Select constant low as source	0x10
		SPI_IO3_SRC_CONST_HIGH	Select constant high as source	0x11*
20:16	IO2	SPI_IO2_SRC_GPIO_0	Select GPIO[0] as source	0x0
		SPI_IO2_SRC_GPIO_1	Select GPIO[1] as source	0x1
		SPI_IO2_SRC_GPIO_2	Select GPIO[2] as source	0x2
		SPI_IO2_SRC_GPIO_3	Select GPIO[3] as source	0x3
		SPI_IO2_SRC_GPIO_4	Select GPIO[4] as source	0x4
		SPI_IO2_SRC_GPIO_5	Select GPIO[5] as source	0x5
		SPI_IO2_SRC_GPIO_6	Select GPIO[6] as source	0x6
		SPI_IO2_SRC_GPIO_7	Select GPIO[7] as source	0x7
		SPI_IO2_SRC_GPIO_8	Select GPIO[8] as source	0x8
		SPI_IO2_SRC_GPIO_9	Select GPIO[9] as source	0x9

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		SPI_IO2_SRC_GPIO_10	Select GPIO[10] as source	0xA
		SPI_IO2_SRC_GPIO_11	Select GPIO[11] as source	0xB
		SPI_IO2_SRC_GPIO_12	Select GPIO[12] as source	0xC
		SPI_IO2_SRC_GPIO_13	Select GPIO[13] as source	0xD
		SPI_IO2_SRC_GPIO_14	Select GPIO[14] as source	0xE
		SPI_IO2_SRC_GPIO_15	Select GPIO[15] as source	0xF
		SPI_IO2_SRC_CONST_LOW	Select constant low as source	0x10
		SPI_IO2_SRC_CONST_HIGH	Select constant high as source	0x11*
12:8	IO1	SPI_IO1_SRC_GPIO_0	Select GPIO[0] as source	0x0
		SPI_IO1_SRC_GPIO_1	Select GPIO[1] as source	0x1
		SPI_IO1_SRC_GPIO_2	Select GPIO[2] as source	0x2
		SPI_IO1_SRC_GPIO_3	Select GPIO[3] as source	0x3
		SPI_IO1_SRC_GPIO_4	Select GPIO[4] as source	0x4
		SPI_IO1_SRC_GPIO_5	Select GPIO[5] as source	0x5
		SPI_IO1_SRC_GPIO_6	Select GPIO[6] as source	0x6
		SPI_IO1_SRC_GPIO_7	Select GPIO[7] as source	0x7
		SPI_IO1_SRC_GPIO_8	Select GPIO[8] as source	0x8
		SPI_IO1_SRC_GPIO_9	Select GPIO[9] as source	0x9
		SPI_IO1_SRC_GPIO_10	Select GPIO[10] as source	0xA
		SPI_IO1_SRC_GPIO_11	Select GPIO[11] as source	0xB
		SPI_IO1_SRC_GPIO_12	Select GPIO[12] as source	0xC
		SPI_IO1_SRC_GPIO_13	Select GPIO[13] as source	0xD
		SPI_IO1_SRC_GPIO_14	Select GPIO[14] as source	0xE
		SPI_IO1_SRC_GPIO_15	Select GPIO[15] as source	0xF
		SPI_IO1_SRC_CONST_LOW	Select constant low as source	0x10
		SPI_IO1_SRC_CONST_HIGH	Select constant high as source	0x11*
4:0	IO0	SPI_IO0_SRC_GPIO_0	Select GPIO[0] as source	0x0
		SPI_IO0_SRC_GPIO_1	Select GPIO[1] as source	0x1
		SPI_IO0_SRC_GPIO_2	Select GPIO[2] as source	0x2
		SPI_IO0_SRC_GPIO_3	Select GPIO[3] as source	0x3
		SPI_IO0_SRC_GPIO_4	Select GPIO[4] as source	0x4

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		SPI_IO0_SRC_GPIO_5	Select GPIO[5] as source	0x5
		SPI_IO0_SRC_GPIO_6	Select GPIO[6] as source	0x6
		SPI_IO0_SRC_GPIO_7	Select GPIO[7] as source	0x7
		SPI_IO0_SRC_GPIO_8	Select GPIO[8] as source	0x8
		SPI_IO0_SRC_GPIO_9	Select GPIO[9] as source	0x9
		SPI_IO0_SRC_GPIO_10	Select GPIO[10] as source	0xA
		SPI_IO0_SRC_GPIO_11	Select GPIO[11] as source	0xB
		SPI_IO0_SRC_GPIO_12	Select GPIO[12] as source	0xC
		SPI_IO0_SRC_GPIO_13	Select GPIO[13] as source	0xD
		SPI_IO0_SRC_GPIO_14	Select GPIO[14] as source	0xE
		SPI_IO0_SRC_GPIO_15	Select GPIO[15] as source	0xF
		SPI_IO0_SRC_CONST_LOW	Select constant low as source	0x10
		SPI_IO0_SRC_CONST_HIGH	Select constant high as source	0x11*

10.5.0.15 GPIO_SRC_UART

Bit Field	Read/Write	Field Name	Description
4:0	RW	RX	UART_RX input selection

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
4:0	RX	UART_RX_SRC_GPIO_0	Select GPIO[0] as source	0x0
		UART_RX_SRC_GPIO_1	Select GPIO[1] as source	0x1
		UART_RX_SRC_GPIO_2	Select GPIO[2] as source	0x2
		UART_RX_SRC_GPIO_3	Select GPIO[3] as source	0x3
		UART_RX_SRC_GPIO_4	Select GPIO[4] as source	0x4
		UART_RX_SRC_GPIO_5	Select GPIO[5] as source	0x5
		UART_RX_SRC_GPIO_6	Select GPIO[6] as source	0x6
		UART_RX_SRC_GPIO_7	Select GPIO[7] as source	0x7
		UART_RX_SRC_GPIO_8	Select GPIO[8] as source	0x8
		UART_RX_SRC_GPIO_9	Select GPIO[9] as source	0x9
		UART_RX_SRC_GPIO_10	Select GPIO[10] as source	0xA
		UART_RX_SRC_GPIO_11	Select GPIO[11] as source	0xB

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		UART_RX_SRC_GPIO_12	Select GPIO[12] as source	0xC
		UART_RX_SRC_GPIO_13	Select GPIO[13] as source	0xD
		UART_RX_SRC_GPIO_14	Select GPIO[14] as source	0xE
		UART_RX_SRC_GPIO_15	Select GPIO[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*

10.5.0.16 GPIO_SRC_I2C

Bit Field	Read/Write	Field Name	Description
12:8	RW	SDA	SDA input selection
4:0	RW	SCL	SCL input selection

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
12:8	SDA	I2C_SDA_SRC_GPIO_0	Select GPIO[0] as source	0x0
		I2C_SDA_SRC_GPIO_1	Select GPIO[1] as source	0x1
		I2C_SDA_SRC_GPIO_2	Select GPIO[2] as source	0x2
		I2C_SDA_SRC_GPIO_3	Select GPIO[3] as source	0x3
		I2C_SDA_SRC_GPIO_4	Select GPIO[4] as source	0x4
		I2C_SDA_SRC_GPIO_5	Select GPIO[5] as source	0x5
		I2C_SDA_SRC_GPIO_6	Select GPIO[6] as source	0x6
		I2C_SDA_SRC_GPIO_7	Select GPIO[7] as source	0x7
		I2C_SDA_SRC_GPIO_8	Select GPIO[8] as source	0x8
		I2C_SDA_SRC_GPIO_9	Select GPIO[9] as source	0x9
		I2C_SDA_SRC_GPIO_10	Select GPIO[10] as source	0xA
		I2C_SDA_SRC_GPIO_11	Select GPIO[11] as source	0xB
		I2C_SDA_SRC_GPIO_12	Select GPIO[12] as source	0xC
		I2C_SDA_SRC_GPIO_13	Select GPIO[13] as source	0xD
		I2C_SDA_SRC_GPIO_14	Select GPIO[14] as source	0xE
		I2C_SDA_SRC_GPIO_15	Select GPIO[15] as source	0xF
		I2C_SDA_SRC_CONST_LOW	Select constant low as source	0x10
		I2C_SDA_SRC_CONST_HIGH	Select constant high as source	0x11*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
4:0	SCL	I2C_SCL_SRC_GPIO_0	Select GPIO[0] as source	0x0
		I2C_SCL_SRC_GPIO_1	Select GPIO[1] as source	0x1
		I2C_SCL_SRC_GPIO_2	Select GPIO[2] as source	0x2
		I2C_SCL_SRC_GPIO_3	Select GPIO[3] as source	0x3
		I2C_SCL_SRC_GPIO_4	Select GPIO[4] as source	0x4
		I2C_SCL_SRC_GPIO_5	Select GPIO[5] as source	0x5
		I2C_SCL_SRC_GPIO_6	Select GPIO[6] as source	0x6
		I2C_SCL_SRC_GPIO_7	Select GPIO[7] as source	0x7
		I2C_SCL_SRC_GPIO_8	Select GPIO[8] as source	0x8
		I2C_SCL_SRC_GPIO_9	Select GPIO[9] as source	0x9
		I2C_SCL_SRC_GPIO_10	Select GPIO[10] as source	0xA
		I2C_SCL_SRC_GPIO_11	Select GPIO[11] as source	0xB
		I2C_SCL_SRC_GPIO_12	Select GPIO[12] as source	0xC
		I2C_SCL_SRC_GPIO_13	Select GPIO[13] as source	0xD
		I2C_SCL_SRC_GPIO_14	Select GPIO[14] as source	0xE
		I2C_SCL_SRC_GPIO_15	Select GPIO[15] as source	0xF
		I2C_SCL_SRC_CONST_LOW	Select constant low as source	0x10
		I2C_SCL_SRC_CONST_HIGH	Select constant high as source	0x11*

10.5.0.17 GPIO_SRC_PCM

Bit Field	Read/Write	Field Name	Description
20:16	RW	SERI	PCM_SERI input selection
12:8	RW	FRAME	PCM_FRAME input selection
4:0	RW	CLK	PCM_CLK input selection

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
20:16	SERI	PCM_SERI_SRC_GPIO_0	Select GPIO[0] as source	0x0
		PCM_SERI_SRC_GPIO_1	Select GPIO[1] as source	0x1
		PCM_SERI_SRC_GPIO_2	Select GPIO[2] as source	0x2
		PCM_SERI_SRC_GPIO_3	Select GPIO[3] as source	0x3
		PCM_SERI_SRC_GPIO_4	Select GPIO[4] as source	0x4

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		PCM_SERI_SRC_GPIO_5	Select GPIO[5] as source	0x5
		PCM_SERI_SRC_GPIO_6	Select GPIO[6] as source	0x6
		PCM_SERI_SRC_GPIO_7	Select GPIO[7] as source	0x7
		PCM_SERI_SRC_GPIO_8	Select GPIO[8] as source	0x8
		PCM_SERI_SRC_GPIO_9	Select GPIO[9] as source	0x9
		PCM_SERI_SRC_GPIO_10	Select GPIO[10] as source	0xA
		PCM_SERI_SRC_GPIO_11	Select GPIO[11] as source	0xB
		PCM_SERI_SRC_GPIO_12	Select GPIO[12] as source	0xC
		PCM_SERI_SRC_GPIO_13	Select GPIO[13] as source	0xD
		PCM_SERI_SRC_GPIO_14	Select GPIO[14] as source	0xE
		PCM_SERI_SRC_GPIO_15	Select GPIO[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*
12:8	FRAME	PCM_FRAME_SRC_GPIO_0	Select GPIO[0] as source	0x0
		PCM_FRAME_SRC_GPIO_1	Select GPIO[1] as source	0x1
		PCM_FRAME_SRC_GPIO_2	Select GPIO[2] as source	0x2
		PCM_FRAME_SRC_GPIO_3	Select GPIO[3] as source	0x3
		PCM_FRAME_SRC_GPIO_4	Select GPIO[4] as source	0x4
		PCM_FRAME_SRC_GPIO_5	Select GPIO[5] as source	0x5
		PCM_FRAME_SRC_GPIO_6	Select GPIO[6] as source	0x6
		PCM_FRAME_SRC_GPIO_7	Select GPIO[7] as source	0x7
		PCM_FRAME_SRC_GPIO_8	Select GPIO[8] as source	0x8
		PCM_FRAME_SRC_GPIO_9	Select GPIO[9] as source	0x9
		PCM_FRAME_SRC_GPIO_10	Select GPIO[10] as source	0xA
		PCM_FRAME_SRC_GPIO_11	Select GPIO[11] as source	0xB
		PCM_FRAME_SRC_GPIO_12	Select GPIO[12] as source	0xC
		PCM_FRAME_SRC_GPIO_13	Select GPIO[13] as source	0xD
		PCM_FRAME_SRC_GPIO_14	Select GPIO[14] as source	0xE
		PCM_FRAME_SRC_GPIO_15	Select GPIO[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*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
4:0	CLK	PCM_CLK_SRC_GPIO_0	Select GPIO[0] as source	0x0
		PCM_CLK_SRC_GPIO_1	Select GPIO[1] as source	0x1
		PCM_CLK_SRC_GPIO_2	Select GPIO[2] as source	0x2
		PCM_CLK_SRC_GPIO_3	Select GPIO[3] as source	0x3
		PCM_CLK_SRC_GPIO_4	Select GPIO[4] as source	0x4
		PCM_CLK_SRC_GPIO_5	Select GPIO[5] as source	0x5
		PCM_CLK_SRC_GPIO_6	Select GPIO[6] as source	0x6
		PCM_CLK_SRC_GPIO_7	Select GPIO[7] as source	0x7
		PCM_CLK_SRC_GPIO_8	Select GPIO[8] as source	0x8
		PCM_CLK_SRC_GPIO_9	Select GPIO[9] as source	0x9
		PCM_CLK_SRC_GPIO_10	Select GPIO[10] as source	0xA
		PCM_CLK_SRC_GPIO_11	Select GPIO[11] as source	0xB
		PCM_CLK_SRC_GPIO_12	Select GPIO[12] as source	0xC
		PCM_CLK_SRC_GPIO_13	Select GPIO[13] as source	0xD
		PCM_CLK_SRC_GPIO_14	Select GPIO[14] as source	0xE
		PCM_CLK_SRC_GPIO_15	Select GPIO[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*

10.5.0.18 GPIO_SRC_LIN

Bit Field	Read/Write	Field Name	Description
5	RW	LIN_POLARITY	LIN polarity
4:0	RW	LIN	LIN input selection

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
5	LIN_POLARITY	LIN_ACTIVE_LOW	LIN active low	0x0
		LIN_ACTIVE_HIGH	LIN active high	0x1*
4:0	LIN	LIN_RX_SRC_GPIO_0	Select GPIO[0] as source	0x0
		LIN_RX_SRC_GPIO_1	Select GPIO[1] as source	0x1
		LIN_RX_SRC_GPIO_2	Select GPIO[2] as source	0x2
		LIN_RX_SRC_GPIO_3	Select GPIO[3] as source	0x3

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		LIN_RX_SRC_GPIO_4	Select GPIO[4] as source	0x4
		LIN_RX_SRC_GPIO_5	Select GPIO[5] as source	0x5
		LIN_RX_SRC_GPIO_6	Select GPIO[6] as source	0x6
		LIN_RX_SRC_GPIO_7	Select GPIO[7] as source	0x7
		LIN_RX_SRC_GPIO_8	Select GPIO[8] as source	0x8
		LIN_RX_SRC_GPIO_9	Select GPIO[9] as source	0x9
		LIN_RX_SRC_GPIO_10	Select GPIO[10] as source	0xA
		LIN_RX_SRC_GPIO_11	Select GPIO[11] as source	0xB
		LIN_RX_SRC_GPIO_12	Select GPIO[12] as source	0xC
		LIN_RX_SRC_GPIO_13	Select GPIO[13] as source	0xD
		LIN_RX_SRC_GPIO_14	Select GPIO[14] as source	0xE
		LIN_RX_SRC_GPIO_15	Select GPIO[15] as source	0xF
		LIN_RX_SRC_CONST_LOW	Select constant low as source	0x10*
		LIN_RX_SRC_CONST_HIGH	Select constant high as source	0x11

10.5.0.19 GPIO_SRC_NMI

Bit Field	Read/Write	Field Name	Description
5	RW	NMI_POLARITY	NMI polarity
4:0	RW	NMI	NMI input selection

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
5	NMI_POLARITY	NMI_ACTIVE_LOW	NMI active low	0x0
		NMI_ACTIVE_HIGH	NMI active high	0x1*
4:0	NMI	NMI_SRC_GPIO_0	Select GPIO[0] as source	0x0
		NMI_SRC_GPIO_1	Select GPIO[1] as source	0x1
		NMI_SRC_GPIO_2	Select GPIO[2] as source	0x2
		NMI_SRC_GPIO_3	Select GPIO[3] as source	0x3
		NMI_SRC_GPIO_4	Select GPIO[4] as source	0x4
		NMI_SRC_GPIO_5	Select GPIO[5] as source	0x5
		NMI_SRC_GPIO_6	Select GPIO[6] as source	0x6
		NMI_SRC_GPIO_7	Select GPIO[7] as source	0x7

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		NMI_SRC_GPIO_8	Select GPIO[8] as source	0x8
		NMI_SRC_GPIO_9	Select GPIO[9] as source	0x9
		NMI_SRC_GPIO_10	Select GPIO[10] as source	0xA
		NMI_SRC_GPIO_11	Select GPIO[11] as source	0xB
		NMI_SRC_GPIO_12	Select GPIO[12] as source	0xC
		NMI_SRC_GPIO_13	Select GPIO[13] as source	0xD
		NMI_SRC_GPIO_14	Select GPIO[14] as source	0xE
		NMI_SRC_GPIO_15	Select GPIO[15] as source	0xF
		NMI_SRC_CONST_LOW	Select constant low as source	0x10*
		NMI_SRC_CONST_HIGH	Select constant high as source	0x11

10.5.0.20 GPIO_SRC_BB_RX

Bit Field	Read/Write	Field Name	Description
20:16	RW	SYNC_P	Baseband controller interface SYNC_P input selection
12:8	RW	CLK	Baseband controller RX clock input selection
4:0	RW	DATA	Baseband controller RX data input selection

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
20:16	SYNC_P	BB_RX_SYNC_P_SRC_GPIO_0	Select GPIO[0] as source	0x0
		BB_RX_SYNC_P_SRC_GPIO_1	Select GPIO[1] as source	0x1
		BB_RX_SYNC_P_SRC_GPIO_2	Select GPIO[5] as source	0x2
		BB_RX_SYNC_P_SRC_GPIO_3	Select GPIO[3] as source	0x3
		BB_RX_SYNC_P_SRC_GPIO_4	Select GPIO[4] as source	0x4
		BB_RX_SYNC_P_SRC_GPIO_5	Select GPIO[5] as source	0x5
		BB_RX_SYNC_P_SRC_GPIO_6	Select GPIO[6] as source	0x6
		BB_RX_SYNC_P_SRC_GPIO_7	Select GPIO[7] as source	0x7
		BB_RX_SYNC_P_SRC_GPIO_8	Select GPIO[8] as source	0x8
		BB_RX_SYNC_P_SRC_GPIO_9	Select GPIO[9] as source	0x9
		BB_RX_SYNC_P_SRC_GPIO_10	Select GPIO[10] as source	0xA
		BB_RX_SYNC_P_SRC_GPIO_11	Select GPIO[11] as source	0xB

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		BB_RX_SYNC_P_SRC_GPIO_12	Select GPIO[12] as source	0xC
		BB_RX_SYNC_P_SRC_GPIO_13	Select GPIO[13] as source	0xD
		BB_RX_SYNC_P_SRC_GPIO_14	Select GPIO[14] as source	0xE
		BB_RX_SYNC_P_SRC_GPIO_15	Select GPIO[15] as source	0xF
		BB_RX_SYNC_P_SRC_CONST_LOW	Select constant low as source	0x10
		BB_RX_SYNC_P_SRC_CONST_HIGH	Select constant high as source	0x11
		BB_RX_SYNC_P_SRC_RF_GPIO2	Select RF front-end GPIO2 output (RX_SYNC_P) as source	0x12*
12:8	CLK	BB_RX_CLK_SRC_GPIO_0	Select GPIO[0] as source	0x0
		BB_RX_CLK_SRC_GPIO_1	Select GPIO[1] as source	0x1
		BB_RX_CLK_SRC_GPIO_2	Select GPIO[2] as source	0x2
		BB_RX_CLK_SRC_GPIO_3	Select GPIO[3] as source	0x3
		BB_RX_CLK_SRC_GPIO_4	Select GPIO[4] as source	0x4
		BB_RX_CLK_SRC_GPIO_5	Select GPIO[5] as source	0x5
		BB_RX_CLK_SRC_GPIO_6	Select GPIO[6] as source	0x6
		BB_RX_CLK_SRC_GPIO_7	Select GPIO[7] as source	0x7
		BB_RX_CLK_SRC_GPIO_8	Select GPIO[8] as source	0x8
		BB_RX_CLK_SRC_GPIO_9	Select GPIO[9] as source	0x9
		BB_RX_CLK_SRC_GPIO_10	Select GPIO[10] as source	0xA
		BB_RX_CLK_SRC_GPIO_11	Select GPIO[11] as source	0xB
		BB_RX_CLK_SRC_GPIO_12	Select GPIO[12] as source	0xC
		BB_RX_CLK_SRC_GPIO_13	Select GPIO[13] as source	0xD
		BB_RX_CLK_SRC_GPIO_14	Select GPIO[14] as source	0xE
		BB_RX_CLK_SRC_GPIO_15	Select GPIO[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*
4:0	DATA	BB_RX_DATA_SRC_GPIO_0	Select GPIO[0] as source	0x0
		BB_RX_DATA_SRC_GPIO_1	Select GPIO[1] as source	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		BB_RX_DATA_SRC_GPIO_2	Select GPIO[2] as source	0x2
		BB_RX_DATA_SRC_GPIO_3	Select GPIO[3] as source	0x3
		BB_RX_DATA_SRC_GPIO_4	Select GPIO[4] as source	0x4
		BB_RX_DATA_SRC_GPIO_5	Select GPIO[5] as source	0x5
		BB_RX_DATA_SRC_GPIO_6	Select GPIO[6] as source	0x6
		BB_RX_DATA_SRC_GPIO_7	Select GPIO[7] as source	0x7
		BB_RX_DATA_SRC_GPIO_8	Select GPIO[8] as source	0x8
		BB_RX_DATA_SRC_GPIO_9	Select GPIO[9] as source	0x9
		BB_RX_DATA_SRC_GPIO_10	Select GPIO[10] as source	0xA
		BB_RX_DATA_SRC_GPIO_11	Select GPIO[11] as source	0xB
		BB_RX_DATA_SRC_GPIO_12	Select GPIO[12] as source	0xC
		BB_RX_DATA_SRC_GPIO_13	Select GPIO[13] as source	0xD
		BB_RX_DATA_SRC_GPIO_14	Select GPIO[14] as source	0xE
		BB_RX_DATA_SRC_GPIO_15	Select GPIO[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*

10.5.0.21 GPIO_SRC_BB_SPI

Bit Field	Read/Write	Field Name	Description
4:0	RW	MISO	Baseband controller SPI_MISO input selection

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
4:0	MISO	BB_SPI_MISO_SRC_GPIO_0	Select GPIO[0] as source	0x0
		BB_SPI_MISO_SRC_GPIO_1	Select GPIO[1] as source	0x1
		BB_SPI_MISO_SRC_GPIO_2	Select GPIO[2] as source	0x2
		BB_SPI_MISO_SRC_GPIO_3	Select GPIO[3] as source	0x3
		BB_SPI_MISO_SRC_GPIO_4	Select GPIO[4] as source	0x4
		BB_SPI_MISO_SRC_GPIO_5	Select GPIO[5] as source	0x5
		BB_SPI_MISO_SRC_GPIO_6	Select GPIO[6] as source	0x6

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		BB_SPI_MISO_SRC_GPIO_7	Select GPIO[7] as source	0x7
		BB_SPI_MISO_SRC_GPIO_8	Select GPIO[8] as source	0x8
		BB_SPI_MISO_SRC_GPIO_9	Select GPIO[9] as source	0x9
		BB_SPI_MISO_SRC_GPIO_10	Select GPIO[10] as source	0xA
		BB_SPI_MISO_SRC_GPIO_11	Select GPIO[11] as source	0xB
		BB_SPI_MISO_SRC_GPIO_12	Select GPIO[12] as source	0xC
		BB_SPI_MISO_SRC_GPIO_13	Select GPIO[13] as source	0xD
		BB_SPI_MISO_SRC_GPIO_14	Select GPIO[14] as source	0xE
		BB_SPI_MISO_SRC_GPIO_15	Select GPIO[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*

10.5.0.22 GPIO_SRC_BB_COEX

Bit Field	Read/Write	Field Name	Description
12:8	RW	WLAN_RX	Baseband controller WLAN_RX input selection
4:0	RW	WLAN_TX	Baseband controller WLAN_TX input selection

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
12:8	WLAN_RX	BB_WLAN_RX_SRC_GPIO_0	Select GPIO[0] as source	0x0
		BB_WLAN_RX_SRC_GPIO_1	Select GPIO[1] as source	0x1
		BB_WLAN_RX_SRC_GPIO_2	Select GPIO[2] as source	0x2
		BB_WLAN_RX_SRC_GPIO_3	Select GPIO[3] as source	0x3
		BB_WLAN_RX_SRC_GPIO_4	Select GPIO[4] as source	0x4
		BB_WLAN_RX_SRC_GPIO_5	Select GPIO[5] as source	0x5
		BB_WLAN_RX_SRC_GPIO_6	Select GPIO[6] as source	0x6
		BB_WLAN_RX_SRC_GPIO_7	Select GPIO[7] as source	0x7
		BB_WLAN_RX_SRC_GPIO_8	Select GPIO[8] as source	0x8
		BB_WLAN_RX_SRC_GPIO_9	Select GPIO[9] as source	0x9
		BB_WLAN_RX_SRC_GPIO_10	Select GPIO[10] as source	0xA

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		BB_WLAN_RX_SRC_GPIO_11	Select GPIO[11] as source	0xB
		BB_WLAN_RX_SRC_GPIO_12	Select GPIO[12] as source	0xC
		BB_WLAN_RX_SRC_GPIO_13	Select GPIO[13] as source	0xD
		BB_WLAN_RX_SRC_GPIO_14	Select GPIO[14] as source	0xE
		BB_WLAN_RX_SRC_GPIO_15	Select GPIO[15] as source	0xF
		BB_WLAN_RX_SRC_CONST_LOW	Select constant low as source	0x10*
		BB_WLAN_RX_SRC_CONST_HIGH	Select constant high as source	0x11
4:0	WLAN_TX	BB_WLAN_TX_SRC_GPIO_0	Select GPIO[0] as source	0x0
		BB_WLAN_TX_SRC_GPIO_1	Select GPIO[1] as source	0x1
		BB_WLAN_TX_SRC_GPIO_2	Select GPIO[2] as source	0x2
		BB_WLAN_TX_SRC_GPIO_3	Select GPIO[3] as source	0x3
		BB_WLAN_TX_SRC_GPIO_4	Select GPIO[4] as source	0x4
		BB_WLAN_TX_SRC_GPIO_5	Select GPIO[5] as source	0x5
		BB_WLAN_TX_SRC_GPIO_6	Select GPIO[6] as source	0x6
		BB_WLAN_TX_SRC_GPIO_7	Select GPIO[7] as source	0x7
		BB_WLAN_TX_SRC_GPIO_8	Select GPIO[8] as source	0x8
		BB_WLAN_TX_SRC_GPIO_9	Select GPIO[9] as source	0x9
		BB_WLAN_TX_SRC_GPIO_10	Select GPIO[10] as source	0xA
		BB_WLAN_TX_SRC_GPIO_11	Select GPIO[11] as source	0xB
		BB_WLAN_TX_SRC_GPIO_12	Select GPIO[12] as source	0xC
		BB_WLAN_TX_SRC_GPIO_13	Select GPIO[13] as source	0xD
		BB_WLAN_TX_SRC_GPIO_14	Select GPIO[14] as source	0xE
		BB_WLAN_TX_SRC_GPIO_15	Select GPIO[15] as source	0xF
		BB_WLAN_TX_SRC_CONST_LOW	Select constant low as source	0x10*
		BB_WLAN_TX_SRC_CONST_HIGH	Select constant high as source	0x11

10.5.0.23 GPIO_SRC_BB_IQ_DATA

Bit Field	Read/Write	Field Name	Description
28:24	RW	IQ_DATA_3	Baseband controller IQ_DATA_3 input selection
20:16	RW	IQ_DATA_2	Baseband controller IQ_DATA_2 input selection
12:8	RW	IQ_DATA_1	Baseband controller IQ_DATA_1 input selection
4:0	RW	IQ_DATA_0	Baseband controller IQ_DATA_0 input selection

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
28:24	IQ_DATA_3	BB_IQ_DATA_3_SRC_GPIO_0	Select GPIO[0] as source	0x0
		BB_IQ_DATA_3_SRC_GPIO_1	Select GPIO[1] as source	0x1
		BB_IQ_DATA_3_SRC_GPIO_2	Select GPIO[2] as source	0x2
		BB_IQ_DATA_3_SRC_GPIO_3	Select GPIO[3] as source	0x3
		BB_IQ_DATA_3_SRC_GPIO_4	Select GPIO[4] as source	0x4
		BB_IQ_DATA_3_SRC_GPIO_5	Select GPIO[5] as source	0x5
		BB_IQ_DATA_3_SRC_GPIO_6	Select GPIO[6] as source	0x6
		BB_IQ_DATA_3_SRC_GPIO_7	Select GPIO[7] as source	0x7
		BB_IQ_DATA_3_SRC_GPIO_8	Select GPIO[8] as source	0x8
		BB_IQ_DATA_3_SRC_GPIO_9	Select GPIO[9] as source	0x9
		BB_IQ_DATA_3_SRC_GPIO_10	Select GPIO[10] as source	0xA
		BB_IQ_DATA_3_SRC_GPIO_11	Select GPIO[11] as source	0xB
		BB_IQ_DATA_3_SRC_GPIO_12	Select GPIO[12] as source	0xC
		BB_IQ_DATA_3_SRC_GPIO_13	Select GPIO[13] as source	0xD
		BB_IQ_DATA_3_SRC_GPIO_14	Select GPIO[14] as source	0xE
		BB_IQ_DATA_3_SRC_GPIO_15	Select GPIO[15] as source	0xF
		BB_IQ_DATA_3_SRC_CONST_LOW	Select constant low as source	0x10
		BB_IQ_DATA_3_SRC_CONST_HIGH	Select constant high as source	0x11
		BB_IQ_DATA_3_SRC_RF_IQ_DATA_3	Select RF front-end IQ_DATA_3 as source	0x12*
20:16	IQ_DATA_2	BB_IQ_DATA_2_SRC_GPIO_0	Select GPIO[0] as source	0x0
		BB_IQ_DATA_2_SRC_GPIO_1	Select GPIO[1] as source	0x1
		BB_IQ_DATA_2_SRC_GPIO_2	Select GPIO[2] as source	0x2
		BB_IQ_DATA_2_SRC_GPIO_3	Select GPIO[3] as source	0x3
		BB_IQ_DATA_2_SRC_GPIO_4	Select GPIO[4] as source	0x4
		BB_IQ_DATA_2_SRC_GPIO_5	Select GPIO[5] as source	0x5
		BB_IQ_DATA_2_SRC_GPIO_6	Select GPIO[6] as source	0x6
		BB_IQ_DATA_2_SRC_GPIO_7	Select GPIO[7] as source	0x7
		BB_IQ_DATA_2_SRC_GPIO_8	Select GPIO[8] as source	0x8
		BB_IQ_DATA_2_SRC_GPIO_9	Select GPIO[9] as source	0x9

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		BB_IQ_DATA_2_SRC_GPIO_10	Select GPIO[10] as source	0xA
		BB_IQ_DATA_2_SRC_GPIO_11	Select GPIO[11] as source	0xB
		BB_IQ_DATA_2_SRC_GPIO_12	Select GPIO[12] as source	0xC
		BB_IQ_DATA_2_SRC_GPIO_13	Select GPIO[13] as source	0xD
		BB_IQ_DATA_2_SRC_GPIO_14	Select GPIO[14] as source	0xE
		BB_IQ_DATA_2_SRC_GPIO_15	Select GPIO[15] as source	0xF
		BB_IQ_DATA_2_SRC_CONST_LOW	Select constant low as source	0x10
		BB_IQ_DATA_2_SRC_CONST_HIGH	Select constant high as source	0x11
		BB_IQ_DATA_2_SRC_RF_IQ_DATA_2	Select RF front-end IQ_DATA_2 as source	0x12*
12:8	IQ_DATA_1	BB_IQ_DATA_1_SRC_GPIO_0	Select GPIO[0] as source	0x0
		BB_IQ_DATA_1_SRC_GPIO_1	Select GPIO[1] as source	0x1
		BB_IQ_DATA_1_SRC_GPIO_2	Select GPIO[2] as source	0x2
		BB_IQ_DATA_1_SRC_GPIO_3	Select GPIO[3] as source	0x3
		BB_IQ_DATA_1_SRC_GPIO_4	Select GPIO[4] as source	0x4
		BB_IQ_DATA_1_SRC_GPIO_5	Select GPIO[5] as source	0x5
		BB_IQ_DATA_1_SRC_GPIO_6	Select GPIO[6] as source	0x6
		BB_IQ_DATA_1_SRC_GPIO_7	Select GPIO[7] as source	0x7
		BB_IQ_DATA_1_SRC_GPIO_8	Select GPIO[8] as source	0x8
		BB_IQ_DATA_1_SRC_GPIO_9	Select GPIO[9] as source	0x9
		BB_IQ_DATA_1_SRC_GPIO_10	Select GPIO[10] as source	0xA
		BB_IQ_DATA_1_SRC_GPIO_11	Select GPIO[11] as source	0xB
		BB_IQ_DATA_1_SRC_GPIO_12	Select GPIO[12] as source	0xC
		BB_IQ_DATA_1_SRC_GPIO_13	Select GPIO[13] as source	0xD
		BB_IQ_DATA_1_SRC_GPIO_14	Select GPIO[14] as source	0xE
		BB_IQ_DATA_1_SRC_GPIO_15	Select GPIO[15] as source	0xF
		BB_IQ_DATA_1_SRC_CONST_LOW	Select constant low as source	0x10
		BB_IQ_DATA_1_SRC_CONST_HIGH	Select constant high as source	0x11
		BB_IQ_DATA_1_SRC_RF_IQ_	Select RF front-end IQ_DATA_1 as	0x12*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		DATA_1	source	
4:0	IQ_DATA_0	BB_IQ_DATA_0_SRC_GPIO_0	Select GPIO[0] as source	0x0
		BB_IQ_DATA_0_SRC_GPIO_1	Select GPIO[1] as source	0x1
		BB_IQ_DATA_0_SRC_GPIO_2	Select GPIO[2] as source	0x2
		BB_IQ_DATA_0_SRC_GPIO_3	Select GPIO[3] as source	0x3
		BB_IQ_DATA_0_SRC_GPIO_4	Select GPIO[4] as source	0x4
		BB_IQ_DATA_0_SRC_GPIO_5	Select GPIO[5] as source	0x5
		BB_IQ_DATA_0_SRC_GPIO_6	Select GPIO[6] as source	0x6
		BB_IQ_DATA_0_SRC_GPIO_7	Select GPIO[7] as source	0x7
		BB_IQ_DATA_0_SRC_GPIO_8	Select GPIO[8] as source	0x8
		BB_IQ_DATA_0_SRC_GPIO_9	Select GPIO[9] as source	0x9
		BB_IQ_DATA_0_SRC_GPIO_10	Select GPIO[10] as source	0xA
		BB_IQ_DATA_0_SRC_GPIO_11	Select GPIO[11] as source	0xB
		BB_IQ_DATA_0_SRC_GPIO_12	Select GPIO[12] as source	0xC
		BB_IQ_DATA_0_SRC_GPIO_13	Select GPIO[13] as source	0xD
		BB_IQ_DATA_0_SRC_GPIO_14	Select GPIO[14] as source	0xE
		BB_IQ_DATA_0_SRC_GPIO_15	Select GPIO[15] as source	0xF
		BB_IQ_DATA_0_SRC_CONST_LOW	Select constant low as source	0x10
		BB_IQ_DATA_0_SRC_CONST_HIGH	Select constant high as source	0x11
		BB_IQ_DATA_0_SRC_RF_IQ_DATA_0	Select RF front-end IQ_DATA_0 as source	0x12*

10.5.0.24 GPIO_SRC_BB_IQ_DATA_P

Bit Field	Read/Write	Field Name	Description
4:0	RW	IQ_DATA_P	Baseband controller IQ_DATA_P input selection

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
4:0	IQ_DATA_P	BB_IQ_DATA_P_SRC_GPIO_0	Select GPIO[0] as source	0x0
		BB_IQ_DATA_P_SRC_GPIO_1	Select GPIO[1] as source	0x1
		BB_IQ_DATA_P_SRC_GPIO_2	Select GPIO[2] as source	0x2

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		BB_IQ_DATA_P_SRC_GPIO_3	Select GPIO[3] as source	0x3
		BB_IQ_DATA_P_SRC_GPIO_4	Select GPIO[4] as source	0x4
		BB_IQ_DATA_P_SRC_GPIO_5	Select GPIO[5] as source	0x5
		BB_IQ_DATA_P_SRC_GPIO_6	Select GPIO[6] as source	0x6
		BB_IQ_DATA_P_SRC_GPIO_7	Select GPIO[7] as source	0x7
		BB_IQ_DATA_P_SRC_GPIO_8	Select GPIO[8] as source	0x8
		BB_IQ_DATA_P_SRC_GPIO_9	Select GPIO[9] as source	0x9
		BB_IQ_DATA_P_SRC_GPIO_10	Select GPIO[10] as source	0xA
		BB_IQ_DATA_P_SRC_GPIO_11	Select GPIO[11] as source	0xB
		BB_IQ_DATA_P_SRC_GPIO_12	Select GPIO[12] as source	0xC
		BB_IQ_DATA_P_SRC_GPIO_13	Select GPIO[13] as source	0xD
		BB_IQ_DATA_P_SRC_GPIO_14	Select GPIO[14] as source	0xE
		BB_IQ_DATA_P_SRC_GPIO_15	Select GPIO[15] as source	0xF
		BB_IQ_DATA_P_SRC_CONST_LOW	Select constant low as source	0x10
		BB_IQ_DATA_P_SRC_CONST_HIGH	Select constant high as source	0x11
		BB_IQ_DATA_P_SRC_RF_IQ_DATA_P	Select RF front-end IQ_DATA_P as source	0x12*

10.5.0.25 GPIO_SRC_RF_SPI

Bit Field	Read/Write	Field Name	Description
20:16	RW	MOSI	RF front-end SPI_MOSI input selection
12:8	RW	CSN	RF front-end SPI_CSN input selection
4:0	RW	CLK	RF front-end SPI_CLK input selection

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
20:16	MOSI	RF_SPI_MOSI_SRC_GPIO_0	Select GPIO[0] as source	0x0
		RF_SPI_MOSI_SRC_GPIO_1	Select GPIO[1] as source	0x1
		RF_SPI_MOSI_SRC_GPIO_2	Select GPIO[2] as source	0x2
		RF_SPI_MOSI_SRC_GPIO_3	Select GPIO[3] as source	0x3
		RF_SPI_MOSI_SRC_GPIO_4	Select GPIO[4] as source	0x4

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		RF_SPI_MOSI_SRC_GPIO_5	Select GPIO[5] as source	0x5
		RF_SPI_MOSI_SRC_GPIO_6	Select GPIO[6] as source	0x6
		RF_SPI_MOSI_SRC_GPIO_7	Select GPIO[7] as source	0x7
		RF_SPI_MOSI_SRC_GPIO_8	Select GPIO[8] as source	0x8
		RF_SPI_MOSI_SRC_GPIO_9	Select GPIO[9] as source	0x9
		RF_SPI_MOSI_SRC_GPIO_10	Select GPIO[10] as source	0xA
		RF_SPI_MOSI_SRC_GPIO_11	Select GPIO[11] as source	0xB
		RF_SPI_MOSI_SRC_GPIO_12	Select GPIO[12] as source	0xC
		RF_SPI_MOSI_SRC_GPIO_13	Select GPIO[13] as source	0xD
		RF_SPI_MOSI_SRC_GPIO_14	Select GPIO[14] as source	0xE
		RF_SPI_MOSI_SRC_GPIO_15	Select GPIO[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*
12:8	CSN	RF_SPI_CSN_SRC_GPIO_0	Select GPIO[0] as source	0x0
		RF_SPI_CSN_SRC_GPIO_1	Select GPIO[1] as source	0x1
		RF_SPI_CSN_SRC_GPIO_2	Select GPIO[2] as source	0x2
		RF_SPI_CSN_SRC_GPIO_3	Select GPIO[3] as source	0x3
		RF_SPI_CSN_SRC_GPIO_4	Select GPIO[4] as source	0x4
		RF_SPI_CSN_SRC_GPIO_5	Select GPIO[5] as source	0x5
		RF_SPI_CSN_SRC_GPIO_6	Select GPIO[6] as source	0x6
		RF_SPI_CSN_SRC_GPIO_7	Select GPIO[7] as source	0x7
		RF_SPI_CSN_SRC_GPIO_8	Select GPIO[8] as source	0x8
		RF_SPI_CSN_SRC_GPIO_9	Select GPIO[9] as source	0x9
		RF_SPI_CSN_SRC_GPIO_10	Select GPIO[10] as source	0xA
		RF_SPI_CSN_SRC_GPIO_11	Select GPIO[11] as source	0xB
		RF_SPI_CSN_SRC_GPIO_12	Select GPIO[12] as source	0xC
		RF_SPI_CSN_SRC_GPIO_13	Select GPIO[13] as source	0xD
		RF_SPI_CSN_SRC_GPIO_14	Select GPIO[14] as source	0xE
		RF_SPI_CSN_SRC_GPIO_15	Select GPIO[15] as source	0xF

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		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*
4:0	CLK	RF_SPI_CLK_SRC_GPIO_0	Select GPIO[0] as source	0x0
		RF_SPI_CLK_SRC_GPIO_1	Select GPIO[1] as source	0x1
		RF_SPI_CLK_SRC_GPIO_2	Select GPIO[2] as source	0x2
		RF_SPI_CLK_SRC_GPIO_3	Select GPIO[3] as source	0x3
		RF_SPI_CLK_SRC_GPIO_4	Select GPIO[4] as source	0x4
		RF_SPI_CLK_SRC_GPIO_5	Select GPIO[5] as source	0x5
		RF_SPI_CLK_SRC_GPIO_6	Select GPIO[6] as source	0x6
		RF_SPI_CLK_SRC_GPIO_7	Select GPIO[7] as source	0x7
		RF_SPI_CLK_SRC_GPIO_8	Select GPIO[8] as source	0x8
		RF_SPI_CLK_SRC_GPIO_9	Select GPIO[9] as source	0x9
		RF_SPI_CLK_SRC_GPIO_10	Select GPIO[10] as source	0xA
		RF_SPI_CLK_SRC_GPIO_11	Select GPIO[11] as source	0xB
		RF_SPI_CLK_SRC_GPIO_12	Select GPIO[12] as source	0xC
		RF_SPI_CLK_SRC_GPIO_13	Select GPIO[13] as source	0xD
		RF_SPI_CLK_SRC_GPIO_14	Select GPIO[14] as source	0xE
		RF_SPI_CLK_SRC_GPIO_15	Select GPIO[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*

10.5.0.26 GPIO_SRC_RF_GPIO03

Bit Field	Read/Write	Field Name	Description
28:24	RW	GPIO3	RF front-end GPIO3 input selection
20:16	RW	GPIO2	RF front-end GPIO2 input selection
12:8	RW	GPIO1	RF front-end GPIO1 input selection
4:0	RW	GPIO0	RF front-end GPIO0 input selection

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
28:24	GPIO3	RF_GPIO3_SRC_GPIO_0	Select GPIO[0] as source	0x0
		RF_GPIO3_SRC_GPIO_1	Select GPIO[1] as source	0x1
		RF_GPIO3_SRC_GPIO_2	Select GPIO[3] as source	0x2
		RF_GPIO3_SRC_GPIO_3	Select GPIO[3] as source	0x3
		RF_GPIO3_SRC_GPIO_4	Select GPIO[4] as source	0x4
		RF_GPIO3_SRC_GPIO_5	Select GPIO[5] as source	0x5
		RF_GPIO3_SRC_GPIO_6	Select GPIO[6] as source	0x6
		RF_GPIO3_SRC_GPIO_7	Select GPIO[7] as source	0x7
		RF_GPIO3_SRC_GPIO_8	Select GPIO[8] as source	0x8
		RF_GPIO3_SRC_GPIO_9	Select GPIO[9] as source	0x9
		RF_GPIO3_SRC_GPIO_10	Select GPIO[10] as source	0xA
		RF_GPIO3_SRC_GPIO_11	Select GPIO[11] as source	0xB
		RF_GPIO3_SRC_GPIO_12	Select GPIO[12] as source	0xC
		RF_GPIO3_SRC_GPIO_13	Select GPIO[13] as source	0xD
		RF_GPIO3_SRC_GPIO_14	Select GPIO[14] as source	0xE
		RF_GPIO3_SRC_GPIO_15	Select GPIO[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*
20:16	GPIO2	RF_GPIO2_SRC_GPIO_0	Select GPIO[0] as source	0x0
		RF_GPIO2_SRC_GPIO_1	Select GPIO[1] as source	0x1
		RF_GPIO2_SRC_GPIO_2	Select GPIO[2] as source	0x2
		RF_GPIO2_SRC_GPIO_3	Select GPIO[3] as source	0x3
		RF_GPIO2_SRC_GPIO_4	Select GPIO[4] as source	0x4
		RF_GPIO2_SRC_GPIO_5	Select GPIO[5] as source	0x5
		RF_GPIO2_SRC_GPIO_6	Select GPIO[6] as source	0x6
		RF_GPIO2_SRC_GPIO_7	Select GPIO[7] as source	0x7
		RF_GPIO2_SRC_GPIO_8	Select GPIO[8] as source	0x8
		RF_GPIO2_SRC_GPIO_9	Select GPIO[9] as source	0x9
		RF_GPIO2_SRC_GPIO_10	Select GPIO[10] as source	0xA

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		RF_GPIO2_SRC_GPIO_11	Select GPIO[11] as source	0xB
		RF_GPIO2_SRC_GPIO_12	Select GPIO[12] as source	0xC
		RF_GPIO2_SRC_GPIO_13	Select GPIO[13] as source	0xD
		RF_GPIO2_SRC_GPIO_14	Select GPIO[14] as source	0xE
		RF_GPIO2_SRC_GPIO_15	Select GPIO[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
12:8	GPIO1	RF_GPIO1_SRC_GPIO_0	Select GPIO[0] as source	0x0
		RF_GPIO1_SRC_GPIO_1	Select GPIO[1] as source	0x1
		RF_GPIO1_SRC_GPIO_2	Select GPIO[1] as source	0x2
		RF_GPIO1_SRC_GPIO_3	Select GPIO[3] as source	0x3
		RF_GPIO1_SRC_GPIO_4	Select GPIO[4] as source	0x4
		RF_GPIO1_SRC_GPIO_5	Select GPIO[5] as source	0x5
		RF_GPIO1_SRC_GPIO_6	Select GPIO[6] as source	0x6
		RF_GPIO1_SRC_GPIO_7	Select GPIO[7] as source	0x7
		RF_GPIO1_SRC_GPIO_8	Select GPIO[8] as source	0x8
		RF_GPIO1_SRC_GPIO_9	Select GPIO[9] as source	0x9
		RF_GPIO1_SRC_GPIO_10	Select GPIO[10] as source	0xA
		RF_GPIO1_SRC_GPIO_11	Select GPIO[11] as source	0xB
		RF_GPIO1_SRC_GPIO_12	Select GPIO[12] as source	0xC
		RF_GPIO1_SRC_GPIO_13	Select GPIO[13] as source	0xD
		RF_GPIO1_SRC_GPIO_14	Select GPIO[14] as source	0xE
		RF_GPIO1_SRC_GPIO_15	Select GPIO[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
4:0	GPIO0	RF_GPIO0_SRC_GPIO_0	Select GPIO[0] as source	0x0
		RF_GPIO0_SRC_GPIO_1	Select GPIO[1] as source	0x1
		RF_GPIO0_SRC_GPIO_2	Select GPIO[0] as source	0x2
		RF_GPIO0_SRC_GPIO_3	Select GPIO[3] as source	0x3
		RF_GPIO0_SRC_GPIO_4	Select GPIO[4] as source	0x4
		RF_GPIO0_SRC_GPIO_5	Select GPIO[5] as source	0x5

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		RF_GPIO0_SRC_GPIO_6	Select GPIO[6] as source	0x6
		RF_GPIO0_SRC_GPIO_7	Select GPIO[7] as source	0x7
		RF_GPIO0_SRC_GPIO_8	Select GPIO[8] as source	0x8
		RF_GPIO0_SRC_GPIO_9	Select GPIO[9] as source	0x9
		RF_GPIO0_SRC_GPIO_10	Select GPIO[10] as source	0xA
		RF_GPIO0_SRC_GPIO_11	Select GPIO[11] as source	0xB
		RF_GPIO0_SRC_GPIO_12	Select GPIO[12] as source	0xC
		RF_GPIO0_SRC_GPIO_13	Select GPIO[13] as source	0xD
		RF_GPIO0_SRC_GPIO_14	Select GPIO[14] as source	0xE
		RF_GPIO0_SRC_GPIO_15	Select GPIO[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

10.5.0.27 GPIO_SRC_RF_GPIO47

Bit Field	Read/Write	Field Name	Description
28:24	RW	GPIO7	RF front-end GPIO7 input selection
20:16	RW	GPIO6	RF front-end GPIO6 input selection
12:8	RW	GPIO5	RF front-end GPIO5 input selection
4:0	RW	GPIO4	RE front-end GPIO4 input selection

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
28:24	GPIO7	RF_GPIO7_SRC_GPIO_0	Select GPIO[0] as source	0x0
		RF_GPIO7_SRC_GPIO_1	Select GPIO[1] as source	0x1
		RF_GPIO7_SRC_GPIO_2	Select GPIO[5] as source	0x2
		RF_GPIO7_SRC_GPIO_3	Select GPIO[3] as source	0x3
		RF_GPIO7_SRC_GPIO_4	Select GPIO[4] as source	0x4
		RF_GPIO7_SRC_GPIO_5	Select GPIO[5] as source	0x5
		RF_GPIO7_SRC_GPIO_6	Select GPIO[6] as source	0x6
		RF_GPIO7_SRC_GPIO_7	Select GPIO[7] as source	0x7
		RF_GPIO7_SRC_GPIO_8	Select GPIO[8] as source	0x8

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		RF_GPIO7_SRC_GPIO_9	Select GPIO[9] as source	0x9
		RF_GPIO7_SRC_GPIO_10	Select GPIO[10] as source	0xA
		RF_GPIO7_SRC_GPIO_11	Select GPIO[11] as source	0xB
		RF_GPIO7_SRC_GPIO_12	Select GPIO[12] as source	0xC
		RF_GPIO7_SRC_GPIO_13	Select GPIO[13] as source	0xD
		RF_GPIO7_SRC_GPIO_14	Select GPIO[14] as source	0xE
		RF_GPIO7_SRC_GPIO_15	Select GPIO[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
20:16	GPIO6	RF_GPIO6_SRC_GPIO_0	Select GPIO[0] as source	0x0
		RF_GPIO6_SRC_GPIO_1	Select GPIO[1] as source	0x1
		RF_GPIO6_SRC_GPIO_2	Select GPIO[5] as source	0x2
		RF_GPIO6_SRC_GPIO_3	Select GPIO[3] as source	0x3
		RF_GPIO6_SRC_GPIO_4	Select GPIO[4] as source	0x4
		RF_GPIO6_SRC_GPIO_5	Select GPIO[5] as source	0x5
		RF_GPIO6_SRC_GPIO_6	Select GPIO[6] as source	0x6
		RF_GPIO6_SRC_GPIO_7	Select GPIO[7] as source	0x7
		RF_GPIO6_SRC_GPIO_8	Select GPIO[8] as source	0x8
		RF_GPIO6_SRC_GPIO_9	Select GPIO[9] as source	0x9
		RF_GPIO6_SRC_GPIO_10	Select GPIO[10] as source	0xA
		RF_GPIO6_SRC_GPIO_11	Select GPIO[11] as source	0xB
		RF_GPIO6_SRC_GPIO_12	Select GPIO[12] as source	0xC
		RF_GPIO6_SRC_GPIO_13	Select GPIO[13] as source	0xD
		RF_GPIO6_SRC_GPIO_14	Select GPIO[14] as source	0xE
		RF_GPIO6_SRC_GPIO_15	Select GPIO[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
12:8	GPIO5	RF_GPIO5_SRC_GPIO_0	Select GPIO[0] as source	0x0
		RF_GPIO5_SRC_GPIO_1	Select GPIO[1] as source	0x1
		RF_GPIO5_SRC_GPIO_2	Select GPIO[5] as source	0x2

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		RF_GPIO5_SRC_GPIO_3	Select GPIO[3] as source	0x3
		RF_GPIO5_SRC_GPIO_4	Select GPIO[4] as source	0x4
		RF_GPIO5_SRC_GPIO_5	Select GPIO[5] as source	0x5
		RF_GPIO5_SRC_GPIO_6	Select GPIO[6] as source	0x6
		RF_GPIO5_SRC_GPIO_7	Select GPIO[7] as source	0x7
		RF_GPIO5_SRC_GPIO_8	Select GPIO[8] as source	0x8
		RF_GPIO5_SRC_GPIO_9	Select GPIO[9] as source	0x9
		RF_GPIO5_SRC_GPIO_10	Select GPIO[10] as source	0xA
		RF_GPIO5_SRC_GPIO_11	Select GPIO[11] as source	0xB
		RF_GPIO5_SRC_GPIO_12	Select GPIO[12] as source	0xC
		RF_GPIO5_SRC_GPIO_13	Select GPIO[13] as source	0xD
		RF_GPIO5_SRC_GPIO_14	Select GPIO[14] as source	0xE
		RF_GPIO5_SRC_GPIO_15	Select GPIO[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
4:0	GPIO4	RF_GPIO4_SRC_GPIO_0	Select GPIO[0] as source	0x0
		RF_GPIO4_SRC_GPIO_1	Select GPIO[1] as source	0x1
		RF_GPIO4_SRC_GPIO_2	Select GPIO[4] as source	0x2
		RF_GPIO4_SRC_GPIO_3	Select GPIO[3] as source	0x3
		RF_GPIO4_SRC_GPIO_4	Select GPIO[4] as source	0x4
		RF_GPIO4_SRC_GPIO_5	Select GPIO[5] as source	0x5
		RF_GPIO4_SRC_GPIO_6	Select GPIO[6] as source	0x6
		RF_GPIO4_SRC_GPIO_7	Select GPIO[7] as source	0x7
		RF_GPIO4_SRC_GPIO_8	Select GPIO[8] as source	0x8
		RF_GPIO4_SRC_GPIO_9	Select GPIO[9] as source	0x9
		RF_GPIO4_SRC_GPIO_10	Select GPIO[10] as source	0xA
		RF_GPIO4_SRC_GPIO_11	Select GPIO[11] as source	0xB
		RF_GPIO4_SRC_GPIO_12	Select GPIO[12] as source	0xC
		RF_GPIO4_SRC_GPIO_13	Select GPIO[13] as source	0xD
		RF_GPIO4_SRC_GPIO_14	Select GPIO[14] as source	0xE

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		RF_GPIO4_SRC_GPIO_15	Select GPIO[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*

10.5.0.28 GPIO_SRC_RF_GPIO89

Bit Field	Read/Write	Field Name	Description
12:8	RW	GPIO9	RF front-end GPIO9 input selection
4:0	RW	GPIO8	RF front-end GPIO8 input selection

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
12:8	GPIO9	RF_GPIO9_SRC_GPIO_0	Select GPIO[0] as source	0x0
		RF_GPIO9_SRC_GPIO_1	Select GPIO[1] as source	0x1
		RF_GPIO9_SRC_GPIO_2	Select GPIO[5] as source	0x2
		RF_GPIO9_SRC_GPIO_3	Select GPIO[3] as source	0x3
		RF_GPIO9_SRC_GPIO_4	Select GPIO[4] as source	0x4
		RF_GPIO9_SRC_GPIO_5	Select GPIO[5] as source	0x5
		RF_GPIO9_SRC_GPIO_6	Select GPIO[6] as source	0x6
		RF_GPIO9_SRC_GPIO_7	Select GPIO[7] as source	0x7
		RF_GPIO9_SRC_GPIO_8	Select GPIO[8] as source	0x8
		RF_GPIO9_SRC_GPIO_9	Select GPIO[9] as source	0x9
		RF_GPIO9_SRC_GPIO_10	Select GPIO[10] as source	0xA
		RF_GPIO9_SRC_GPIO_11	Select GPIO[11] as source	0xB
		RF_GPIO9_SRC_GPIO_12	Select GPIO[12] as source	0xC
		RF_GPIO9_SRC_GPIO_13	Select GPIO[13] as source	0xD
		RF_GPIO9_SRC_GPIO_14	Select GPIO[14] as source	0xE
		RF_GPIO9_SRC_GPIO_15	Select GPIO[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
4:0	GPIO8	RF_GPIO8_SRC_GPIO_0	Select GPIO[0] as source	0x0

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		RF_GPIO8_SRC_GPIO_1	Select GPIO[1] as source	0x1
		RF_GPIO8_SRC_GPIO_2	Select GPIO[5] as source	0x2
		RF_GPIO8_SRC_GPIO_3	Select GPIO[3] as source	0x3
		RF_GPIO8_SRC_GPIO_4	Select GPIO[4] as source	0x4
		RF_GPIO8_SRC_GPIO_5	Select GPIO[5] as source	0x5
		RF_GPIO8_SRC_GPIO_6	Select GPIO[6] as source	0x6
		RF_GPIO8_SRC_GPIO_7	Select GPIO[7] as source	0x7
		RF_GPIO8_SRC_GPIO_8	Select GPIO[8] as source	0x8
		RF_GPIO8_SRC_GPIO_9	Select GPIO[9] as source	0x9
		RF_GPIO8_SRC_GPIO_10	Select GPIO[10] as source	0xA
		RF_GPIO8_SRC_GPIO_11	Select GPIO[11] as source	0xB
		RF_GPIO8_SRC_GPIO_12	Select GPIO[12] as source	0xC
		RF_GPIO8_SRC_GPIO_13	Select GPIO[13] as source	0xD
		RF_GPIO8_SRC_GPIO_14	Select GPIO[14] as source	0xE
		RF_GPIO8_SRC_GPIO_15	Select GPIO[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

10.5.0.29 GPIO_SRC_RF_CTE

Bit Field	Read/Write	Field Name	Description
12:8	RW	CTE_MODE	RF front-end CTE_MODE input selection
4:0	RW	CTE_SAMPLE_P	RF front-end CTE_SAMPLE_P input selection

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
12:8	CTE_MODE	RF_CTE_MODE_SRC_GPIO_0	Select GPIO[0] as source	0x0
		RF_CTE_MODE_SRC_GPIO_1	Select GPIO[1] as source	0x1
		RF_CTE_MODE_SRC_GPIO_2	Select GPIO[5] as source	0x2
		RF_CTE_MODE_SRC_GPIO_3	Select GPIO[3] as source	0x3
		RF_CTE_MODE_SRC_GPIO_4	Select GPIO[4] as source	0x4
		RF_CTE_MODE_SRC_GPIO_5	Select GPIO[5] as source	0x5
		RF_CTE_MODE_SRC_GPIO_6	Select GPIO[6] as source	0x6

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		RF_CTE_MODE_SRC_GPIO_7	Select GPIO[7] as source	0x7
		RF_CTE_MODE_SRC_GPIO_8	Select GPIO[8] as source	0x8
		RF_CTE_MODE_SRC_GPIO_9	Select GPIO[9] as source	0x9
		RF_CTE_MODE_SRC_GPIO_10	Select GPIO[10] as source	0xA
		RF_CTE_MODE_SRC_GPIO_11	Select GPIO[11] as source	0xB
		RF_CTE_MODE_SRC_GPIO_12	Select GPIO[12] as source	0xC
		RF_CTE_MODE_SRC_GPIO_13	Select GPIO[13] as source	0xD
		RF_CTE_MODE_SRC_GPIO_14	Select GPIO[14] as source	0xE
		RF_CTE_MODE_SRC_GPIO_15	Select GPIO[15] as source	0xF
		RF_CTE_MODE_SRC_CONST_LOW	Select constant low as source	0x10
		RF_CTE_MODE_SRC_CONST_HIGH	Select constant high as source	0x11
		RF_CTE_MODE_SRC_BB_CTE_MODE	Select baseband CTE_MODE as source	0x12*
4:0	CTE_SAMPLE_P	RF_CTE_SAMPLE_P_SRC_GPIO_0	Select GPIO[0] as source	0x0
		RF_CTE_SAMPLE_P_SRC_GPIO_1	Select GPIO[1] as source	0x1
		RF_CTE_SAMPLE_P_SRC_GPIO_2	Select GPIO[5] as source	0x2
		RF_CTE_SAMPLE_P_SRC_GPIO_3	Select GPIO[3] as source	0x3
		RF_CTE_SAMPLE_P_SRC_GPIO_4	Select GPIO[4] as source	0x4
		RF_CTE_SAMPLE_P_SRC_GPIO_5	Select GPIO[5] as source	0x5
		RF_CTE_SAMPLE_P_SRC_GPIO_6	Select GPIO[6] as source	0x6
		RF_CTE_SAMPLE_P_SRC_GPIO_7	Select GPIO[7] as source	0x7
		RF_CTE_SAMPLE_P_SRC_GPIO_8	Select GPIO[8] as source	0x8
		RF_CTE_SAMPLE_P_SRC_GPIO_9	Select GPIO[9] as source	0x9
		RF_CTE_SAMPLE_P_SRC_GPIO_10	Select GPIO[10] as source	0xA
		RF_CTE_SAMPLE_P_SRC_GPIO_11	Select GPIO[11] as source	0xB
		RF_CTE_SAMPLE_P_SRC_GPIO_12	Select GPIO[12] as source	0xC
		RF_CTE_SAMPLE_P_SRC_GPIO_13	Select GPIO[13] as source	0xD
		RF_CTE_SAMPLE_P_SRC_GPIO_14	Select GPIO[14] as source	0xE

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		RF_CTE_SAMPLE_P_SRC_GPIO_15	Select GPIO[15] as source	0xF
		RF_CTE_SAMPLE_P_SRC_CONST_LOW	Select constant low as source	0x10
		RF_CTE_SAMPLE_P_SRC_CONST_HIGH	Select constant high as source	0x11
		RF_CTE_SAMPLE_P_SRC_BB_CTE_MODE	Select baseband CTE_SAMPLE_P as source	0x12*

10.5.0.30 GPIO_SRC_ASCC

Bit Field	Read/Write	Field Name	Description
12:8	RW	ASYNC_CLOCK	ASCC asynchronous clock source input selection
4:0	RW	SYNC_PULSE	ASCC synchronization pulse input selection

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
12:8	ASYNC_CLOCK	ASCC_ASYNC_CLOCK_SRC_GPIO_0	Select GPIO[0] as source	0x0
		ASCC_ASYNC_CLOCK_SRC_GPIO_1	Select GPIO[1] as source	0x1
		ASCC_ASYNC_CLOCK_SRC_GPIO_2	Select GPIO[5] as source	0x2
		ASCC_ASYNC_CLOCK_SRC_GPIO_3	Select GPIO[3] as source	0x3
		ASCC_ASYNC_CLOCK_SRC_GPIO_4	Select GPIO[4] as source	0x4
		ASCC_ASYNC_CLOCK_SRC_GPIO_5	Select GPIO[5] as source	0x5
		ASCC_ASYNC_CLOCK_SRC_GPIO_6	Select GPIO[6] as source	0x6
		ASCC_ASYNC_CLOCK_SRC_GPIO_7	Select GPIO[7] as source	0x7
		ASCC_ASYNC_CLOCK_SRC_GPIO_8	Select GPIO[8] as source	0x8
		ASCC_ASYNC_CLOCK_SRC_GPIO_9	Select GPIO[9] as source	0x9
		ASCC_ASYNC_CLOCK_SRC_GPIO_10	Select GPIO[10] as source	0xA

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		ASCC_ASYNC_CLOCK_SRC_GPIO_11	Select GPIO[11] as source	0xB
		ASCC_ASYNC_CLOCK_SRC_GPIO_12	Select GPIO[12] as source	0xC
		ASCC_ASYNC_CLOCK_SRC_GPIO_13	Select GPIO[13] as source	0xD
		ASCC_ASYNC_CLOCK_SRC_GPIO_14	Select GPIO[14] as source	0xE
		ASCC_ASYNC_CLOCK_SRC_GPIO_15	Select GPIO[15] as source	0xF
		ASCC_ASYNC_CLOCK_SRC_CONST_LOW	Select constant low as source	0x10*
		ASCC_ASYNC_CLOCK_SRC_CONST_HIGH	Select constant high as source	0x11
		ASCC_ASYNC_CLOCK_SRC_STANDBYCLK	Select STANDBYCLK as source	0x12
4:0	SYNC_PULSE	ASCC_SYNC_PULSE_SRC_GPIO_0	Select GPIO[0] as source	0x0
		ASCC_SYNC_PULSE_SRC_GPIO_1	Select GPIO[1] as source	0x1
		ASCC_SYNC_PULSE_SRC_GPIO_2	Select GPIO[5] as source	0x2
		ASCC_SYNC_PULSE_SRC_GPIO_3	Select GPIO[3] as source	0x3
		ASCC_SYNC_PULSE_SRC_GPIO_4	Select GPIO[4] as source	0x4
		ASCC_SYNC_PULSE_SRC_GPIO_5	Select GPIO[5] as source	0x5
		ASCC_SYNC_PULSE_SRC_GPIO_6	Select GPIO[6] as source	0x6
		ASCC_SYNC_PULSE_SRC_GPIO_7	Select GPIO[7] as source	0x7
		ASCC_SYNC_PULSE_SRC_GPIO_8	Select GPIO[8] as source	0x8
		ASCC_SYNC_PULSE_SRC_GPIO_9	Select GPIO[9] as source	0x9
		ASCC_SYNC_PULSE_SRC_GPIO_10	Select GPIO[10] as source	0xA
		ASCC_SYNC_PULSE_SRC_GPIO_11	Select GPIO[11] as source	0xB
		ASCC_SYNC_PULSE_SRC_GPIO_12	Select GPIO[12] as source	0xC
		ASCC_SYNC_PULSE_SRC_GPIO_13	Select GPIO[13] as source	0xD
		ASCC_SYNC_PULSE_SRC_GPIO_14	Select GPIO[14] as source	0xE

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		ASCC_SYNC_PULSE_SRC_GPIO_15	Select GPIO[15] as source	0xF
		ASCC_SYNC_PULSE_SRC_CONST_LOW	Select constant low as source	0x10*
		ASCC_SYNC_PULSE_SRC_CONST_HIGH	Select constant high as source	0x11
		ASCC_SYNC_PULSE_SRC_STANDBYCLK	Select STANDBYCLK as source	0x12

10.5.0.31 GPIO_SRC_EXTCLK

Bit Field	Read/Write	Field Name	Description
4:0	RW	EXTCLK	EXTCLK input selection

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
4:0	EXTCLK	EXTCLK_SRC_GPIO_0	Select GPIO[0] as source	0x0
		EXTCLK_SRC_GPIO_1	Select GPIO[1] as source	0x1
		EXTCLK_SRC_GPIO_2	Select GPIO[2] as source	0x2
		EXTCLK_SRC_GPIO_3	Select GPIO[3] as source	0x3
		EXTCLK_SRC_GPIO_4	Select GPIO[4] as source	0x4
		EXTCLK_SRC_GPIO_5	Select GPIO[5] as source	0x5
		EXTCLK_SRC_GPIO_6	Select GPIO[6] as source	0x6
		EXTCLK_SRC_GPIO_7	Select GPIO[7] as source	0x7
		EXTCLK_SRC_GPIO_8	Select GPIO[8] as source	0x8
		EXTCLK_SRC_GPIO_9	Select GPIO[9] as source	0x9
		EXTCLK_SRC_GPIO_10	Select GPIO[10] as source	0xA
		EXTCLK_SRC_GPIO_11	Select GPIO[11] as source	0xB
		EXTCLK_SRC_GPIO_12	Select GPIO[12] as source	0xC
		EXTCLK_SRC_GPIO_13	Select GPIO[13] as source	0xD
		EXTCLK_SRC_GPIO_14	Select GPIO[14] as source	0xE
		EXTCLK_SRC_GPIO_15	Select GPIO[15] as source	0xF
		EXTCLK_SRC_CONST_LOW	Select constant low as source	0x10*
		EXTCLK_SRC_CONST_HIGH	Select constant high as source	0x11

10.5.0.32 ACS_AOUT_CTRL

Bit Field	Read/Write	Field Name	Description
21	RW	RTC_CLOCK_GPIO0_STOP_EDGE	Stop edge for RTC clock output on AOUT
20:19	RW	RTC_CLOCK_GPIO0_STOP_SRC	Stop source for RTC clock output on AOUT
18:16	RW	RTC_CLOCK_GPIO0_START	Start event for RTC clock output on AOUT (RTC prescaler and counter need to be enabled)
13	RW	AOUT_IOUT_SEL_TO_GPIO	Selection between AOUT voltage or PTAT current source to GPIO
12:8	RW	AOUT_TO_GPIO	Select to which GPIO the AOUT voltage or PTAT current provided
5:0	RW	TEST_AOUT	AOUT test signal selection

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
21	RTC_CLOCK_GPIO0_STOP_EDGE	GPIO0_RTC_CLK_STOP_RISING	Stop to output RTC clock on rising edge	0x0*
		GPIO0_RTC_CLK_STOP_FALLING	Stop to output RTC clock on falling edge	0x1
20:19	RTC_CLOCK_GPIO0_STOP_SRC	GPIO0_RTC_CLK_STOP_GPIO0	Stop to output RTC clock on GPIO0 event	0x0*
		GPIO0_RTC_CLK_STOP_GPIO1	Stop to output RTC clock on GPIO1 event	0x1
		GPIO0_RTC_CLK_STOP_GPIO2	Stop to output RTC clock on GPIO2 event	0x2
		GPIO0_RTC_CLK_STOP_GPIO3	Stop to output RTC clock on GPIO3 event	0x3
18:16	RTC_CLOCK_GPIO0_START	GPIO0_RTC_CLK_DISABLE	No start event (GPIO0 not driven)	0x0*
		GPIO0_RTC_CLK_125MS	Start to output RTC clock every 125 ms	0x1
		GPIO0_RTC_CLK_250MS	Start to output RTC clock every 250 ms	0x2
		GPIO0_RTC_CLK_500MS	Start to output RTC clock every 500 ms	0x3
		GPIO0_RTC_CLK_1S	Start to output RTC clock every 1 s	0x4
		GPIO0_RTC_CLK_2S	Start to output RTC clock every 2 s	0x5
		GPIO0_RTC_CLK_4S	Start to output RTC clock every 4 s	0x6
		GPIO0_RTC_CLK_8S	Start to output RTC clock every 8 s	0x7
13	AOUT_IOUT_SEL_TO_GPIO	SEL_IOUT_TO_GPIO	Select PTAT iref current source	0x0

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		SEL_AOUT_TO_GPIO	Select AOUT voltage source	0x1*
12:8	AOUT_TO_GPIO	AOUT_TO_GPIO_0	Select GPIO[0] as output	0x0
		AOUT_TO_GPIO_1	Select GPIO[1] as output	0x1
		AOUT_TO_GPIO_2	Select GPIO[2] as output	0x2
		AOUT_TO_GPIO_3	Select GPIO[3] as output	0x3
		AOUT_TO_GPIO_4	Select GPIO[4] as output	0x4
		AOUT_TO_GPIO_5	Select GPIO[5] as output	0x5
		AOUT_TO_GPIO_6	Select GPIO[6] as output	0x6
		AOUT_TO_GPIO_7	Select GPIO[7] as output	0x7
		AOUT_TO_GPIO_8	Select GPIO[8] as output	0x8
		AOUT_TO_GPIO_9	Select GPIO[9] as output	0x9
		AOUT_TO_GPIO_10	Select GPIO[10] as output	0xA
		AOUT_TO_GPIO_11	Select GPIO[11] as output	0xB
		AOUT_TO_GPIO_12	Select GPIO[12] as output	0xC
		AOUT_TO_GPIO_13	Select GPIO[13] as output	0xD
		AOUT_TO_GPIO_14	Select GPIO[14] as output	0xE
		AOUT_TO_GPIO_15	Select GPIO[15] as output	0xF
		AOUT_NOT_CONNECTED_TO_GPIO	None selected	0x10*
5:0	TEST_AOUT	AOUT_VSSA	AOUT grounded	0x0*
		AOUT_VCC	VCC	0x1
		AOUT_IPTAT	Band-gap ptat pmos current source	0x2
		AOUT_IVBE	Band-gap vbe ptat current source	0x3
		AOUT_IREF_1U	Band-gap 1uA pmos current source	0x4
		AOUT_IREF_50N	Band-gap 50nA pmos current source	0x5
		AOUT_IREF_10N	Band-gap 10nA nmos current source	0x6
		AOUT_VREG_BG	Band-gap regulated supply voltage / flash reference voltage	0x7
		AOUT_VREF_0P75V	Band-gap reference voltage 0p75V	0x8
		AOUT_VREF_0P75V_buf	Band-gap reference voltage 0p75V after internal buffer	0x9
		AOUT_VREF_0P67V	Band-gap reference voltage 0p67V	0xA

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		AOUT_WEDAC	Band-gap WEDAC reference voltage	0xB
		AOUT_TEMP_SENSOR_VPTAT	Temperature sensor output voltage	0xC
		AOUT_NOT_USED	UNUSED Signal	0xD
		AOUT_LSAD_INTERNAL_VREF	LSAD internal vref	0xE
		AOUT_THERMISTOR_CURRENT	Thermistor current	0xF
		AOUT_IREF_1N_OUTPUT	PMU PTAT iref current source	0x10
		AOUT_VDDACS_OUTPUT	PMU vddacs voltage	0x11
		AOUT_RC_INTENAL_SUPPLY	RC intenal supply	0x12
		AOUT_RC32_INTENAL_SUPPLY	RC32 intenal supply	0x13
		AOUT_VDDA_SW	Vdda switch voltage	0x14
		AOUT_VDDSYN_SW	Vddsyn switch voltage	0x15
		AOUT_VDDRF_SW	Vddrf switch voltage	0x16
		AOUT_VDDT	VDDT switch voltage	0x17
		AOUT_VDDCCAO	VDDCCAO switch voltage	0x18
		AOUT_VDDSENSOR	VDDSENSOR switch voltage	0x19
		AOUT_VDDM	VDDM voltage output	0x1A
		AOUT_VDDC	VDDC voltage output	0x1B
		AOUT_VDDPA	VDDPA voltage output	0x1C
		AOUT_VDDPA_ISENSE	VDDPA current sensing	0x1D
		AOUT_FLASH0_TM0	Instance Flash 0 TM0 connected to AOUT	0x1E
		AOUT_VDDM_DRAM7_C	VDDM DRAM7 Core supply	0x1F
		AOUT_VDDA	VDDA voltage output	0x20
		AOUT_VDDRF	VDDRF voltage output	0x21
		AOUT_VDDC_RF	VDDC RF power island supply	0x22
		AOUT_VDDFLASH	VDDFLASH voltage output	0x23
		AOUT_HIZ	HIZ (measure leakage)	0x24
		AOUT_RC32_IBP_12N_NTC	RC32 NTC current	0x25
		AOUT_VDDM_DRAM7_P	VDDM DRAM7 Peripheral supply	0x26
		AOUT_VDDC_CC	VDDC CryptoCell power island supply	0x27
		AOUT_VDDC_BB	VDDC Baseband power island supply	0x28

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		DOUT_SP1	Digital spare 1	0x29
		DOUT_SP2	Digital spare 2	0x2A
		DOUT_BG_READY	Band-gap ready signal	0x2B
		DOUT_VDDRF_READY	vddrf ready signal	0x2C
		DOUT_VDDC_READY	Vddc ready signal	0x2D
		DOUT_VDDM_READY	Vddm ready signal	0x2E
		DOUT_VDDCP_READY	vddcp ready signal	0x2F
		DOUT_SP3	Digital spare 3	0x30
		DOUT_VDDFLASH_READY	vddflash ready signal	0x31
		DOUT_CLK_PRESENT	Clock present from clock detector	0x32
		DOUT_XTAL_OK	Xtal ok	0x33
		DOUT_XTAL_CLK	Xtal clock	0x34
		DOUT_RC32_CLK	RC 32kHz clock	0x35
		DOUT_DCDC_ACTIVATED	DC-DC converter activated signal	0x36
		DOUT_DCDC_OVERLOAD	DC-DC converter overload signal	0x37
		DOUT_DCDC_VCC_READY	DC-DC converter vcc ready signal	0x38
		DOUT_SP4	Digital spare 4	0x39
		DOUT_SP5	Digital spare 5	0x3A
		DOUT_SP6	Digital spare 6	0x3B
		DOUT_STANDBY_CLK	Standby clock	0x3C
		DOUT_SAR_COMPARATOR	Comparator output of the SAR	0x3D
		DOUT_CP_CLK	Charge Pump clock	0x3E
		DOUT_ACOMP_OUT	Analog Comparator output signal	0x3F

CHAPTER 11

Communication Interfaces

This group of topics describes the various RSL15 communication interfaces, whose functionality can enhance numerous types of applications.

11.1 I2C Interfaces

The I2C interfaces are compatible with the UM10204 I2C specification. I2C uses a two-wire interface that includes a bidirectional clock line (SCL) and a bidirectional data line (SDA). The I2C interfaces are typically used for communication with external sensors and storage devices, and as control signals for radios.

The I2C interfaces support both master and slave mode operation, and can be configured to connect to the DMA master through a DMA channel to support moving data without processor intervention. There are two interfaces on RSL15—I2C0 and I2C1—and the description that follows applies to both of them.

NOTE: Each I2C interface has its own set of registers. In the text below, a reference to an `I2C*_XXX` register (ex., `I2C*_CFG`) refers to the I2C0 and I2C1 versions of this register.

For information about configuring the GPIOs for the I2C interface, see Section 2.7 “Interfaces” on page 49.

IMPORTANT: I2C interfaces are defined using open-collector pads for the I2C 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 working with this interface must use either the internal pull-up resistors implemented for the GPIO pads, or a set of external pull-up resistors on the GPIOs assigned to be the I2C bus lines.

The I2C 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 I2C interface is designed to handle bus traffic operating at up to 1 MHz.

The example timing diagram shown in the "I2C Signal Timing Diagram" figure (Figure 53) provides information about the important elements in an I2C 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, indicating 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 (clocked in on a rising edge of SCL) provide the address of the device with which the master device wants to communicate. A device that is properly addressed needs to acknowledge this transaction if communications are to conform to the I2C standard.

Read/Write Direction Bit

During the first byte transmitted, the eighth bit on the SDA (clocked in on a rising edge of SCL) indicates the direction of the 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.

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 indicate to the sender that the byte has been received. The device receiving data needs to acknowledge each data byte (ACK), 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 (NACK), the receiving device does not respond. To ACK, the receiving device needs to pull the SDA low. A receiving slave device needs to NACK if the slave device is not the device that has been addressed, or if the device cannot handle the byte received. A master device needs to NACK if the master is receiving and wants to end the transaction. If a NACK is encountered, the device transmitting data needs to 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^2C transaction.

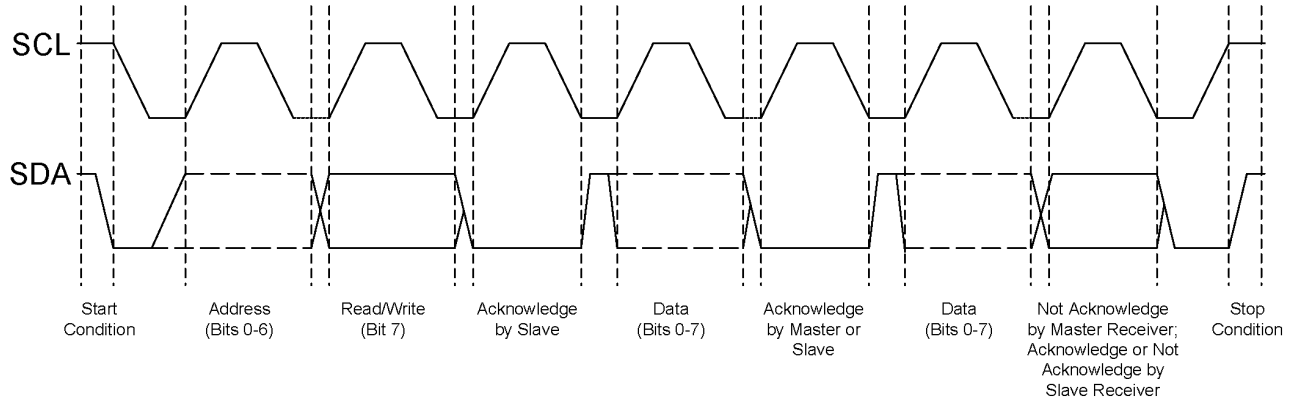


Figure 53. I^2C Signal Timing Diagram

The `I2C*_CTRL` registers are the interface control registers. They contain write-only control bits and read-only status bits to indicate the status of key control settings. Each of the write-only bits is used to trigger an event. Information about these bits is summarized in the "I2C Control Bits" table (Table 20).

Table 20. I2C Control Bits

Bit Field	Field Name	Field Description
9	`I2C_CTRL_LAST_DATA_STATUS`	Read-only bit indicating if the current byte is the last data byte for a transfer
8	`I2C_CTRL_ENABLE_STATUS`	Read-only bit indicating that an I^2C interface is enabled or disabled
6	`I2C_CTRL_LAST_DATA`	Indicate that the current data is the last byte
5	`I2C_CTRL_STOP`	Issue a stop condition on the I^2C interface

Table 20. I2C Control Bits (Continued)

Bit Field	Field Name	Field Description
4	I2C_CTRL_NACK	Issue a negative acknowledgment of a sent/received byte.
3	I2C_CTRL_ACK	Issue an acknowledgment of a sent/received byte. Resume I2C internal state machine
2	I2C_CTRL_RESET	Reset the I2C interface
1	I2C_CTRL_DISABLE	Disable the I2C interface
0	I2C_CTRL_ENABLE	Enable the I2C interface

The I2C*_STATUS registers contain bits that are defined as read-only event bits, state bits, or write-only control bits used for updating the state of the I2C interface. The event bits are used to indicate that an event has occurred, and the state bits are used to indicate an interface operating state. Information about these status bits is shown in the "I2C Event and State Status Bits" table (Table 21).

Table 21. I2C Event and State Status Bits

Bit Field	Field Name	Bit Type	Field Description
26	I2C_STATUS_STOP_OR_REPEATED_START_DETECTED	Event	Indicate if a stop or repeated start has been detected
25	I2C_STATUS_REPEATED_START_DETECTED	Event	Indicate if an I2C repeated start has been detected during an active transaction in slave mode
22	I2C_STATUS_BUS_ERROR	State	Indicate that a bus error or conflict has occurred during transmission
21	I2C_STATUS_BUSY	Event	Indicate that the reception or transmission of the data is ongoing
20	I2C_STATUS_START_PENDING	Event	Master frame start pending status bit
19	I2C_STATUS_MASTER_MODE	Event	Master mode status bit
18	I2C_STATUS_STOP_DETECTED	Event	Indicate if an I2C stop bit has been detected
17	I2C_STATUS_DATA_EVENT	State	Indicate that an I2C interface either needs data to transmit or has received data
16	I2C_STATUS_TX_REQ	State	Indicate that a TX data can be written
15	I2C_STATUS_RX_REQ	State	Indicate that an RX data byte is ready to be read
14	I2C_STATUS_CLK_STRETCH	State	Indicate that the I2C clock (SCL) is being stretched
13	I2C_STATUS_LINE_FREE	State	Indicate that the I2C bus is free
12	I2C_STATUS_ADDR_DATA	State	Indicate if the information sent on the bus is an address or data byte
11	I2C_STATUS_READ_WRITE	State	Indicate if the current I2C transaction is a read or a write

Table 21. I2C Event and State Status Bits (Continued)

Bit Field	Field Name	Bit Type	Field Description
10	I2C_STATUS_GEN_CALL	State	Indicate if the current I2C transaction used the general call address or the interface's regular address
9	I2C_STATUS_ACK	State	Indicate if the data has been acknowledged (ACK) or not-acknowledged (NACK)
8	I2C_STATUS_OVERRUN	State	Indicate that an overrun has occurred when receiving data
4	I2C_STATUS_TX_REQ_SET	Control	Write 1 to set the TX_REQ status flag, flushing the current contents of the transmit buffers
3	I2C_STATUS_REPEATED_START_ DETECTED_CLEAR	Control	Write 1 to clear the REPEATED_START_ DETECTED status flag
2	I2C_STATUS_STOP_DETECTED_ CLEAR	Control	Write 1 to clear the STOP_DETECTED status flag
1	I2C_STATUS_BUS_ERROR_CLEAR	Control	Write 1 to clear the BUS_ERROR status flag
0	I2C_STATUS_OVERRUN_CLEAR	Control	Write 1 to clear the OVERRUN status flag

NOTE: The interfaces support both master and slave operations, which can be simultaneously enabled. A user application can use the interface in both modes, provided the application ensures that the I^2C bus is not currently in use. When attempting to initialize a transfer as a master device, check the I2C_STATUS_LINE_FREE bit in the I2C*_STATUS registers to verify that the I^2C bus is not currently in use..

IMPORTANT: When multiple masters co-exist on the bus, an I2C*_STATUS_BUS_ERROR is detected if the bit driven onto the serial data line is different from the data read from the serial data line. As soon as a BUS_ERROR is detected, the I^2C interface stops driving the SCL and SDA lines, and an interrupt is generated if the BUS_ERROR_INT_ENABLE bit is set. The BUS_ERROR bit remains asserted until the SDA line is released by the other devices on the bus. A user application needs to wait for the bus to be free again and issue a stop condition to reset the slaves connected on the bus.

The I^2C interfaces use the I2C*_TX_DATA and I2C*_RX_DATA registers to respectively transmit and receive data. The I^2C interfaces use the I2C*_STATUS_TX_REQ and I2C*_STATUS_RX_REQ bits in the I2C*_STATUS registers to request data for transmission and to indicate when a new byte is available to be read. Writing to and reading from I2C*_TX_DATA and I2C*_RX_DATA automatically clears the corresponding *TX_REQ bit or RX_REQ bit (respectively) in hardware. As use of the I2C*_RX_DATA registers can advance the I^2C interface state machine, these registers are mirrored in the I2C*_RX_DATA_MIRROR registers without impacting the I^2C state machine. Reading from these registers does not clear the I2C*_STATUS_RX_REQ bit in the I2C_STATUS register. A write only bit called I2C*_STATUS_REQ_SET in the I2C_STATUS register can be used to set the I2C*_STATUS_TX_REQ register and discard the current data in the buffer.

NOTE: The I2C*_STATUS_TX_REQ status flag is useful in slave mode, when a master ends a transaction and forces a stop condition (I^2C). At this point the TX_REQ flag is zero, which indicates that there

is still a byte pending in the TX buffer. To clear this state without resetting the interface, set I2C*_STATUS_TX_REQ_SET high to request new data again. This does not generate a new interrupt (TX); only the underlying buffer is flushed.

Each byte of an I2C transaction must be ACK or NACK. Use the I2C*_CFG_AUTO_ACK_ENABLE bit from the I2C*_CFG registers to select whether the application handles this acknowledgment manually, or the interface handles it automatically. In either case, if the application does not provide data in time or move the interface's internal state machine forward, 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 I2C state machine is described in "Operation Using Manual Acknowledgment" on page 581 for manual acknowledgment mode, and in "Operation Using Automatic Acknowledgment" on page 582 for automatic acknowledgment mode.

NOTE: If, at any point when operating in either master or slave mode, a user application needs to reset the I2C bus lines (for example, after encountering a bus error condition), the application needs to write the I2C_CTRL_RESET bit from the I2C_CTRL register. This resets both the I2C bus and the internal state machines of the I2C interface, allowing the system to cleanly start again from a known state.

11.1.1 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 I2C transfer. For communications to proceed on the I2C interface, the following relation between the system clock (which is used internally to clock the I2C interface) and the SCL must hold:

$$F_{SYSCLK} \geq 3.125 \times F_{SCL}$$

When in slave mode, the I2C interfaces 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_CFG_SLAVE bit fields in the I2C*_CFG registers are used to configure the settling time required for the SDA line, by selecting the number of SYSCLK cycles between the time when data is put on the SDA line and when the SCL line is released. It is important that the software be able to keep up with data transmitting from the master at the full rate (1 Mbps in Fast-mode Plus), to minimize the overhead due to clock stretching. For more information, see "Setting the System Clock" on page 581.

To enable a device in slave mode, set the I2C_CFG_SLAVE bit from the appropriate I2C*_CFG register. Both master and slave operation can be enabled at the same time. The application can use the I2C_STATUS_MASTER_MODE bit field to identify which state machine is active when an action is being requested by the interface. The device responds to communications on both the I2C general call address (0x00; I2C write transactions only) and at a programmable slave address that can be selected using the I2C_CFG_SLAVE_ADDRESS bit fields in the I2C*_CFG registers. During a transaction, the I2C_STATUS_GEN_CALL bit in the appropriate 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_CTRL_LAST_DATA bit from the appropriate I2C*_CTRL register to automatically NACK the last data byte, to indicate to a master device that it needs to send a stop condition.

NOTE: The I2C_CTRL_LAST_DATA bit is cleared when a stop condition is received.

11.1.2 Master Mode Specific Configuration

In master mode, the device provides the serial clock signal and controls all transfer information. To configure the clock signal, you need to select a prescaling division by 3 using the I2C_CFG_MASTER_PRESCALE bit field in the appropriate I2C*_CFG register. The interface clock frequency for a given configuration can be calculated using this formula:

$$F_{I2C\ MASTER} = \frac{F_{SYSCLK}}{3\times(I2C\_CFG\_MASTER\_PRESCALE + 1)}$$

NOTE: The System clock needs to be set to at least 1.5 MHz for Fast-mode (400 kbps), and at least 6 MHz for Fast-mode Plus (1 Mbps). For more information, see "Setting the System Clock" on the next page

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^{2}C bus lines are not currently in use, by reading the I2C_STATUS_LINE_FREE bit in the appropriate 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 11.1.4 “I2C Interrupts” on page 581).
4. Complete the transaction by sending a stop condition. A stop condition can be generated using either the I2C_CTRL_STOP bit or the I2C_CTRL_LAST_DATA bit in the I2C*_CTRL registers.

When using the I2C_CTRL_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. Multi-master mode can be supported in software by handling the BUS_ERROR status bit, which is set when a conflict occurs. The arbitration procedure and clock synchronization are active in master mode.

NOTE: If the I^{2}C 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: The I2C_CTRL1_LAST_DATA bit is cleared when a stop condition is received on the I^{2}C bus.

When using the I2C_CTRL_STOP bit, a stop condition is issued immediately. This bit is normally set when the I2C_CTRL_ACK bit or I2C_CTRL_NACK bit in the I2C_CTRL register is set. In master mode, a stop condition is automatically generated if data or an address has not been acknowledged by the slave.

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.

The following list shows the interrupt triggers which can be enabled, and the reasons why they would be triggered:

- TX_INT_ENABLE (triggered when a byte is transmitted)
- RX_INT_ENABLE (triggered when a byte is received)
- BUS_ERROR_INT_ENABLE (triggered when a bus error occurs)
- OVERRUN_INT_ENABLE (triggered when any data has been received that has overwritten the RX data buffer before the data in the buffer could be read)
- STOP_INT_ENABLE (triggered when a stop event occurs)

- REPEATED_START_INT_ENABLE (triggered when a repeated start event occurs)

11.1.3 Setting the System Clock

A maximum transmission rate of 400 kbps (Fast-mode) is fully supported by the I^{2}C interface. In this mode, the system clock must be set to at least 1.5 MHz for Fast-mode (400 kbps) for a system with a sufficient pull-up resistance on SDA to provide a pull-up time of less than 150 ns.

A maximum transmission rate of 1 Mbps (Fast-mode plus) is supported, with the exception of the pad drive strength required by the I^{2}C standard (20 mA at 0.4 V). Multiple GPIO pads can be used in parallel to increase the drive strength, but this might not be sufficient to meet the specifications, especially at low VDDO. In this mode, the sytem clock must be set to at least 6 MHz for Fast-mode Plus (1 Mbps) for a system with a sufficient pull-up resistance on a reasonably fast SDA to provide a pull-up time of less than 100 ns.

11.1.4 I^{2}C Interrupts

The I^{2}C interfaces use associated interrupts which, when enabled, signal the receipt of a correct address byte and the completion of each data byte in the transaction. Interrupts can also be enabled to identify bus-related conditions, such as bus errors during arbitration, stop conditions on the bus, and repeated start conditions (in slave mode only). In addition, the buffer overrun interrupt is available to identify firmware issues.

See Section 3.3.2 “Nested Vector Interrupt Controller (NVIC)” on page 58 for information regarding interrupt configuration and handling interrupts for the Arm Cortex-M33 processor.

Several status indicators from the I2C*_STATUS registers 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 has been generated in response to a recognized sequence, including a start condition and address on the I^{2}C interface. No interrupt is generated for I^{2}C transactions that use an address which has not been recognized by the I^{2}C interface.
- If the I2C_STATUS_STOP_DETECTED bit is set, the interrupt has been generated in response to a stop interrupt being a condition for an active transaction on the I^{2}C interface. To clear the status of this bit, write a 1 to the I2C_STATUS_STOP_DETECTED_CLEAR bit.
- 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 in order to continue with the data transmission.
- If the I2C_STATUS_BUS_ERROR bit is set, then a bus error has occurred.
- If the REPEATED_START_DETECTED bit is set, a repeated start condition has been detected by the device. This interrupt occurs during an active transaction in slave mode.

11.1.4.1 Operation Using Manual Acknowledgment

If an application is using manual acknowledgment (i.e., the I2C*_CFG_AUTO_ACK_ENABLE bit from the I2C*_CFG registers is disabled), the application needs to use the I2C_CTRL_NACK and I2C_CTRL_ACK bits from the I2C*_CTRL registers to acknowledge (ACK) or not-acknowledge (NACK) transaction bytes.

When handling a transaction, an interrupt is issued whenever data or an acknowledgment 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 if stop interrupts are enabled) on the stop condition that is expected to follow the NACK.

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: it does not require data nor acknowledgment 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 a resume command by writing a 1 to the `I2C_CTRL_RESUME` bit command, which continues the transfer.

NOTE: Use of the `I2C_CTRL_NACK, I2C_CTRL_ACK` and `I2C_CTRL_STOP` bits from the `I2C*_CTRL` registers is only defined when an acknowledgment is needed during a data transfer. Interface behavior in response to these control bits being set at other times is undefined.

11.1.4.2 Operation Using Automatic Acknowledgment

When operating using auto acknowledgment (i.e., the `I2C_CFG_AUTO_ACK_ENABLE` bit from the appropriate `I2C*_CFG` 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, if the firmware is able to handle each data byte fast enough. A DMA channel can be used in auto acknowledgment mode to reduce CPU utilization.

NOTE: DMA-based transfers are only supported in auto acknowledge mode.

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 if stop interrupts are enabled) on the stop condition that is expected to follow the NACK.

Additionally, interrupts are generated if data has been supplied but cannot be sent because the interface has received a NACK from the slave device, or is currently idle (for instance, after 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 has been 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 acknowledgment 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.

11.1.4.3 I2C Main Modes of Operation

The operation modes of the I2C interfaces fall into two different categories: master modes, and slave modes. The "I2C Master Modes of Operation" table (Table 22) and the "I2C Slave Modes of Operation" table (Table 23) show the master modes of operation and slave modes of operation, respectively, along with details of how they function.

Table 22. I2C Master Modes of Operation

Mode	Auto ACK Off, Write Frame	Auto ACK, Read Frame	Auto ACK On, Write Frame	Auto ACK On, Read Frame
Start	Receive start, interrupt if repeated start and repeated start interrupt are enabled	Receive start, interrupt if repeated start and repeated start interrupt are enabled	Receive start, interrupt if repeated start and repeated start interrupt are enabled	Receive start, interrupt if repeated start and repeated start interrupt are enabled
Addr	Receive addr	Receive addr	Receive addr	Receive addr
Ack	Interrupt if addr is matched, stretch clock until ACK/NACK, send ACK/NACK	Interrupt if addr is matched, stretch clock until ACK/NACK, send ACK/NACK	ACK if addr is matched	ACK if addr is matched
Data0	Receive RX data	Interrupt if TX data not available, stretch until TX data available, send TX data	Receive RX data	Interrupt if TX data not available, stretch until TX data available, send TX data, interrupt
Ack	Interrupt, stretch clock until ACK/NACK, send ACK/NACK	ReceiveACK/NACKfrom remote	Interrupt, stretch clock until RX data is read, ACK	Receive ACK/NACK from remote
Data N	Receive RX data	Interrupt if TX data not available, stretch until TX data available, send TX data	Receive RX data	Interrupt if TX data not available, stretch until TX data available, interrupt, send TX data
Ack N	Interrupt, stretch clock until ACK/NACK, send ACK/NACK	Receive ACK/NACK from remote	Interrupt, stretch clock until RX data is read, ACK	Receive ACK/NACK from remote
Stop	Receive stop, interrupt if stop interrupt is enabled	Receive stop, interrupt if stop interrupt is enabled	Receive stop, interrupt if stop interrupt is enabled	Receive stop, interrupt if stop interrupt is enabled

Table 23. I2C Slave Modes of Operation

Mode	Auto ACK Off, Write Frame	Auto ACK, Read Frame	Auto ACK On, Write Frame	Auto ACK On, Read Frame
Start	Send start	Send start	Send start	Send start
Addr	Sendaddr	Sendaddr	Sendaddr	Sendaddr
Ack	Receive ACK/NACK	Receive ACK/NACK	Receive ACK/NACK	Receive ACK/NACK
Data0	Interrupt, stretch clock until resume, receive RX data	Interrupt if TX data not available, stretch clock until TX data available, send TX data	Receive RX data	Interrupt if TX data not available, stretch clock until TX data available, interrupt, send TX data,
Ack	Interrupt, stretch clock until ACK/NACK, send ACK/NACK	Receive ACK/NACK from remote	Interrupt, stretch clock until RX data is read, ACK	Receive ACKk/NACK from remote

Table 23. I2C Slave Modes of Operation (Continued)

Mode	Auto ACK Off, Write Frame	Auto ACK, Read Frame	Auto ACK On, Write Frame	Auto ACK On, Read Frame
Data N	Receive RX data	Interrupt if TX data not available, stretch clock until TX data available, send TX data	Receive RX data	Interrupt if TX data not available, stretch clock until TX data available, interrupt, send TX data
Ack N	Interrupt, stretch clock until ACK/NACK, send ACK/NACK	Receive ACK/NACK from remote	Interrupt, stretch clock until RX data is read, ACK	Receive ACK/NACK from remote
Stop	Sendstop, interrupt, interrupt if stop interrupt is enabled	Sendstop, interrupt, interrupt if stop interrupt is enabled	Send stop, interrupt, interrupt if stop interrupt is enabled	Sendstop, interrupt, interrupt if stop interrupt is enabled

Field Name	Value Symbol	Value Description	Hex Value
REPEATED_START_INT_ENABLE	I2C_REPEATED_START_INT_DISABLE	Repeated start interrupts are not generated	0x0*
	I2C_REPEATED_START_INT_ENABLE	A repeated start interrupt is generated when a repeated start condition occurs during an active transaction in slave mode	0x1
CONNECT_IN_STANDBY	I2C_DISCONNECT_IN_STANDBY	Disconnect the I2C lines when running on standby clock	0x0*
	I2C_CONNECT_IN_STANDBY	Keep the I2C lines connected when running on standby clock	0x1
TX_DMA_ENABLE	I2C_TX_DMA_DISABLE	No TX DMA request is generated	0x0*
	I2C_TX_DMA_ENABLE	A TX DMA request is generated when new data is requested by the I2C interface	0x1
RX_DMA_ENABLE	I2C_RX_DMA_DISABLE	No RX DMA request is generated	0x0*
	I2C_RX_DMA_ENABLE	An RX DMA request is generated when new data is received by the I2C interface	0x1
TX_INT_ENABLE	I2C_TX_INT_DISABLE	No TX interrupt is raised	0x0*

Field Name	Value Symbol	Value Description	Hex Value
	I2C_TX_INT_ENABLE	A TX interrupt is raised when new data is requested by the I2C interface	0x1
RX_INT_ENABLE	I2C_RX_INT_DISABLE	No RX interrupt is raised	0x0*
	I2C_RX_INT_ENABLE	An RX interrupt is raised when new data is received by the I2C interface	0x1
BUS_ERROR_INT_ENABLE	I2C_BUS_ERROR_INT_DISABLE	No bus error interrupt is raised when an overrun is detected	0x0*
	I2C_BUS_ERROR_INT_ENABLE	A bus error interrupt is raised when an overrun occurs on the I2C interface	0x1
OVERRUN_INT_ENABLE	I2C_OVERRUN_INT_DISABLE	No overrun interrupt is raised when an overrun is detected	0x0*
	I2C_OVERRUN_INT_ENABLE	An overrun interrupt is raised when an overrun occurs on the I2C interface	0x1
STOP_INT_ENABLE	I2C_STOP_INT_DISABLE	Stop interrupts are not generated	0x0*
	I2C_STOP_INT_ENABLE	A stop interrupt is 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
SLAVE_PRESCALE	I2C_SLAVE_PRESCALE_1	Slave Standard-mode: at least 250 ns +10% data set-up time with SYSCLK = 3 MHz; Slave Fast-mode: at least 100 ns	0x0*

Field Name	Value Symbol	Value Description	Hex Value
		+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	
	I2C_SLAVE_PRESCALE_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_PRESCALE_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_PRESCALE_4	Slave Standard-Mode: at least 250 ns +10% data set-up time with SYSCLK = 12 MHz	0x3
	I2C_SLAVE_PRESCALE_5	Slave Standard-Mode: at least 250 ns +10% data set-up time with SYSCLK = 16 MHz	0x4
	I2C_SLAVE_PRESCALE_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	0x5

Field Name	Value Symbol	Value Description	Hex Value
		with SYSCLK = 48 MHz	
	I2C_SLAVE_PRESCALE_7	Slave Standard-Mode: at least 250 ns +10% data set-up time with SYSCLK = 24 MHz	0x6
	I2C_SLAVE_PRESCALE_14	Slave Standard-Mode: at least 250 ns +10% data set-up time with SYSCLK = 48 MHz	0xD
	I2C_SLAVE_PRESCALE_32	Maximum number of clock stretching cycles between SDA output and SCL release	0x1F
MASTER_ PRESCALE	I2C_MASTER_PRESCALE_3	Master mode: prescale SCL from SYSCLK by 3	0x0*
	I2C_MASTER_PRESCALE_6	Master mode: prescale SCL from SYSCLK by 6	0x1
	I2C_MASTER_PRESCALE_9	Master mode: prescale SCL from SYSCLK by 9	0x2
	I2C_MASTER_PRESCALE_12	Master mode: prescale SCL from SYSCLK by 12	0x3
	I2C_MASTER_PRESCALE_15	Master mode: prescale SCL from SYSCLK by 15	0x4
	I2C_MASTER_PRESCALE_18	Master mode: prescale SCL from SYSCLK by 18	0x5
	I2C_MASTER_PRESCALE_21	Master mode: prescale SCL from SYSCLK by 21	0x6
	I2C_MASTER_PRESCALE_24	Master mode: prescale SCL from SYSCLK by 24	0x7
	I2C_MASTER_PRESCALE_27	Master mode: prescale SCL from SYSCLK by 27	0x8
	I2C_MASTER_PRESCALE_30	Master mode: prescale SCL from SYSCLK by 30	0x9
	I2C_MASTER_PRESCALE_33	Master mode: prescale SCL from SYSCLK by 33	0xA
	I2C_MASTER_PRESCALE_36	Master mode: prescale SCL from SYSCLK by 36	0xB
	I2C_MASTER_PRESCALE_39	Master mode: prescale SCL from SYSCLK by 39	0xC

Field Name	Value Symbol	Value Description	Hex Value
	I2C_MASTER_PRESCALE_42	Master mode: prescale SCL from SYSCLK by 42	0xD
	I2C_MASTER_PRESCALE_45	Master mode: prescale SCL from SYSCLK by 45	0xE
	I2C_MASTER_PRESCALE_48	Master mode: prescale SCL from SYSCLK by 48	0xF
	I2C_MASTER_PRESCALE_51	Master mode: prescale SCL from SYSCLK by 51	0x10
	I2C_MASTER_PRESCALE_54	Master mode: prescale SCL from SYSCLK by 54	0x11
	I2C_MASTER_PRESCALE_57	Master mode: prescale SCL from SYSCLK by 57	0x12
	I2C_MASTER_PRESCALE_60	Master mode: prescale SCL from SYSCLK by 60	0x13
	I2C_MASTER_PRESCALE_120	Master mode: prescale SCL from SYSCLK by 120	0x27
	I2C_MASTER_PRESCALE_768	Master mode: prescale SCL from SYSCLK by 768	0xFF
SLAVE	I2C_SLAVE_DISABLE	Disable I2C interface slave mode operation	0x0*
	I2C_SLAVE_ENABLE	Enable I2C interface slave mode operation	0x1

For registers, see Section 11.7.1 “I2C Registers” on page 609.

11.2 LIN

The RSL15 system includes a local interconnect network (LIN) interface that allows the system to communicate with other devices via a LIN bus, which is typically used in an automobile's automotive system. The LIN module operates as a responder node on the bus, and supports version 2.2 of the LIN specification. The LIN port is an asynchronous 2-wire interface.

The LIN bus itself is a two wire interface, with one wire consisting of the LIN bus, and the other wire connected to ground. The LIN bus is a 12 V bus, and therefore cannot be interfaced directly to the RSL15 system. Instead, the RSL15's LIN interface must be connected to a LIN transceiver that converts the signals transmitted by the RSL15 system to a higher voltage that can be used on the LIN bus..

The LIN interface is multiplexed onto the GPIO pads, which can be configured as the input and output signals that the LIN interface utilizes, with the necessary physical pad configuration. For more information about configuring the multiplexed GPIO functionality, see Chapter 10 "General Purpose Input/Output" on page 511.

The LIN interface itself only has one signal, the RX signal. This must be configured when one wishes to use the LIN interface. The TX signal can be implemented using a UART (see Section 11.6 “Universal Asynchronous Receiver-Transmitter (UART) Interfaces” on page 607). The baud rate of the LIN bus is automatically retrieved from the LIN frame received. RSL15‘s LIN module supports a baud-rate of 1000-20000 b/s.

Data transfers using the LIN interface can be controlled directly by the processor, or indirectly using the DMA. The DMA can be triggered on LIN data reads via the DMA_CTRL register, using that register’s DMA_CTRL_TRIGGER_SOURCE bitfield.

11.2.1 LIN Responder Mode

The LIN module that handles responder mode operations and the receiving of data from the LIN bus is controlled by a finite state machine (FSM). The FSM is started by enabling the interface via the LIN_CTRL_ENABLE bit in the LIN_CTRL register, and the LIN_CFG_INIT bit in the LIN_CFG register. Once these two bits have been set, the device must wait for 400 clock cycles for the FSM to initialize itself. When these 400 cycles are over, the interface prepares to start receiving data by entering a WAIT_BREAK_SYNC state. In this state, the module waits for a valid break and sync pattern from the LIN bus. Once this pattern is received, the module interprets the next data as the protected identifier (PID).

Upon reception of the PID, a LIN interrupt is fired and the LIN_CTRL_RHC bit in the LIN_CTRL register is set. In the LIN interrupt, the firmware is responsible for interpreting the PID and preparing the LIN module appropriately. Each unique PID has a different interpretation based on the design of the system. Each PID usually signifies if the current LIN frame is a read or write packet, as well as the length of the packet. Some PIDs include specially reserved values; these can be found by reviewing the LIN specification.

NOTE: When the LIN_CTRL register is read, it resets the value of the status bits.

Configuring the LIN module after interpreting the PID is a straightforward process. In both read and write frames, the length of the message is set by writing to the corresponding register. For writes, the frame length in bytes is written to the LIN_DLB_DLBT field of the LIN_DLB register. For reads, the frame length in bytes is written to the LIN_DLBR_DLBR field in the LIN_DLBR register. If writing data, the data to be written must then be written to the LIN_DATA bank of registers, starting at LIN_DATA0. The LIN_DATA_WORD* registers can also be used to provide a method for writing word sized data. Once the LIN_DATA* registers are written to, the LIN module handles either clocking out the LIN data on the TXD pin (via UART), or clocking the LIN data in on the RXD pin.

If the frame is a receive frame, another interrupt fires when the number of bytes set in the LIN_DLBR register are read. At this point, the LIN_CTRL_RXF bit in the LIN_CTRL register is set, and received data can be read from the LIN_DATA* registers. The LIN frame is now complete.

11.2.2 Lin Commander Mode

The LIN commander node is responsible for starting and constructing the LIN frame. The LIN bus is pulled-up by default; therefore, a LIN frame consists of a break that includes 14 bits of low data. This BREAK sequence is followed by an alternating pattern of high and low levels, lasting eight bits (0x55). This is the SYNC sequence. These two sequences form the break-sync sequence, which indicates the start of a LIN frame. The LIN module looks for this pattern so it can start reception of the LIN frame. After the break, the commander device is responsible for sending the next byte, which contains the parity bits and PID, via UART. For information on how to calculate the parity bits, consult the LIN specification.

Since all the data that the LIN transceiver receives on its TXD pin is reflected back onto the RXD pin, the commander node starting the transmitted frame receives the same break sync sequence back, which triggers the LIN interrupt. At this point, the commander device responds to the interrupt in the same manner that responder mode does.

For registers, see Section 11.7.2 “LIN Registers” on page 618.

11.3 Pulse Code Modulation (PCM) Interface

RSL15 has access to one pulse code modulation (PCM) interface, which can be used to stream control, configuration or signal data into and out of the microcontroller.

The PCM interface connects to the processor through the Arm Cortex-M33 processor's peripheral bus. There are two possible ways the PCM interface can handle transmission and reception buffers:

- By using the internally available data transmission and reception interrupts. These interrupts can be enabled by setting the `PCM_CFG_RX_TX_INT_ENABLE` bit in the `PCM_CFG` register.
- By connecting to the DMA with two channels supporting transmit and receive operations. The data request lines to the DMA can be enabled by setting the `PCM_CFG_TX_DMA_ENABLE` and `PCM_CFG_RX_DMA_ENABLE` bits in the `PCM_CFG` register.

The "High-Level PCM Interface Module Block Diagram" figure (Figure 54) shows a high-level view of the PCM interface.

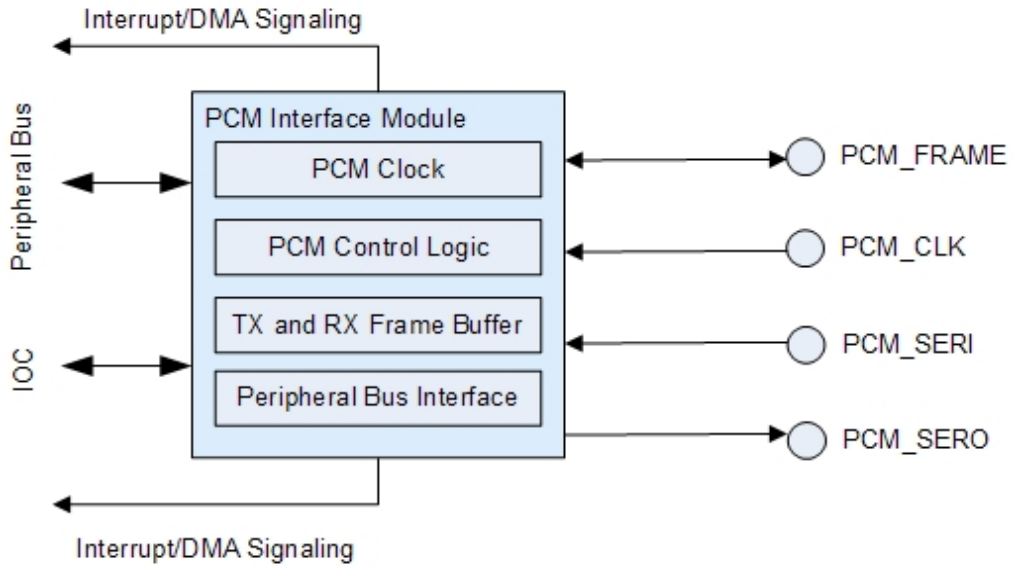


Figure 54. High-Level PCM Interface Module Block Diagram

The PCM interface is multiplexed onto the GPIO pads, which can be physically configured for the input and output signals that form the PCM interface. For more information about configuring the multiplexed GPIO functionality, see Section 10.2 “Functional Configuration” on page 512.

These interfaces make use of four external signals in communications. These signals are:

- An 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 clock unless this clock is sourced within the RSL15 system. Similarly, disabling the bus pull-up and pull-down resistors is not recommended for the PCM frame signal in slave mode. For all other configurations, attempting to use this interface without the pull resistors can cause 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. When using the PCM interface in any mode, set this bit and ensure that the proper GPIO multiplexing has been selected.

NOTE: While the PCM interface is disabled, the internal data registers are held at their current state, and the internal clocks are gated. Enabling the interface resumes the state machine of the interface, and disabling the interface pauses the interface's state machine. Note that the DMA is not gated by the interface enable, so a DMA request can be generated with the PCM interface disabled.

The PCM interface can be configured as either a master device (controlling the frame signal, and potentially providing the interface PCM_CLK through an internally routed GPIO function), or 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 in the PCM_CTRL register as appropriate. 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 in the following ways:

- Internally, selecting a clock source as the GPIO output mode for the pad that is also used for the PCM clock, to route an internal clock signal within the RSL15 system to the same pad
- Externally, using an externally generated clock signal

The PCM clock frequency can be the same as the system clock frequency. External clocks can also be used to drive the PCM interfaces; these clocks can be sourced via an external PCM device, or one of the output clocks from the RSL15 system can be used (e.g. USRCLK).

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_CFG_CLK_POLARITY bit from the PCM_CFG register. By default, the PCM interface samples its inputs on the falling edge of the PCM clock, outputting its signals on the rising clock edge. When configuring the PCM interface for I^2S standard compatibility mode, we recommend that you configure the interface to sample on the rising edged of the clock. This is done by setting the PCM_CFG_CLK_POLARITY bit in the PCM_CFG register.

A PCM interface can be reset using the PCM_CTRL_RESET bit from the PCM_CTRL register. This resets the internal data registers, clocks, and state machine on the next PCM clock edge. Reset the interface to restart in a known good state after any error is detected.

IMPORTANT: The PCM interface configuration and internal status registers are tightly synchronized to the input PCM clock. As a result, the PCM interfaces are only reset, and the internal configurations of the PCM interfaces are only updated, on their selected PCM clock edges.

11.3.1 PCM Signal Configuration

The PCM interface uses a large number of configuration options to help 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 the "Signal Timing Diagram: Default PCM Configuration" figure (Figure 55), below.

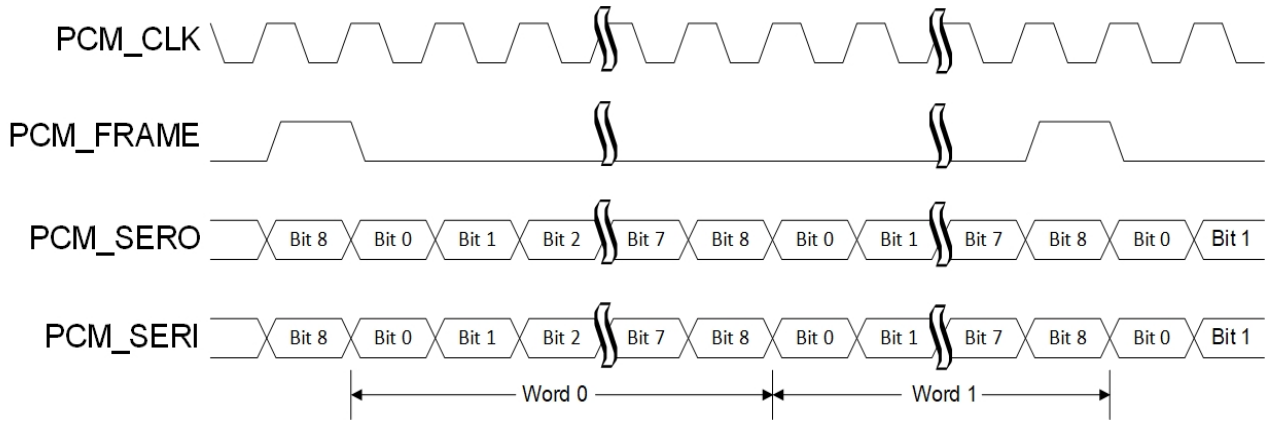


Figure 55. Signal Timing Diagram: Default PCM Configuration

For more information about the required signal configuration and other interface configurations that are needed for interface compatibility with I^2S, including an example timing diagram, see Section 11.3.4 “I2S Configuration and Usage” on page 597.

IMPORTANT: To ensure defined behavior, the PCM interface needs to be in IDLE state when changing any configuration settings.

11.3.1.1 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 `PCM_CFG_SUBFRAME` bit of the `PCM_CFG` register is enabled). Each aspect of the PCM frame signal can be configured as follows (all bits are in the `PCM_CFG` register):

Interval

In the default configuration of the `PCM_CFG_FRAME_LENGTH` bit field (configured for two words per frame) and the `PCM_CFG_SUBFRAME` 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 `PCM_CFG_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 therefore the number of words per data frame. The `PCM_CFG_FRAME_LENGTH` bit field is not used for PCM slave mode; it needs to be left in the default configuration (configured for two words per frame), regardless of the actual frame length used, to produce the expected behavior. If the `PCM_CFG_SUBFRAME` bit is configured to enable sub-frames, a frame signal is generated with every word of each data frame.

IMPORTANT: Configuring the PCM_CFG_FRAME_LENGTH bit field to something other than two words per frame, when setting the PCM_CFG_SUBFRAME bit, results in a configuration where the user receives a PCM frame signal for every sub-frame word. Despite the well-defined behavior of the frame signal generated for this configuration, the handling of data and interrupts by the PCM interfaces 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 PCM_CFG_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. Alternatively, 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 PCM_CFG_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_CLK 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 the "Frame Signal Timing Examples" figure (Figure 56). In all cases, the PCM_CFG_FRAME_ALIGN bit has been set to PCM_FRAME_ALIGN_FIRST (which differs from the default configuration shown in the "Signal Timing Diagram: Default PCM Configuration" figure (Figure 55)), to prevent any confusion as to the effects of the other frame signal configuration bits. The frame signal configurations shown are:

1. Configured to indicate the sub-frame words of a frame with a short pulse frame signal
2. Configured to indicate the sub-frame 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

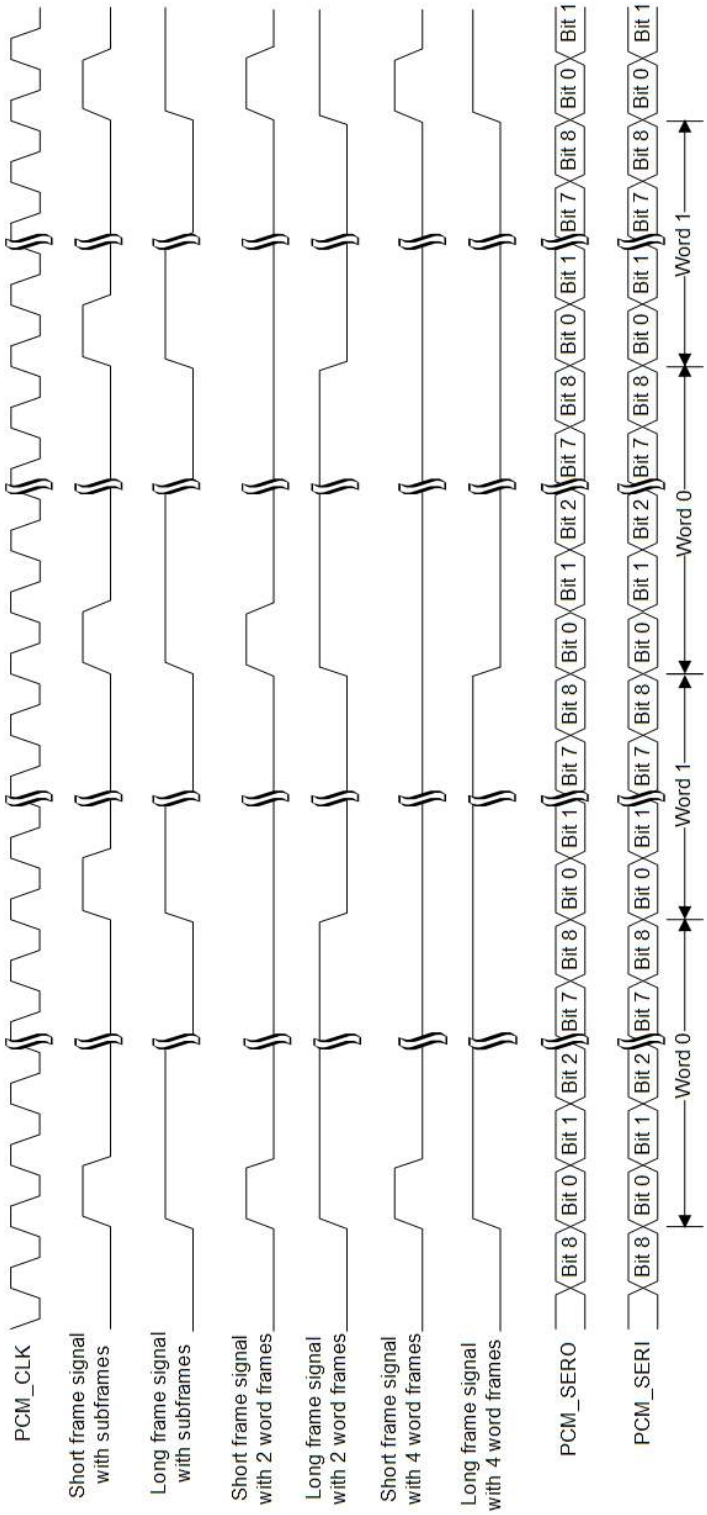


Figure 56. Frame Signal Timing Examples

The "PCM Frame Signal Configuration Options" table (Table 24) summarizes the options for PCM_FRAME signal configuration.

Table 24. PCM Frame Signal Configuration Options

Configuration Option	Settings	Description
Sub-frames (Master mode only)	Sub-frames disabled (default), sub-frames enabled	When sub-frames are enabled, the PCM interface provides a frame signal for every word in the frame.
Frame Size	2 (default), 4, 6, 8, 10, 12, 14 or 16 words by frames	Specifies the number of words that are expected in each frame (only applicable when sub-frames are disabled).
Frame signal width	Short (default), Long	A short PCM frame signal is held high for one PCM clock cycle. A long PCM frame signal is held high for one half of the PCM frame. NOTE: If sub-frame is enabled and the word size is odd, the frame signal is held high for a portion of the frame equal to half of the word size rounded down.
Alignment	Align to first data bit, align to last data bit (default)	The first (and possibly only) cycle in which the PCM frame signal is high; it is aligned with the specified data bit (default)

11.3.1.2 Data Serial Input and Output Configuration

The PCM interface allows data bits to be configured for transmission and reception in either an MSB first or an LSB first ordering. To select between these two configurations, set the PCM_CFG_BIT_ORDER bit in the PCM_CFG register.

The length of each word of a PCM transaction can be set to be between 8 and 32 bits per word, by setting the PCM_CFG_WORD_SIZE bit field in the PCM_CFG register. When transmitting data that uses less than 32 bits per word, the PCM_CFG_TX_ALIGN bit from the PCM_CFG 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).

The "PCM Data Configuration Options" table (Table 25) summarizes the options available for data configuration.

Table 25. PCM Data Configuration Options

Configuration Option	Settings	Description
Data ordering	MSB first (default), LSB first	Selects whether the data is transmitted starting with the MSB or LSB
Word size	8- to 32-bit (default) by words	Specifies the number of bits in each PCM word
Transmit and receive bit-field	MSB bit-field (default), or the LSB bit-field	Selects the data alignment (MSB or LSB)

11.3.2 PCM/I^{2}S Data Interface

Each PCM interface has two single word transmit registers called PCM_TX_DATA*, and two single word receive registers called PCM_RX_DATA*. The first word received after the rising edge of the frame is stored in the PCM_RX_DATA0 register, and the second word is stored in the PCM_RX_DATA1 register. Similarly, the PCM_TX_DATA0 register is transferred first, followed by the PCM_TX_DATA1 register.

If the word size is greater than 2, both receive registers are overwritten with the next words, and the data in both transmit registers is sent. The order of capture and send is always the same.

NOTE: There are always two words transmitted or received per frame, never only one.

11.3.3 PCM Interrupt Configuration

Each PCM interface uses two associated interrupts. These interrupts control transmission and reception of PCM data, and report any errors that have occurred while transmitting or receiving PCM data. These interrupts can be configured by setting the PCM_CFG_RX_TX_INT_ENABLE, PCM_CFG_OVERRUN_INT_EN, and PCM_CFG_UNDERRUN_INT_EN bit fields in the PCM_CFG register. See Section 3.3.2 “Nested Vector Interrupt Controller (NVIC)” on page 58 for information regarding interrupt configuration and handling interrupts for the Arm Cortex-M33 processor.

If a user application is transmitting data from a PCM interface, the PCM_RX_TX interrupt sends signals at the following times:

- When the PCM_TX_DATA* registers' value starts to be transmitted using the PCM interface
- When the PCM_STATUS_TX_REQ bit of the PCM_STATUS register is set, signifying that the next data word(s) (of the specified size) to be transmitted can now be loaded into the PCM_TX_DATA* registers by the application
- When the PCM_STATUS_RX_REQ bit of the PCM_STATUS register is set, signifying that a data word(s) of the specified size has been successfully received and written to the PCM_RX_DATA* registers

The PCM transmit and receive interrupts are generated for every two words of data transmitted in a valid PCM frame, regardless of the length of the PCM frame. The data register that acknowledges the transmission or reception can be configured by setting the PCM_CFG_TX_ACK_SEL and PCM_CFG_RX_ACK_SEL bit fields of the PCM_CFG register.

The PCM_ERROR interrupt indicates one of the following conditions:

- An overrun has occurred: the received data from the PCM_RX_DATA* registers has not been read before being overwritten. The PCM_STATUS_OVERRUN bit in the PCM_STATUS register can be read for overrun status.
- An underrun has occurred: new data has not been copied to the PCM_TX_DATA* registers before the occurance of the next PCM transmission. The PCM_STATUS_UNDERRUN bit in the PCM_STATUS register can be read for underrun status.

NOTE: The status bits for overrun and underrun errors in the PCM_STATUS register can be cleared by setting the associated clear bits in the same register.

The "PCM Interrupt Options" table (Table 26) summarizes the options available for PCM interrupt configuration.

Table 26. PCM Interrupt Options

Interrupt Type	Signal Name	Status Register Name	Description
Data received	pcm_rx_tx_irq	RX_REQ_STATUS	Interrupt when a new data word is ready to be read by processor
Data sent		TX_REQ_STATUS	Interrupt when a data word has been sent, transmit register is ready for a new value
Overrun detected	*pcm_err_irq*	OVERRUN_STATUS	A new data word has been received on the interface before the last received data has been read
Underrun detected		UNDERRUN_STATUS	A new data word has been sent on the interface before previous data has been written on the transmit register

11.3.4 I²S Configuration and Usage

The PCM interface can be configured to be compatible with the I²S interface standard. The "Required Configuration Settings for I2S Configuration" table (Table 27) lists the required signal and data management settings for the PCM interface to enable I²S communication.

Table 27. Required Configuration Settings for I²S Configuration

Configuration Bit or Bit Field	Setting
PCM_CFG_FRAME_LENGTH	PCM_MULTIWORD_2
PCM_CFG_FRAME_WIDTH	PCM_FRAME_WIDTH_LONG
PCM_CFG_FRAME_ALIGN	PCM_FRAME_ALIGN_LAST
PCM_CFG_BIT_ORDER	PCM_BIT_ORDER_MSB_FIRST
PCM_CFG_SUBFRAME	PCM_SUBFRAME_DISABLE
PCM_CFG_CLK_POLARITY	PCM_SAMPLE_RISING_EDGE[1]

In master mode, the I²S configuration for the PCM interface has no effect.

In slave mode, the behavior of the PCM interfaces is slightly different from how it acts in any other configuration. When configured for slave mode and a long frame width, the PCM interfaces synchronize 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). This means that if the word size used by the master is larger than the size configured in the corresponding PCM interface, the interface pads the remaining bits with zeroes for both words. If the size is less than the size configured, the interface only sends and receives the MSB bits of both words.

As shown in the I²S signal timing diagram (see the "Signal Timing Diagram: I2S Configuration" figure (Figure 57)), after transmitting the LSB of the current data word, the device repeatedly transmits zeros 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 I²S signals that transmit a number of bits different from the number specified in the defined word size.

[1]The PCM interfaces produce a data stream that is I²S standard compliant 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.

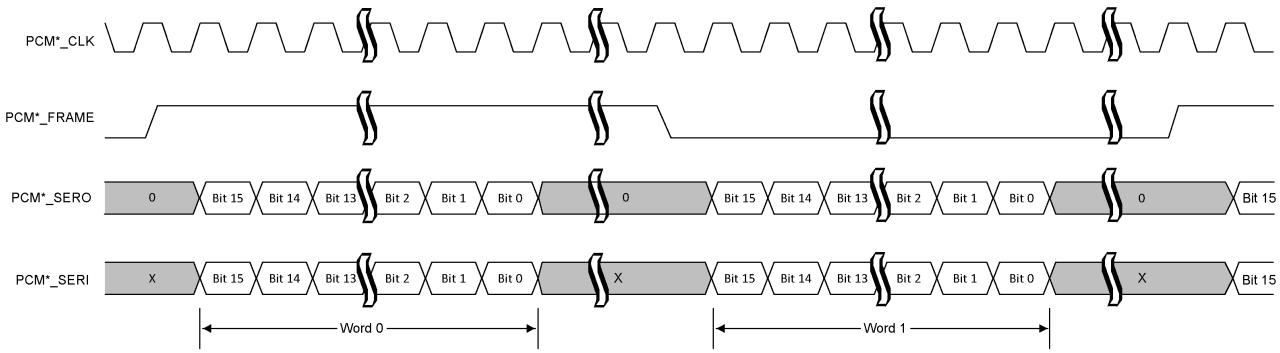


Figure 57. Signal Timing Diagram: I²S Configuration

For registers, see Section 11.7.3 “PCM Registers” on page 624.

11.4 PWM

The RSL15 system contains five pulse-width modulator (PWM) drivers, which 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.

The PWM drivers are multiplexed onto the GPIO pads, which can be configured as output signals with the necessary physical pad configuration (`GPIO_MODE_PWM*`). The GPIOs support output of the PWM signals with the specified period and high-time in each period and the inverse of the specified signal (`GPIO_MODE_PWM*_INV`). For more information about configuring the multiplexed GPIO functionality, see Chapter 10 "General Purpose Input/Output" on page 511.

The timing and shape of the signal produced by each PWM is defined by clock signals that are divided from SYSCLK or SLOWCLK, and by the `PWM_PERIOD*` register and `PWM_HIGH*` register.

The PWM drivers are supported by a clock (PWM*CLK) that is sourced from SLOWCLK or SYSCLK using the `CLK_DIV_CFG2_PWM_CLK_SRC` bit field from the `CLK_DIV_CFG2` register.

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 ($\mathbf{PWM\_CFG\_PWM\_PERIOD + 1}$) PWM*CLK cycles in length, with a maximum length of 4096 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 ($\mathbf{PWM\_CFG\_PWM\_HIGH + 1}$) PWM*CLK cycles of each period, to a maximum equal to the period of that PWM signal. After ($\mathbf{PWM\_CFG\_PWM\_HIGH + 1}$) 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 58 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%.

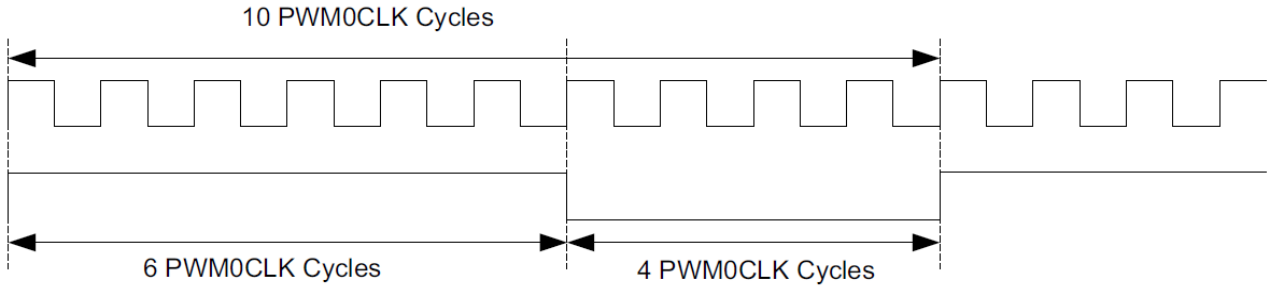


Figure 58. PWM Sample Configuration

Each PWM driver can be independently enabled and disabled by configuring the PWM_CTRL_PWM*_ENABLE bit field from the PWM_CTRL register.

The PWM drivers also support dithering. Each PWM channel has 8-bit dithering based on the van der Corput sequence. To configure dithering, the PWM_HIGH register's PWM_HIGH*_HIGH_FRACTIONAL bitfield of the corresponding driver must be set to a value higher than 1, to a maximum of 255.

NOTE: PWM channel 0 must be active to have dithering on any PWM channel.

When dithering is active, the PWM driver adds or subtracts a single clock cycle over a PWM_PERIOD, where the total amount of cycles added over 256 periods is equal to the value set in the PWM_HIGH*_HIGH_FRACTIONAL bitfield in the PWM_HIGH register.

If a PWM driver uses the same period and clock divisor as PWM0, the relative timing between the PWM driver and PWM0 can be defined. To enable offset configuration, set the PWM_OFFSET*_OFFSET_ENABLE bit from the PWM_OFFSET* register. When this configuration is enabled, the number of cycles between the rising edge for PWM0 and the rising edge for the configured PWM* is equal to the value of the PWM_OFFSET*_OFFSET bit field from the PWM_OFFSET* 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 PWM* does not share its period and clock divisor configuration with PWM0.

11.4.1 ACS_PWM

The ACS-PWM block is another PWM driver that can output a signal during low power modes when most of the device is powered down. This is possible because ACS-PWM is part of the Analog Control Sub System (ACS), which is always on. The ACS-PWM is mapped to GPIO4 and cannot be changed. Before enabling the ACS-PWM, GPIO4 must be set to GPIO_MODE_DISABLE and GPIO_NO_PULL.

The ACS-PWM clock is sourced from the 32,768 Hz RTCCLK.

The ACS-PWM is enabled via the ACS_PWM_AO_CTRL_PWM_ENABLE bit in the ACS_PWM_AO_CTRL register. This register is also used to disable the module and reset the counter via the ACS_PWM_AO_CTRL_PWM_DISABLE and ACS_PWM_AO_CTRL_PWM_RESET bits appropriately.

The ACS-PWM block functions similarly to the standard PWM module, except that the ACS-PWM's period and high states are limited to a range of 1 to 256, instead of 1 to 4096, due to the 8-bit size of the ACS_PWM_AO_CFG_PERIOD bit field and ACS_PWM_AO_CFG_HIGH bit field in the ACS_PWM_AO_CFG register. For information on how the PERIOD and HIGH bit fields configure the PWM waveform, see Section 11.4 "PWM" on page 598.

For registers, see Section 11.7.4 "PWM Registers" on page 630.

11.5 Serial Peripheral Interfaces (SPI)

The RSL15 system includes two Serial Peripheral Interfaces (SPIs). The SPI is a synchronous 4-wire interface that includes clock, chip select, and IO pads. The SPI supports single and dual I/O modes, as well as the ability to add two additional I/O pins in a quad I/O mode. These SPI interfaces allow the system to communicate with external components, including external analog front ends, external controllers, wireless transceivers, and non-volatile memories (NVMs).

NOTE: The SPIs support master/slave configuration as well as half/full duplex mode. Master mode supports single, dual, and quad I/O configurations as well as configurable clock polarity and active phase configurations. Slave mode supports only the single I/O configuration and configuration of the clock's active phase.

The SPI interfaces are multiplexed onto the GPIO pads, which can be configured for the input and output signals that form each SPI by using the necessary physical pad configuration. For more information about configuring the multiplexed GPIO functionality, see Section 10.2 "Functional Configuration" on page 512.

The SPI interfaces can be enabled or disabled using the SPI_CTRL_ENABLE and SPI_CTRL_DISABLE write only bits in the SPI*_CTRL registers. When the SPI_CTRL_DISABLE bit is set to 1, any ongoing transactions on the specified SPI interface are completed and the interface is disabled.

Data transfers using the SPI interfaces can be controlled directly by the host processor, or indirectly by using the DMA. To enable DMA control, use the SPI_CFG_TX_DMA_ENABLE and SPI_CFG_RX_DMA_ENABLE bits in the SPI*_CFG registers. The RX/TX interrupts and RX/TX DMA requests can be generated concurrently.

11.5.1 SPI Data Transfers

The SPI interfaces can be configured to operate as an SPI master device or an SPI slave device by setting the SPI_CFG_SLAVE bit in the SPI*_CFG register to 0 or 1 respectively. When configured as SPI master, the SPI_CFG_MODE bit field selects the number of data pins used for communication. In slave mode, this field has no effect and the mode is always set to SPI_MODE_SPI (two separate RX and TX data pins). See the definition of SPI*_IO[0:3], below in this topic, for an explanation of the data pins used for communication.

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 RSL15. The SPI*_CLK signal is derived from SYSCLK using a power of two prescaler (2^1 to 2^{10}), configurable via the SPI_CFG_PRESCALE bit field from the SPI*_CFG register, using the following equation:

$$f_{SPI*\_CLK} = \frac{f_{SYSCLK}}{2^{(SPI*\_CFG\_PRESCALE+1)}}$$

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:

$$f_{SYS\,\mathrm{CLK}} \geq 4 \times f_{\mathrm{SPI*\_CLK}}$$

Due to the re-timing of the input SPI*_CLK, the output on SPI*_IO1 of the SPI slave pad, which is generated by writing to the `SPI*_TX_DATA` register, is delayed up to one SYSCLK period when:

$$f_{SYS\,\mathrm{CLK}} = 4 \times f_{\mathrm{SPI*\_CLK}}$$

Output can be delayed up to one half of a clock cycle when:

$$f_{SYS\,\mathrm{CLK}} = 5 \times f_{\mathrm{SPI*\_CLK}}$$

No delay in output (i.e., the output delay only consists of logic and pad delays) occurs when:

$$f_{SYS\,\mathrm{CLK}} \geq 6 \times f_{\mathrm{SPI*\_CLK}}$$

For both master and slave mode, the `SPI_CFG_CLK_PHASE` bit in the `SPI*_CFG` registers is used to control the times when data changes and when data is sampled. An SPI using normal phase updates output signals on the falling edge of SPI*_CLK, and samples input signals on the rising edge of SPI*_CLK. If the phase is inverted, output signals change on the rising edge of SPI*_CLK and input signals are sampled on the falling edge. The `SPI_CFG_CLK_POLARITY` bit in the `SPI_CFG` register can be used to set the idle state of SPI*_CLK. Setting `SPI_CFG_CLK_POLARITY` bit to 0 causes SPI*_CLK to idle low, while setting the bit to 1 causes SPI*_CLK to idle high.

NOTE: This is slightly different from, though still compatible with, the typical definition of SPI modes 0-3, in that the SPI phase is defined in terms of rising and falling edges, instead of leading and trailing edges. To convert between standard SPI modes and these registers, use the "SPI Mode to Register Conversion" table (Table 28)

Table 28. SPI Mode to Register Conversion

SPI_Mode	SPI_CFG_CLK_POLARITY	SPI_CFG_CLK_PHASE
0	0	0
1	0	1
2	1	1
3	1	0

*SPI*_CS*

In master mode, the SPI*_CS pad is an output controlled by the `SPI_CTRL_CS*` bits from the `SPI*_CTRL` registers. The signal from this pad is generally routed to the chip select input of a slave device. Write to `SPI_CTRL_CS_1` to set, or write to `SPI_CTRL_CS_0` to clear.

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 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 required between a falling edge on the SPI*_CS pad and the first edge on SPI*_CLK pad is:

$$\frac{1}{2} \cdot \text{SPI*\_CLK Period} + 2 \cdot \text{SYSCLK Periods}$$

*SPI*_IO[0:3]*

These pads are used as data pins for the SPI. The `SPI_CFG_MODE` bit field in the `SPI*_CFG` register is used to configure the number of data pins used in SPI master mode. The SPI slave operates only in SPI normal mode; therefore, only SPI*_IO0 and SPI*_IO1 pins are available. These pads are used to receive or transmit data when the following conditions are met:

- SPI*_CLK and SPI*_CS indicate that data needs to be transferred.
- The `SPI_CTRL_MODE*` bits from the `SPI*_CTRL` register are configured for an SPI read, write, or read-write (full-duplex) transfer.

The SPI timing for half and full duplex modes is shown in the "SPI Slave Mode Timing" figure (Figure 59).

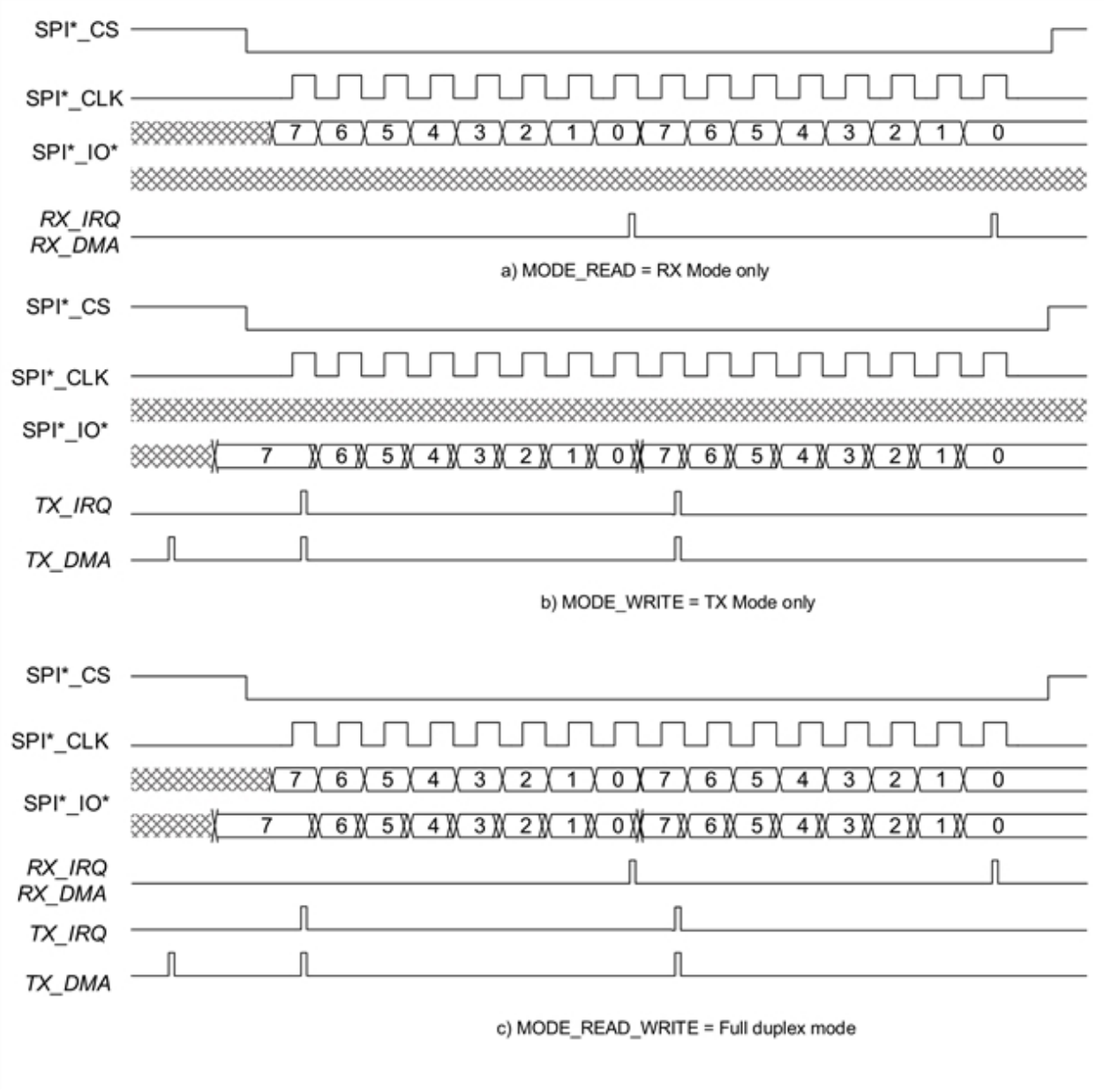


Figure 59. SPI Slave Mode Timing

The SPI operation is configured and controlled by the `SPI*_CFG` and `SPI*_CTRL` registers. Write or read access to these registers is not restricted when the SPI is active or idle. All updates in these registers are applied immediately, excluding the `SPI_CFG_CLK_POLARITY` and `SPI_CFG_WORD_SIZE` fields in the `SPI*_CFG` register. Updating these registers with the same field values as their original content does not corrupt SPI communication, even if the SPI is active. As a result, you can update only the fields of interest without affecting the functionality associated with the remaining fields. It is recommended that you only update these registers when the SPI is idle.

The SPIs support a configurable word size for each transfer of 1 to 32 bits of data per word, as specified in the SPI_CFG_WORD_SIZE bit field in the SPI*_CFG register. All data transactions using an SPI start with the MSB of the word received. During such transactions, (SPI_CFG_WORD_SIZE + 1) bits of data per word are transferred.

- If the SPI is busy, the update to this register is not transferred to the active counter until the countdown reaches zero and the SPI is idle.
- In slave mode, the WORD_SIZE might not be updated during the transmission of the last two bits.
- In dual mode, the WORD_SIZE is required to be a multiple of 2. If the value written is not a multiple of 2, it is automatically reset to the next multiple of 2.
- In quad mode, the WORD_SIZE is required to be a multiple of 4. If the value written is not a multiple of 4, it is automatically set to the next multiple of 4.

Fields in the SPI*_CTRL register are used to select either half duplex or full duplex mode, as follows:

- Set the SPI*_CTRL_MODE_NOP field for no read or write operation
- Set the SPI*_CTRL_MODE_WRITE field for write-only data operation
- Set the SPI*_CTRL_MODE_READ field for read-only data operation
- Set the SPI*_CTRL_MODE_READ_WRITE field for read and write data operation (full duplex in normal SPI mode)

NOTE: In DSPI and QSPI modes, this last setting is not valid. The application must set MODE_WRITE and MODE_READ during different phases of the transaction.

In SPI master mode, a read transaction is started when the SPI*_RX_DATA register is read, or when the SPI_CTRL_START bit in the SPI*_CTRL register is set to 1. A write or a read/write operation is started when the TX buffer is full. If it is empty, the transfer starts when data is written to the SPI_TX_DATA register. The SPI interface has parallel buffers for both transmitted and received data.

SPI*_TX_DATA
: Shift register containing the data to transmit using the SPI

SPI*_RX_DATA
: Shift register containing the recent data word received from the SPI

Two additional registers can be used to read the data received:

SPI*_RX_DATA_NO_START
: This register is for reading the received data without starting a new read transaction when in read mode. Read request flag SPI_STATUS_RX_REQ in the SPI*_STATUS register is cleared automatically on reading the SPI*_RX_DATA_NO_START register.

SPI*_RX_DATA_MIRROR
: This register is for reading the received data without starting a new read transaction when in read mode and without clearing the read request flag. It is used for debugging without modifying the behavior of the program.

The SPI*_STATUS register can be used by an application to identify the state of the SPI interface. The states are as follows:

- The SPI_STATUS_BUSY bit in the SPI*_STATUS register indicates that the reception or transmission of the data is ongoing. This bit is useful for determining when a transaction has completed on the bus. The application can use the SPI_STATUS_TX_REQ and SPI_STATUS_RX_REQ bits in this register to determine when to read and write to or from the interface.
- When the SPI_STATUS_TX_REQ bit is set to 1, the SPI is ready to accept new data in the SPI*_TX_DATA register for transmission. The application must wait for this bit to be set before writing to the SPI*_TX_DATA register.
- The SPI_STATUS_RX_REQ bit in the SPI*_STATUS register indicates when new data is available to be read. The application must wait for this bit to be set before reading from the SPI*_RX_DATA register to ensure that the data is valid.
- The SPI_STATUS_CS_RISE bit in the SPI*_STATUS register indicates that a rising edge has been detected on the SPI*_CS pin. An SPI slave can use this interrupt to determine whether an SPI master is ending a data transfer.
- The SPI_STATUS_OVERRUN and SPI_STATUS_UNDERRUN interrupts in the SPI*_STATUS register are used to determine whether a buffer overrun or buffer underrun condition has occurred during SPI communications.
- SPI_STATUS_TX_REQ_SET in the SPI*_STATUS register is provided in slave mode when a master ends a transaction by setting CS high (SPI). At this point the SPI*_STATUS_TX_REQ flag in the same register is zero, which indicates that there is still a byte pending in the TX buffer. To clear this state without resetting the interface, SPI_STATUS_TX_REQ_SET can be set high to request new data. However, this does not generate a new interrupt (TX); only the underlying buffer is flushed empty.

NOTE: When the interface is disabled, any ongoing data transmission is allowed to complete before the interface shuts down.

NOTE: In slave mode, the first value to be transmitted in a frame (i.e., after the CS line goes low) needs to be written into the buffer prior to the CS going low. Therefore, after initializing the SPI interface in slave mode, the SPI*_TX_DATA register must be written to before an externally-connected master initiates a read transaction. Otherwise the SPI_STATUS_UNDERRUN bit in the SPI*_STATUS register is set to indicate that an underrun has occurred.

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.

11.5.2 SPI Interrupts

Each of the SPI interfaces uses three interrupts that can indicate the following four events:

- The start of a data transmission (TX interrupt)
- The completion of a data transmission (TX interrupt)
- The completion of data being received (RX interrupt)
- Buffer overrun/underrun/CS rise events (SPI communication interrupt)

See Section 3.3.2 “Nested Vector Interrupt Controller (NVIC)” on page 58 for information regarding interrupt configuration and handling interrupts for the Arm Cortex-M33 processor.

If a user application is transmitting data using an SPI interface, the SPI*_TX_START interrupt indicates the following:

- 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

The SPI*_TX_END interrupt can also be used when transmitting data. This interrupt indicates when the transmission is finished, which can be useful when transmitting the last data word at the end of a transmission. This lets the application know when it is safe to reconfigure or disable the SPI interface, or to transition into a low power mode.

If a user application is receiving data from an SPI interface, the SPI*_RX interrupt indicates that a data word of the specified size has been successfully received and written to the SPI*_RX_DATA register.

SPI*_COM interrupts indicate buffer overrun, underrun, and a rising edge transition on the SPI*_CS pin. An overrun occurs when another data word has been loaded into the SPI*_RX_DATA register prior to the SPI*_RX_DATA register being read. An underrun occurs when data is taken from the SPI*_TX_DATA register for transmission before the SPI*_TX_DATA register has been updated with new data to be transmitted. A CS rise is detected when the interface is enabled in slave mode and the CS goes from low to high.

The status of the different interrupts can be monitored in the SPI*_STATUS register. The different status bits can be reset by writing 1 to the corresponding clear bit locations.

NOTE: In slave mode, if CS rises before a word transaction is completed, that transaction is aborted and the data is lost.

11.5.3 SPI DMA Control

The SPI can operate in DMA mode by enabling the bits SPI_CFG_TX_DMA_ENABLE and/or SPI_CFG_RX_DMA_ENABLE in the SPI*_CFG register. This allows the SPI to communicate with the DMA controller, which responds to requests by writing or reading to or from the data register.

NOTE: The DMA requests are not gated by the SPI*_STATUS_ENABLE bit of the SPI interface. Therefore, for TX_DMA transfers, the SPI causes the data to be transferred from the DMA into the SPI*_TX_DATA register even if the SPI is disabled. However, the data itself is not shifted out on the SPI until the interface is enabled.

Additionally, while using the DMA to control data transfers over an SPI interface, the SPI*_COM interrupts indicate when an error has occurred.

- An overrun occurs when the DMA cannot read the received data from the SPI*_RX_DATA register and transfer this data to the DMA buffer before it is overwritten. To monitor for overrun events, set the SPI_CFG_OVERRUN_INT_ENABLE bit in the SPI*_CFG register. If an overrun occurs and this event monitor is enabled, the SPI_STATUS_OVERRUN 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_CFG_UNDERRUN_INT_ENABLE bit in the SPI*_CFG register. If an underrun occurs and this event monitor is enabled, the SPI_STATUS_UNDERRUN bit in the appropriate SPI*_STATUS register is set.

For registers, see Section 11.7.5 “SPI Registers” on page 633.

11.6 Universal Asynchronous Receiver-Transmitter (UART) Interfaces

The general-purpose Universal Asynchronous Receiver-Transmitter (UART) interface provides support for communicating with devices that use standard UART and RS-232 transmission protocols.

The UART interface is multiplexed onto the GPIO pads. These can be configured for 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, and drive strength configuration for the TX signal). For more information about configuring the multiplexed GPIO functionality, see Chapter 10 "General Purpose Input/Output" on page 511.

The UART interface can be enabled using the `UART_CTRL_ENABLE` bit or disabled using the `UART_CTRL_DISABLE` bit, both in the `UART_CTRL` register.

NOTE: When the `UART_CTRL_DISABLE` bit is set to 1, any ongoing transaction, TX or RX, is completed before disabling the interface, and the UART transmit line `UART_TX` is set high.

A reset can be performed on the UART interface by writing to the `UART_CTRL_RESET` bit in the `UART_CTRL` register. This causes the UART interface to be disabled, and the register values to be reset to their power-on defaults.

The UART interface operates in half- or full-duplex mode using a standard data format of one start bit, eight data bits, one stop bit, and no parity bits. All data bytes being sent or received are interpreted as starting with the LSB. The "UART Transaction Waveform" figure (Figure 60) shows the waveform for a UART transmit or receive transaction.

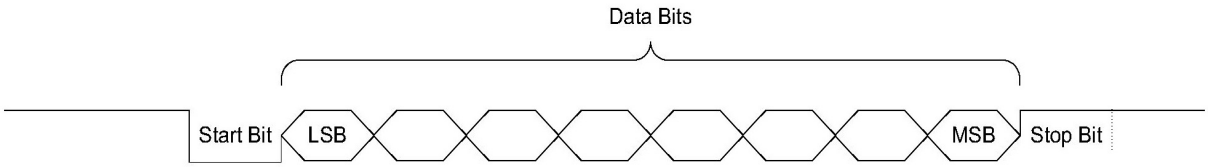


Figure 60. UART Transaction Waveform

Data transmission is started by writing 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_CTRL` register.

The UART interface runs on UARTCLK, which is sourced from a scaled version of SYSCLK. UARTCLK is scaled by setting the `UARTCLK_PRESCALE` bits in the `CLK_DIV_CFG0` register. Definitions for standard baud rates are provided for the `UART_CFG_CNT_STEP`bit field in the `UART_CFG` register.

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, it is recommended that all UART communications use a clock with a maximum of 2% error versus the expected target frequency.

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, with UARTCLK frequency in Hz:

$$\text{CNT\_STEP} = \text{round}\left(\frac{\text{baud\_rate x 262144}}{f_{UARTCLK}}\right) - 1$$

For a UARTCLK frequency of 1 MHz, the baud rate is determined as follows:

$$\text{baud\_rate} = (\text{CNT\_STEP} + 1)\,\text{x}\,3.81469727$$

$$\text{CNT\_STEP} = \text{round}\left(\frac{\text{baud\_rate}}{3.81469727}\right) - 1$$

For example, if a baud rate of 115200 bps is desired, the value of CNT_STEP must be set to 30198 for a UARTCLK frequency of 1 MHz, which results in an actual baud rate of 115203.587 bps.

Data transfers using the UART interface can be controlled by the host processor using a memory-mapped register interface and/or the DMA block. The UART_CFG_TX_DMA_ENABLE and UART_CFG_RX_DMA_ENABLE bits in the UART_CFG register determine which request, whether read or write, needs to be completed using the DMA block. Both RX/TX interrupts and RX/TX DMA requests can be generated concurrently; if DMA requests are used, the data transfer requested is fulfilled by the DMA before the interrupt service routine is executed. User applications that do not use DMA or interrupts can poll the UART_STATUS_TX_REQ and UART_STATUS_RX_REQ bits in the UART_STATUS register to control data transfers. When UART_STATUS_TX_REQ is set to 1, the application can write to the UART_TX_DATA register to send data over the UART interface. When a new byte has been received over the UART interface and written to the UART_RX_DATA register, UART_STATUS_RX_REQ is set to 1. To avoid loss of data, the application must read the received byte using the processor or DMA before the next byte is received; otherwise an overrun occurs, and the UART_STATUS_OVERRUN flag in the UART_STATUS register is set to 1.

11.6.1 UART Interrupts

The UART interface uses three associated interrupts that separately control transmission of UART data, reception of UART data, and handling of UART transmission errors. See Section 3.3.2 “Nested Vector Interrupt Controller (NVIC)” on page 58 for information regarding interrupt configuration and handling interrupts for the Arm Cortex-M33 processor.

If a user application is transmitting data from the UART interface, the UART_TX interrupt indicates that data transmission can begin, and that the application can now load the first byte into the UART_TX_DATA register if this is the first interrupt, or load the next byte for subsequent interrupts. The next TX interrupt is generated as soon as the TX transmission has begun. To enable the UART_TX interrupt, set the UART_CFG_TX_INT_ENABLE bit in the UART_CFG register to 1.

If a user application is receiving data from the UART interface, the UART_RX interrupt indicates that a data byte has been successfully received and has been written to the UART_RX_DATA register. To enable the UART_RX interrupt, set the UART_CFG_RX_INT_ENABLE bit in the UART_CFG register to 1.

When using the DMA or the register interface to control data transfers over a UART interface, the UART_ERROR interrupt indicates that an overrun has occurred when data received from the UART interface has not been read by the DMA or processor before being overwritten. Detection of an overrun condition is enabled using the UART_CTRL_OVERRUN_INT_ENABLE bit in the UART_CTRL register. When this occurs, the UART_STATUS_OVERRUN_STATUS sticky bit from the UART_STATUS register is set. To clear it, write 1 to the UART_STATUS_OVERRUN_CLEAR bit.

NOTE: Once a value of 1 is written to the UART_STATUS_OVERRUN_CLEAR bit, it is automatically reset in the next clock cycle. Consequently, this bit is always read back as 0.

For registers, see Section 11.7.6 “UART Registers” on page 639.

11.7 Communication Interfaces Registers

For the user’s convenience, the registers for the communication interfaces are grouped in separate sections according to interface.

11.7.1 I2C Registers

Register Name	Register Description	Address
I2C0_CFG	Configuration Register	0x40000D00
I2C0_CTRL	Control Register	0x40000D04
I2C0_ADDR_START	Master Address and Start Register	0x40000D08
I2C0_STATUS	Status Register	0x40000D0C
I2C0_TX_DATA	Transmit Data Register	0x40000D10
I2C0_RX_DATA	Receive Data Register	0x40000D14
I2C0_RX_DATA_MIRROR	Receive Data Mirror Register	0x40000D18
I2C0_ID_NUM	I2C ID number	0x40000DFC
I2C1_CFG	Configuration Register	0x40000E00
I2C1_CTRL	Control Register	0x40000E04
I2C1_ADDR_START	Master Address and Start Register	0x40000E08
I2C1_STATUS	Status Register	0x40000E0C
I2C1_TX_DATA	Transmit Data Register	0x40000E10
I2C1_RX_DATA	Receive Data Register	0x40000E14
I2C1_RX_DATA_MIRROR	Receive Data Mirror Register	0x40000E18
I2C1_ID_NUM	I2C ID number	0x40000EFC

11.7.1.1 I2C_CFG

Bit Field	Read/Write	Field Name	Description
30	RW	REPEATED_START_INT_ENABLE	Configure whether repeated start interrupts will be generated by the I2C interface for active transactions in slave mode
29	RW	CONNECT_IN_STANDBY	Control if the I2C lines are connected when running on the standby clock
28	RW	TX_DMA_ENABLE	Enable/disable the TX DMA request
27	RW	RX_DMA_ENABLE	Enable/disable the RX DMA request
26	RW	TX_INT_ENABLE	Enable/disable the TX interrupt
25	RW	RX_INT_ENABLE	Enable/disable the RX interrupt

Bit Field	Read/Write	Field Name	Description
24	RW	BUS_ERROR_INT_ENABLE	Enable/disable the bus error interrupt
23	RW	OVERRUN_INT_ENABLE	Enable/disable the overrun interrupt
22	RW	STOP_INT_ENABLE	Configure whether stop interrupts will be generated by the I2C interface
21	RW	AUTO_ACK_ENABLE	Select whether acknowledgement is automatically generated or not
20:16	RW	SLAVE_PRESCALE	Controls the number of SYSCLK wait cycles in case of clock streching (in slave mode) between the moment the data is put on the SDA line and the SCL line is released.
15:8	RW	MASTER_PRESCALE	Prescaler used to divide SYSCLK to the correct SCL frequency when operating in master mode. SCL is prescaled by (PRESCALE + 1) * 3.
7:1	RW	SLAVE_ADDRESS	Set the I2C slave address for this device
0	RW	SLAVE	Select whether the I2C interface is enabled for slave mode or not

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
30	REPEATED_START_INT_ENABLE	I2C_REPEATED_START_INT_DISABLE	Repeated start interrupts are not generated	0x0*
		I2C_REPEATED_START_INT_ENABLE	A repeated start interrupt is generated when a repeated start condition occurs during an active transaction in slave mode	0x1
29	CONNECT_IN_STANDBY	I2C_DISCONNECT_IN_STANDBY	Disconnect the I2C lines when running on standby clock	0x0*
		I2C_CONNECT_IN_STANDBY	Keep the I2C lines connected when running on standby clock	0x1
28	TX_DMA_ENABLE	I2C_TX_DMA_DISABLE	No TX DMA request is generated	0x0*
		I2C_TX_DMA_ENABLE	A TX DMA request is generated when new data is requested by the I2C interface	0x1
27	RX_DMA_ENABLE	I2C_RX_DMA_DISABLE	No RX DMA request is generated	0x0*
		I2C_RX_DMA_ENABLE	An RX DMA request is generated when new data is received by the I2C interface	0x1
26	TX_INT_ENABLE	I2C_TX_INT_DISABLE	No TX interrupt is raised	0x0*
		I2C_TX_INT_ENABLE	A TX interrupt is raised when new data is requested by the I2C interface	0x1
25	RX_INT_ENABLE	I2C_RX_INT_DISABLE	No RX interrupt is raised	0x0*
		I2C_RX_INT_ENABLE	An RX interrupt is raised when new data is	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
			received by the I2C interface	
24	BUS_ERROR_INT_ ENABLE	I2C_BUS_ERROR_INT_DISABLE	No bus error interrupt is raised when an overrun is detected	0x0*
		I2C_BUS_ERROR_INT_ENABLE	A bus error interrupt is raised when an overrun occurs on the I2C interface	0x1
23	OVERRUN_INT_ ENABLE	I2C_OVERRUN_INT_DISABLE	No overrun interrupt is raised when an overrun is detected	0x0*
		I2C_OVERRUN_INT_ENABLE	An overrun interrupt is raised when an overrun occurs on the I2C interface	0x1
22	STOP_INT_ENABLE	I2C_STOP_INT_DISABLE	Stop interrupts are not generated	0x0*
		I2C_STOP_INT_ENABLE	A stop interrupt is generated when a stop condition occurs for an active transaction	0x1
21	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
20:16	SLAVE_PRESCALE	I2C_SLAVE_PRESCALE_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_PRESCALE_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_PRESCALE_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_PRESCALE_4	Slave Standard-Mode: at least 250 ns +10% data set-up time with SYSCLK = 12 MHz	0x3

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		I2C_SLAVE_PRESCALE_5	Slave Standard-Mode: at least 250 ns +10% data set-up time with SYSCLK = 16 MHz	0x4
		I2C_SLAVE_PRESCALE_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_PRESCALE_7	Slave Standard-Mode: at least 250 ns +10% data set-up time with SYSCLK = 24 MHz	0x6
		I2C_SLAVE_PRESCALE_14	Slave Standard-Mode: at least 250 ns +10% data set-up time with SYSCLK = 48 MHz	0xD
		I2C_SLAVE_PRESCALE_32	Maximum number of clock stretching cycles between SDA output and SCL release	0x1F
15:8	MASTER_PRESCALE	I2C_MASTER_PRESCALE_3	Master mode: prescale SCL from SYSCLK by 3	0x0*
		I2C_MASTER_PRESCALE_6	Master mode: prescale SCL from SYSCLK by 6	0x1
		I2C_MASTER_PRESCALE_9	Master mode: prescale SCL from SYSCLK by 9	0x2
		I2C_MASTER_PRESCALE_12	Master mode: prescale SCL from SYSCLK by 12	0x3
		I2C_MASTER_PRESCALE_15	Master mode: prescale SCL from SYSCLK by 15	0x4
		I2C_MASTER_PRESCALE_18	Master mode: prescale SCL from SYSCLK by 18	0x5
		I2C_MASTER_PRESCALE_21	Master mode: prescale SCL from SYSCLK by 21	0x6
		I2C_MASTER_PRESCALE_24	Master mode: prescale SCL from SYSCLK by 24	0x7
		I2C_MASTER_PRESCALE_27	Master mode: prescale SCL from SYSCLK by 27	0x8
		I2C_MASTER_PRESCALE_30	Master mode: prescale SCL from SYSCLK by 30	0x9
		I2C_MASTER_PRESCALE_33	Master mode: prescale SCL from SYSCLK	0xA

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
			by 33	
		I2C_MASTER_PRESCALE_36	Master mode: prescale SCL from SYSCLK by 36	0xB
		I2C_MASTER_PRESCALE_39	Master mode: prescale SCL from SYSCLK by 39	0xC
		I2C_MASTER_PRESCALE_42	Master mode: prescale SCL from SYSCLK by 42	0xD
		I2C_MASTER_PRESCALE_45	Master mode: prescale SCL from SYSCLK by 45	0xE
		I2C_MASTER_PRESCALE_48	Master mode: prescale SCL from SYSCLK by 48	0xF
		I2C_MASTER_PRESCALE_51	Master mode: prescale SCL from SYSCLK by 51	0x10
		I2C_MASTER_PRESCALE_54	Master mode: prescale SCL from SYSCLK by 54	0x11
		I2C_MASTER_PRESCALE_57	Master mode: prescale SCL from SYSCLK by 57	0x12
		I2C_MASTER_PRESCALE_60	Master mode: prescale SCL from SYSCLK by 60	0x13
		I2C_MASTER_PRESCALE_120	Master mode: prescale SCL from SYSCLK by 120	0x27
		I2C_MASTER_PRESCALE_768	Master mode: prescale SCL from SYSCLK by 768	0xFF
0	SLAVE	I2C_SLAVE_DISABLE	Disable I2C interface slave mode operation	0x0*
		I2C_SLAVE_ENABLE	Enable I2C interface slave mode operation	0x1

11.7.1.2 I2C_CTRL

Bit Field	Read/Write	Field Name	Description
9	R	LAST_DATA_STATUS	I2C last data status
8	R	ENABLE_STATUS	I2C enable status
6	W	LAST_DATA	Indicate that the current data is the last byte of a data transfer
5	W	STOP	Issue a stop condition on the I2C interface bus
4	W	NACK	Issue a not acknowledge on the I2C interface bus

Bit Field	Read/Write	Field Name	Description
3	W	ACK	Issue an acknowledge on the I2C interface bus
2	W	RESET	Reset the I2C interface
1	W	DISABLE	Disable the I2C interface
0	W	ENABLE	Enable the I2C interface

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
9	LAST_DATA_STATUS	I2C_LAST_DATA_DISABLED	I2C last data is disabled	0x0*
		I2C_LAST_DATA_ENABLED	I2C last data is enabled	0x1
8	ENABLE_STATUS	I2C_STATUS_DISABLED	I2C interface is disabled	0x0*
		I2C_STATUS_ENABLED	I2C interface is enabled	0x1
6	LAST_DATA	I2C_LAST_DATA	Indicate that the current data is the last byte of a data transfer	0x1
5	STOP	I2C_STOP	Issue a stop condition on the I2C interface bus	0x1
4	NACK	I2C_NACK	Issue a not acknowledge on the I2C interface bus	0x1
3	ACK	I2C_ACK	Issue an acknowledge on the I2C interface bus	0x1
		I2C_RESUME	Resume I2C interface after receiving address ACK/NACK in master read auto ack disable mode	0x1
2	RESET	I2C_RESET	Reset the I2C interface	0x1
1	DISABLE	I2C_DISABLE	Disable the I2C interface	0x1
0	ENABLE	I2C_ENABLE	Enable the I2C interface	0x1

11.7.1.3 I2C_ADDR_START

Bit Field	Read/Write	Field Name	Description
7:1	RW	ADDRESS	I2C address to use for the transaction
0	RW	READ_WRITE	Select whether a read or a write transaction is started

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
0	READ_WRITE	I2C_START_WRITE	Start an I2C write transaction	0x0*
		I2C_START_READ	Start an I2C read transaction	0x1

11.7.1.4 I2C_STATUS

Bit Field	Read/Write	Field Name	Description
26	R	STOP_OR_REPEATED_START_DETECTED	Indicate if STOP_DETECTED or REPEATED_START_DETECTED bit is set
25	R	REPEATED_START_DETECTED	Indicate if an I2C repeated start has been detected during an active transaction in slave mode
22	R	BUS_ERROR	Bus error status bit
21	R	BUSY	Indicate that the reception or transmission of the data is ongoing
20	R	START_PENDING	Master frame start pending status bit
19	R	MASTER_MODE	Master mode status bit
18	R	STOP_DETECTED	Indicate if an I2C stop bit has been detected
17	R	DATA_EVENT	Indicate that I2C interface either needs data to transmit or has received data
16	R	TX_REQ	Indicate that a TX data can be written
15	R	RX_REQ	Indicate that a RX data can be read
14	R	CLK_STRETCH	Clock stretching flag
13	R	LINE_FREE	Line free flag
12	R	ADDR_DATA	Address / Data byte
11	R	READ_WRITE	Read / Write frame
10	R	GEN_CALL	General call flag
9	R	ACK	Acknowledge status
8	R	OVERRUN	Indicate that an overrun has occurred when receiving data
4	W	TX_REQ_SET	Set TX_REQ status flag
3	W	REPEATED_START_DETECTED_CLEAR	Clear REPEATED_START_DETECTED status flag
2	W	STOP_DETECTED_CLEAR	Clear STOP_DETECTED status flag
1	W	BUS_ERROR_CLEAR	Clear BUS_ERROR status flag
0	W	OVERRUN_CLEAR	Clear OVERRUN status flag

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
26	STOP_OR_REPEATED_START_DETECTED	I2C_STOP_NOR_REPEATED_START_DETECTED	Neither STOP_DETECTED nor REPEATED_START_DETECTED bits are set	0x0*
		I2C_STOP_OR_REPEATED_START_DETECTED	STOP_DETECTED or REPEATED_START_DETECTED bit is set	0x1
25	REPEATED_START_DETECTED	I2C_NO_REPEATED_START_DETECTED	No repeated start condition has been detected on the I2C bus during an active transaction in slave mode	0x0*
		I2C_REPEATED_START_DETECTED	A repeated start condition has been detected on the I2C bus during an active transaction in slave mode	0x1
22	BUS_ERROR	I2C_NO_BUS_ERROR	No I2C bus error has occurred	0x0*
		I2C_BUS_ERROR	An I2C bus error has occurred	0x1
21	BUSY	I2C_IDLE	I2C interface is idle	0x0*
		I2C_BUSY	I2C interface is busy	0x1
20	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
19	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
18	STOP_DETECTED	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
17	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
16	TX_REQ	I2C_TX_NO_REQ	I2C TX data has already been written	0x0
		I2C_TX_REQ	I2C TX data can be written	0x1*
15	RX_REQ	I2C_RX_NO_REQ	No new I2C RX data available	0x0*
		I2C_RX_REQ	New I2C RX data available	0x1
14	CLK_STRETCH	I2C_CLK_NOT_STRETCHED	I2C interface is currently not clock streching	0x0*
		I2C_CLK_STRETCHED	I2C interface is currently clock streching	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
13	LINE_FREE	I2C_BUS_BUSY	I2C transaction ongoing	0x0
		I2C_BUS_FREE	I2C bus is free	0x1*
12	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
11	READ_WRITE	I2C_IS_WRITE	The current I2C transfer is a write	0x0*
		I2C_IS_READ	The current I2C transfer is a read	0x1
10	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
9	ACK	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
8	OVERRUN	I2C_OVERRUN_FALSE	No I2C input overrun detected	0x0*
		I2C_OVERRUN_TRUE	I2C input overrun detected	0x1
4	TX_REQ_SET	I2C_TX_REQ_SET	Set the TX_REQ status flag	0x1
3	REPEATED_START_ DETECTED_CLEAR	I2C_REPEATED_START_ DETECTED_CLEAR	Clear the REPEATED_START_ DETECTED status flag	0x1
2	STOP_DETECTED_ CLEAR	I2C_STOP_DETECTED_CLEAR	Clear the STOP_DETECTED status flag	0x1
1	BUS_ERROR_CLEAR	I2C_BUS_ERROR_CLEAR	Clear the BUS_ERROR status flag	0x1
0	OVERRUN_CLEAR	I2C_OVERRUN_CLEAR	Clear the OVERRUN status flag	0x1

11.7.1.5 I2C_TX_DATA

Bit Field	Read/Write	Field Name	Description
7:0	RW	TX_DATA	Single byte buffer for data transmitted over the I2C interface

11.7.1.6 I2C_RX_DATA

Bit Field	Read/Write	Field Name	Description
7:0	R	RX_DATA	Single byte buffer for data received over the I2C interface

11.7.1.7 I2C_RX_DATA_MIRROR

Bit Field	Read/Write	Field Name	Description

11.7.1.8 I2C_ID_NUM

Bit Field	Read/Write	Field Name	Description
22	R	I2C_WATCHDOG	Implementation of the watchdog counter
21	R	I2C_DEBUG	Implementation of the debug interface
20	R	I2C_DMA	Implementation of the DMA interface
19:16	R	I2C_NUMBER	I2C instance number
15:8	R	I2C_MAJOR_REVISION	I2C Major Revision number
7:0	R	I2C_MINOR_REVISION	I2C Minor Revision number

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
22	I2C_WATCHDOG	I2C_WATCHDOG_DISABLED	Watchdog counter is not implemented	0x0*
		I2C_WATCHDOG_ENABLED	Watchdog counter is implemented	0x1
21	I2C_DEBUG	I2C_DEBUG_DISABLED	Debug interface is not implemented	0x0*
		I2C_DEBUG_ENABLED	Debug interface is implemented	0x1
20	I2C_DMA	I2C_DMA_DISABLED	DMA interface is not implemented	0x0*
		I2C_DMA_ENABLED	DMA interface is implemented	0x1
15:8	I2C_MAJOR_REVISION	I2C_MAJOR_REVISION	I2C revision 1.0	0x1*
7:0	I2C_MINOR_REVISION	I2C_MINOR_REVISION	I2C revision 1.0	0x0*

11.7.2 LIN Registers

Register Name	Register Description	Address
LIN0_CFG	LIN Configuration Register	0x40002000
LIN0_CTRL	LIN Control Register	0x40002004
LIN0_ERROR	LIN Error Register	0x40002008
LIN0_PID	LIN Protected Identifier Register	0x4000200C
LIN0_DLB	LIN number of bytes to transmit	0x40002010
LIN0_DLBR	LIN number of bytes to receive	0x40002014
LIN0_DATA	LIN Data Byte Register (Byte access)	0x40002018-0x40002034
LIN0_DATA_WORD0	LIN Data Byte Register (Word access)	0x40002038
LIN0_DATA_WORD1	LIN Data Byte Register (Word access)	0x4000203C
LIN0_CHECKSUM	LIN Checksum Register	0x40002040
LIN0_SYNCH	LIN	0x400020F4
LIN0_ID_NUM	LIN ID Number	0x400020FC

11.7.2.1 LIN_CFG

Bit Field	Read/Write	Field Name	Description
8	RW	CHECKSUM_MODE	Select checksum mode
4	RW	INIT	Start C617 initialization
0	R	STANDBY	C617 state

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
8	CHECKSUM_MODE	ID_IN_CHECKSUM	Enhance check sum, include ID	0x0*
		ONLY_DATA_IN_CHECKSUM	Classic check sum, only data byte	0x1
4	INIT	NOT_INIT_C617		0x0*
		INIT_C617		0x1
0	STANDBY	STANDBY_STATE		0x0*
		NORMAL_MODE		0x1

11.7.2.2 LIN_CTRL

Bit Field	Read/Write	Field Name	Description
8	W	RESET	Reset the LIN module to default configuration
7	R	RHC	This bit is set, when frame header is successfully received (synchronization passed and frame identifier received). Bit is cleared during transition to RX_ID state. Setting of bit to high by controller has higher priority than bit clear by CM33 (frame identifier received). Bit is cleared on MCU read. Bit is cleared during transition to RX_ID state. Setting of bit to high by controller has higher priority than bit clear by CM33
6	R	RXF	This bit is set when LIN controller successfully receives all data bytes and checksum is correct. Bit is cleared during transition to RX_ID state. Setting of bit to high by controller has higher priority than bit clear by CM33.
5	R	TXF	This bit is set when controller starts to transmit. Bit is cleared when all bytes are transmitted or bit error occurs or break/sync is detected or during transition to RX_ID state.
4	R	ENABLE_STATUS	LIN enable status
1	W	DISABLE	Disable the LIN
0	W	ENABLE	Enable the LIN

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
8	RESET	LIN_RESET	Reset the LIN module	0x1
7	RHC	NO_LIN_FH_RX	No frame header received	0x0*
		LIN_FH_RX	Frame header received	0x1
6	RXF	LIN_RX_ONGOING	Recevied process on-going	0x0*
		LIN_RX_DONE	Received process done	0x1
5	TXF	LIN_TX_DONE	Transmit process done	0x0*
		LIN_TX_ONGOING	Transmit process on-going	0x1
4	ENABLE_STATUS	LIN_STATUS_DISABLE	LIN inactives	0x0*
		LIN_STATUS_ENABLE	LIN actives	0x1
1	DISABLE	LIN_DISABLE	LIN disables	0x1
0	ENABLE	LIN_ENABLE	LIN enables	0x1

11.7.2.3 LIN_ERROR

Bit Field	Read/Write	Field Name	Description
11	W	CLR_CE	Clear Checksum error flag
10	W	CLR_PE	Clear Parity error flag
9	W	CLR_BE	Clear Bit error flag
8	W	CLR_FE	Clear Frame error flag
7	R	CE	Checksum error: this bit is set to high when all data bytes are received but value of calculated checksum does not equal the value of 255.
6	R	PE	Parity error: this bit is set when parity of received identifier is different from calculated parity.
5	R	BE	Bit error: this bit is set when transmitted data is different from read back data. This bit is cleared on MCU read.
4	R	FE	Framing error: this bit is set when received stop bit is 0 in sync field, identifier or data byte.
3	R	CE_WRC	Checksum error: this bit is set to high when all data bytes are received but value of calculated checksum does not equal the value of 255. This bit is cleared on MCU read.
2	R	PE_WRC	Parity error: this bit is set when parity of received identifier is different from calculated parity. This bit is cleared on MCU read.
1	R	BE_WRC	Bit error: this bit is set when transmitted data is different from read back data. This bit is cleared on MCU read.
0	R	FE_WRC	Framing error: this bit is set when received stop bit is 0 in sync field, identifier or data byte. This bit is cleared on MCU read.

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
11	CLR_CE	CLR_CE	Clear checksum error flag	0x1
10	CLR_PE	CLR_PE	Clear parity error flag	0x1
9	CLR_BE	CLR_BE	Clear bit error flag	0x1
8	CLR_FE	CLR_FE	Clear frame error flag	0x1
7	CE	NO_CHECKSUM_ERROR	No checksum error	0x0*
		CHECKSUM_ERROR	Checksum error occurs	0x1
6	PE	NO_PARITY_ERROR	No parity error	0x0*
		PARITY_ERROR	Parity error occurs	0x1
5	BE	NO_BIT_ERROR	No bit error	0x0*
		BIT_ERROR	Bit error occurs	0x1
4	FE	NO_FRAME_ERROR	No frame error	0x0*
		FRAME_ERROR	Frame error occurs	0x1
3	CE_WRC	NO_CHECKSUM_ERROR_WRC	No checksum error	0x0*
		CHECKSUM_ERROR_WRC	Checksum error occurs	0x1
2	PE_WRC	NO_PARITY_ERROR_WRC	No parity error	0x0*
		PARITY_ERROR_WRC	Parity error occurs	0x1
1	BE_WRC	NO_BIT_ERROR_WRC	No bit error	0x0*
		BIT_ERROR_WRC	Bit error occurs	0x1
0	FE_WRC	NO_FRAME_ERROR_WRC	No frame error	0x0*
		FRAME_ERROR_WRC	Frame error occurs	0x1

11.7.2.4 LIN_PID

Bit Field	Read/Write	Field Name	Description
7:0	R	PID	LIN Protected Identifier register

11.7.2.5 LIN_DLB

Bit Field	Read/Write	Field Name	Description
4	RW	DELAY	Complete the PID stop bit before sending the first byte
2:0	RW	DLBT	Number of data bytes to transmit.

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
4	DELAY	NO_DELAY	No delay added	0x0*
		ADD_DELAY	Finished the Stop bit before sending the byte.	0x1
2:0	DLBT	TX_1_BYTE	1 byte will be transmitted	0x0*
		TX_2_BYTES	2 bytes will be transmitted	0x1
		TX_3_BYTES	3 bytes will be transmitted	0x2
		TX_4_BYTES	4 bytes will be transmitted	0x3
		TX_5_BYTES	5 bytes will be transmitted	0x4
		TX_6_BYTES	6 bytes will be transmitted	0x5
		TX_7_BYTES	7 bytes will be transmitted	0x6
		TX_8_BYTES	8 bytes will be transmitted	0x7

11.7.2.6 LIN_DLBR

Bit Field	Read/Write	Field Name	Description
2:0	RW	DLBR	Number of data bytes to receive.

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
2:0	DLBR	RX_1_BYTE	1 byte will be received	0x0*
		RX_2_BYTES	2 bytes will be received	0x1
		RX_3_BYTES	3 bytes will be received	0x2
		RX_4_BYTES	4 bytes will be received	0x3
		RX_5_BYTES	5 bytes will be received	0x4
		RX_6_BYTES	6 bytes will be received	0x5
		RX_7_BYTES	7 bytes will be received	0x6
		RX_8_BYTES	8 bytes will be received	0x7

11.7.2.7 LIN_DATA

Bit Field	Read/Write	Field Name	Description
7:0	RW	DATA	LIN Data byte received or to transmit

11.7.2.8 LIN_DATA_WORD0

Bit Field	Read/Write	Field Name	Description
31:24	RW	DATA3	LIN Data[3] byte received or to transmit
23:16	RW	DATA2	LIN Data[2] byte received or to transmit
15:8	RW	DATA1	LIN Data[1] byte received or to transmit
7:0	RW	DATA0	LIN Data[0] byte received or to transmit

11.7.2.9 LIN_DATA_WORD1

Bit Field	Read/Write	Field Name	Description
31:24	RW	DATA7	LIN Data[7] byte received or to transmit
23:16	RW	DATA6	LIN Data[6] byte received or to transmit
15:8	RW	DATA5	LIN Data[5] byte received or to transmit
7:0	RW	DATA4	LIN Data[4] byte received or to transmit

11.7.2.10 LIN_CHECKSUM

Bit Field	Read/Write	Field Name	Description
7:0	R	CHECKSUM	Checksum

11.7.2.11 LIN_SYNCH

Bit Field	Read/Write	Field Name	Description
13:0	R	TSYNC	Duration of TSYNC

11.7.2.12 LIN_ID_NUM

Bit Field	Read/Write	Field Name	Description
15:8	R	LIN_MAJOR_REVISION	LIN Major Revision number
7:0	R	LIN_MINOR_REVISION	LIN Minor Revision number

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
15:8	LIN_MAJOR_REVISION	LIN_MAJOR_REVISION	LIN revision 1.0	0x1*
7:0	LIN_MINOR_REVISION	LIN_MINOR_REVISION	LIN revision 1.0	0x0*

11.7.3 PCM Registers

Register Name	Register Description	Address
PCM0_CFG	PCM Configuration Register	0x40001000
PCM0_CTRL	PCM Control Register	0x40001004
PCM0_STATUS	PCM Status Register	0x40001008
PCM0_TX_DATA0	PCM Transmit Data 0 Register	0x40001010
PCM0_TX_DATA1	PCM Transmit Data 1 Register	0x40001014
PCM0_RX_DATA0	PCM Receive Data 0 Register	0x40001018
PCM0_RX_DATA1	PCM Receive Data 1 Register	0x4000101C
PCM0_ID_NUM	PCM ID number	0x400010FC

11.7.3.1 PCM_CTRL

Bit Field	Read/Write	Field Name	Description
8	R	ENABLE_STATUS	PCM enable status
2	W	RESET	Reset the PCM interface
1	W	DISABLE	Disable the PCM interface
0	W	ENABLE	Enable the PCM interface

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
8	ENABLE_STATUS	PCM_STATUS_DISABLED	PCM interface is disabled	0x0*
		PCM_STATUS_ENABLED	PCM interface is enabled	0x1
2	RESET	PCM_RESET	Reset the PCM interface	0x1
1	DISABLE	PCM_DISABLE	Disable the PCM interface	0x1
0	ENABLE	PCM_ENABLE	Enable the PCM interface	0x1

11.7.3.2 PCM_CFG

Bit Field	Read/Write	Field Name	Description
25	RW	TX_ACK_SEL	Select which TX word acknowleges the TX request
24	RW	RX_ACK_SEL	Select which RX word acknowleges the RX request
23	RW	TX_DMA_ENABLE	Enable/disable the TX DMA request
22	RW	RX_DMA_ENABLE	Enable/disable the RX DMA request
21	RW	TX_IOC_ENABLE	Enable/disable the TX IOC request

Bit Field	Read/Write	Field Name	Description
20	RW	RX_IOC_ENABLE	Enable/disable the RX IOC request
19	RW	RX_TX_INT_ENABLE	Enable/disable the RX_TX interrupt
17	RW	OVERRUN_INT_EN	Enable/disable the overrun interrupt
16	RW	UNDERRUN_INT_EN	Enable/disable the underrun interrupt
15	RW	CLK_POLARITY	Select the PCM clock polarity
14	RW	TX_DATA_ALIGN	Select the TX data alignment
13	RW	RX_DATA_ALIGN	Select the RX data alignment
12:8	RW	WORD_SIZE	Select the number of bits per PCM word
7	RW	BIT_ORDER	Select whether the data is transmitted starting with the MSB or LSB
6	RW	FRAME_ALIGN	Align the PCM frame signal to the first/last bit
5	RW	FRAME_WIDTH	Use a long/short PCM frame signal
4:2	RW	FRAME_LENGTH	Select the number of words per PCM frame
1	RW	SUBFRAME	Enable the frame duration for each word
0	RW	SLAVE	Use the PCM interface as a master/slave

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
25	TX_ACK_SEL	PCM_TX_ACK_DATA0	TX data0 acknowledges the TX request	0x0*
		PCM_TX_ACK_DATA1	TX data1 acknowledges the TX request	0x1
24	RX_ACK_SEL	PCM_RX_ACK_DATA0	RX data0 acknowledges the RX request	0x0*
		PCM_RX_ACK_DATA1	RX data1 acknowledges the RX request	0x1
23	TX_DMA_ENABLE	PCM_TX_DMA_DISABLE	No TX DMA request is generated	0x0*
		PCM_TX_DMA_ENABLE	A TX DMA request is generated when new data is requested by the PCM interface	0x1
22	RX_DMA_ENABLE	PCM_RX_DMA_DISABLE	No RX DMA request is generated	0x0*
		PCM_RX_DMA_ENABLE	An RX DMA request is generated when new data is received by the PCM interface	0x1
21	TX_IOC_ENABLE	PCM_TX_IOC_DISABLE	No TX IOC request is generated	0x0*
		PCM_TX_IOC_ENABLE	A TX IOC request is generated when new data is requested by the PCM interface	0x1
20	RX_IOC_ENABLE	PCM_RX_IOC_DISABLE	No RX IOC request is generated	0x0*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		PCM_RX_IOC_ENABLE	An RX IOC request is generated when new data is received by the PCM interface	0x1
19	RX_TX_INT_ENABLE	PCM_RX_TX_INT_DISABLE	No RX_TX interrupt is raised	0x0*
		PCM_RX_TX_INT_ENABLE	An RX_TX interrupt is raised when new data is received or requested by the PCM interface	0x1
17	OVERRUN_INT_EN	PCM_OVERRUN_INT_DISABLE	No ERROR interrupt is raised when an overrun is detected	0x0*
		PCM_OVERRUN_INT_ENABLE	An ERROR interrupt is raised when an overrun occurs on the PCM interface	0x1
16	UNDERRUN_INT_EN	PCM_UNDERRUN_INT_DISABLE	No ERROR interrupt is raised when an underrun is detected	0x0*
		PCM_UNDERRUN_INT_ENABLE	An ERROR interrupt is raised when an underrun occurs on the PCM interface	0x1
15	CLK_POLARITY	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
14	TX_DATA_ALIGN	PCM_TX_DATA_ALIGN_MSB	TX data is aligned on the MSB of the 32-bit TX buffer	0x0*
		PCM_TX_DATA_ALIGN_LSB	TX data is aligned on the LSB of the 32-bit TX buffer	0x1
13	RX_DATA_ALIGN	PCM_RX_DATA_ALIGN_MSB	RX data is aligned on the MSB of the 32-bit RX buffer	0x0*
		PCM_RX_DATA_ALIGN_LSB	RX data is aligned on the LSB of the 32-bit RX buffer	0x1
12:8	WORD_SIZE	PCM_WORD_SIZE_8	Use 8-bit words	0x0
		PCM_WORD_SIZE_9	Use 9-bit words	0x1
		PCM_WORD_SIZE_10	Use 10-bit words	0x2
		PCM_WORD_SIZE_11	Use 11-bit words	0x3
		PCM_WORD_SIZE_12	Use 12-bit words	0x4
		PCM_WORD_SIZE_13	Use 13-bit words	0x5
		PCM_WORD_SIZE_14	Use 14-bit words	0x6
		PCM_WORD_SIZE_15	Use 15-bit words	0x7

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		PCM_WORD_SIZE_16	Use 16-bit words	0x8*
		PCM_WORD_SIZE_17	Use 17-bit words	0x9
		PCM_WORD_SIZE_18	Use 18-bit words	0xA
		PCM_WORD_SIZE_19	Use 19-bit words	0xB
		PCM_WORD_SIZE_20	Use 20-bit words	0xC
		PCM_WORD_SIZE_21	Use 21-bit words	0xD
		PCM_WORD_SIZE_22	Use 22-bit words	0xE
		PCM_WORD_SIZE_23	Use 23-bit words	0xF
		PCM_WORD_SIZE_24	Use 24-bit words	0x10
		PCM_WORD_SIZE_25	Use 25-bit words	0x11
		PCM_WORD_SIZE_26	Use 26-bit words	0x12
		PCM_WORD_SIZE_27	Use 27-bit words	0x13
		PCM_WORD_SIZE_28	Use 28-bit words	0x14
		PCM_WORD_SIZE_29	Use 29-bit words	0x15
		PCM_WORD_SIZE_30	Use 30-bit words	0x16
		PCM_WORD_SIZE_31	Use 31-bit words	0x17
		PCM_WORD_SIZE_32	Use 32-bit words	0x18
7	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
6	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
5	FRAME_WIDTH	PCM_FRAME_WIDTH_SHORT	The frame is high for one PCM clock period.	0x0*
		PCM_FRAME_WIDTH_LONG	The frame is high for half of the frame length.	0x1
4:2	FRAME_LENGTH	PCM_MULTIWORD_2	A frame contains 2 words.	0x0*
		PCM_MULTIWORD_4	A frame contains 4 words.	0x1
		PCM_MULTIWORD_6	A frame contains 6 words.	0x2
		PCM_MULTIWORD_8	A frame contains 8 words.	0x3
		PCM_MULTIWORD_10	A frame contains 10 words.	0x4

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		PCM_MULTIWORD_12	A frame contains 12 words.	0x5
		PCM_MULTIWORD_14	A frame contains 14 words.	0x6
		PCM_MULTIWORD_16	A frame contains 16 words.	0x7
1	SUBFRAME	PCM_SUBFRAME_DISABLE	Generate a frame signal every frame.	0x0*
		PCM_SUBFRAME_ENABLE	Generate a frame signal every word.	0x1
0	SLAVE	PCM_SELECT_MASTER	The PCM interface generates the PCM_ FRAME.	0x0
		PCM_SELECT_SLAVE	The PCM interface received an external PCM_FRAME.	0x1*

11.7.3.3 PCM_STATUS

Bit Field	Read/Write	Field Name	Description
12	R	BUSY	Indicate that the reception or transmission of the data is ongoing
11	R	TX_REQ	Indicate that TX data can be written
10	R	RX_REQ	Indicate that RX data can be read
9	R	OVERRUN	Indicate that an overrun occurred when receiving data
8	R	UNDERRUN	Indicate that an underrun occurred when transmitting data
1	W	OVERRUN_CLEAR	Clear the overrun status flag
0	W	UNDERRUN_CLEAR	Clear the underrun status flag

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
12	BUSY	PCM_IDLE	PCM idle	0x0*
		PCM_BUSY	PCM busy	0x1
11	TX_REQ	PCM_TX_NO_REQ	PCM TX data has already been written	0x0
		PCM_TX_REQ	PCM TX data can be written	0x1*
10	RX_REQ	PCM_RX_NO_REQ	No new PCM RX data available	0x0*
		PCM_RX_REQ	New PCM RX data available	0x1
9	OVERRUN	PCM_OVERRUN_FALSE	No PCM input overrun detected	0x0*
		PCM_OVERRUN_TRUE	PCM input overrun detected	0x1
8	UNDERRUN	PCM_UNDERRUN_FALSE	No PCM output underrun detected	0x0*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		PCM_UNDERRUN_TRUE	PCM output underrun detected	0x1
1	OVERRUN_CLEAR	PCM_OVERRUN_CLEAR	Clear the PCM overrun bit	0x1
0	UNDERRUN_CLEAR	PCM_UNDERRUN_CLEAR	Clear the PCM underrun bit	0x1

11.7.3.4 PCM_TX_DATA0

Bit Field	Read/Write	Field Name	Description
31:0	RW	TX_DATA0	Data to transmit on channel 0

11.7.3.5 PCM_TX_DATA1

Bit Field	Read/Write	Field Name	Description
31:0	RW	TX_DATA1	Data to transmit on channel 1

11.7.3.6 PCM_RX_DATA0

Bit Field	Read/Write	Field Name	Description
31:0	R	RX_DATA0	Data received on channel 0

11.7.3.7 PCM_RX_DATA1

Bit Field	Read/Write	Field Name	Description
31:0	R	RX_DATA1	Data received on channel 1

11.7.3.8 PCM_ID_NUM

Bit Field	Read/Write	Field Name	Description
19:16	R	PCM_NUMBER	PCM Instance number
15:8	R	PCM_MAJOR_REVISION	PCM Major Revision number
7:0	R	PCM_MINOR_REVISION	PCM Minor Revision number

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
15:8	PCM_MAJOR_ REVISION	PCM_MAJOR_REVISION	PCM revision 1.0	0x1*
7:0	PCM_MINOR_REVISION	PCM_MINOR_REVISION	PCM revision 1.0	0x0*

11.7.4 PWM Registers

Register Name	Register Description	Address
PWM_PERIOD	PWM Period Register	0x40001100-0x40001110
PWM_CTRL	PWM Control Register	0x40001114
PWM_OFFSET	PWM Offset Register	0x40001118-0x40001124
PWM_HIGH	PWM High configuration Register	0x40001128-0x40001138
PWM_ID_NUM	PWM ID number	0x400011FC
ACS_PWM_AO_CFG	ACS PWM Always-On Configuration Register	0x40001B90
ACS_PWM_AO_CTRL	ACS PWM Always-On Control Register	0x40001B94
ACS_PWM_AO_COUNT	ACS PWM Always-On counter	0x40001B98

11.7.4.1 PWM_PERIOD

Bit Field	Read/Write	Field Name	Description
11:0	RW	PERIOD	PWM period

11.7.4.2 PWM_CTRL

Bit Field	Read/Write	Field Name	Description
28:24	W	RESET	PWM[4:0] channel reset
20:16	R	ENABLE_STATUS	PWM enable status
12:8	W	DISABLE	Disable the PWM[4:0] channel
4:0	W	ENABLE	Enable the PWM[4:0] channel

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
28:24	RESET	PWM0_RESET	Disable the PWM0 channel, reset the PWM_CFG0 and PWM_OFFSET0 registers and the related counter.	0x1
		PWM1_RESET	Disable the PWM1 channel, reset the PWM_CFG1 and PWM_OFFSET1 registers and the related counter.	0x2
		PWM2_RESET	Disable the PWM2 channel, reset the PWM_CFG2 and PWM_OFFSET2 registers and the related counter.	0x4
		PWM3_RESET	Disable the PWM3 channel, reset the PWM_CFG3 and PWM_OFFSET3 registers and the related counter.	0x8

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		PWM4_RESET	Disable the PWM4 channel, reset the PWM_CFG4 and PWM_OFFSET4 registers and the related counter.	0x10
20:16	ENABLE_STATUS	PWM0_STATUS_DISABLE	PWM0 channel disable	0x0*
		PWM1_STATUS_DISABLE	PWM1 channel disable	0x0*
		PWM2_STATUS_DISABLE	PWM2 channel disable	0x0*
		PWM3_STATUS_DISABLE	PWM3 channel disable	0x0*
		PWM4_STATUS_DISABLE	PWM4 channel disable	0x0*
		PWM0_STATUS_ENABLE	PWM0 channel enable	0x1
		PWM1_STATUS_ENABLE	PWM1 channel enable	0x2
		PWM2_STATUS_ENABLE	PWM2 channel enable	0x4
		PWM3_STATUS_ENABLE	PWM3 channel enable	0x8
		PWM4_STATUS_ENABLE	PWM4 channel enable	0x10
12:8	DISABLE	PWM0_DISABLE	Disable the PWM0 channel	0x1
		PWM1_DISABLE	Disable the PWM1 channel	0x2
		PWM2_DISABLE	Disable the PWM2 channel	0x4
		PWM3_DISABLE	Disable the PWM3 channel	0x8
		PWM4_DISABLE	Disable the PWM4 channel	0x10
4:0	ENABLE	PWM0_ENABLE	Enable the PWM0 channel	0x1
		PWM1_ENABLE	Enable the PWM1 channel	0x2
		PWM2_ENABLE	Enable the PWM2 channel	0x4
		PWM3_ENABLE	Enable the PWM3 channel	0x8
		PWM4_ENABLE	Enable the PWM4 channel	0x10

11.7.4.3 PWM_OFFSET

Bit Field	Read/Write	Field Name	Description
12	RW	OFFSET_ENABLE	Enable/disable the PWM offset function
11:0	RW	OFFSET	PWM(0) to PWM(i) offset

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
12	OFFSET_ENABLE	PWM_OFFSET_DISABLE	Disable the PWM offset	0x0*
		PWM_OFFSET_ENABLE	Enable the PWM offset	0x1

11.7.4.4 PWM_HIGH

Bit Field	Read/Write	Field Name	Description
19:8	RW	HIGH	PWM high duration
7:0	RW	HIGH_FRACTIONAL	PWM high fractional

11.7.4.5 PWM_ID_NUM

Bit Field	Read/Write	Field Name	Description
18:16	R	PWM_NUMBER	PWM number of channel
15:8	R	PWM_MAJOR_REVISION	PWM Major Revision number
7:0	R	PWM_MINOR_REVISION	PWM Minor Revision number

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
15:8	PWM_MAJOR_ REVISION	PWM_MAJOR_REVISION	PWM revision 1.0	0x1*
7:0	PWM_MINOR_ REVISION	PWM_MINOR_REVISION	PWM revision 1.0	0x0*

11.7.4.6 ACS_PWM_AO_CFG

Bit Field	Read/Write	Field Name	Description
15:8	RW	HIGH	PWM high duty cycle
7:0	RW	PERIOD	PWM period

11.7.4.7 ACS_PWM_AO_CTRL

Bit Field	Read/Write	Field Name	Description
8	R	ENABLE_STATUS	Status of the PWM enable
2	W	RESET	Reset the PWM
1	W	DISABLE	Disable the PWM
0	W	ENABLE	Enable the PWM

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
8	ENABLE_STATUS	PWM_IS_DISABLED	PWM is disabled	0x0*
		PWM_IS_ENABLED	PWM is enabled	0x1
2	RESET	PWM_RESET	Reset PWM	0x1
1	DISABLE	PWM_DISABLE	Disable PWM	0x1
0	ENABLE	PWM_ENABLE	Enable PWM	0x1

11.7.4.8 ACS_PWM_AO_COUNT

Bit Field	Read/Write	Field Name	Description
7:0	R	COUNTER	PWM counter

11.7.5 SPI Registers

Register Name	Register Description	Address
SPI0_CFG	SPI Configuration Register	0x40000B00
SPI0_CTRL	SPI Control Register	0x40000B04
SPI0_STATUS	SPI Status Register	0x40000B08
SPI0_TX_DATA	SPI Transmit Data Register	0x40000B0C
SPI0_RX_DATA	SPI Received Data Register	0x40000B10
SPI0_RX_DATA_NO_START	SPI Received Data No Start Register	0x40000B14
SPI0_RX_DATA_MIRROR	SPI Received Data Mirror Register	0x40000B18
SPI0_ID_NUM	SPI ID number	0x40000BFC
SPI1_CFG	SPI Configuration Register	0x40000C00
SPI1_CTRL	SPI Control Register	0x40000C04
SPI1_STATUS	SPI Status Register	0x40000C08
SPI1_TX_DATA	SPI Transmit Data Register	0x40000C0C
SPI1_RX_DATA	SPI Received Data Register	0x40000C10
SPI1_RX_DATA_NO_START	SPI Received Data No Start Register	0x40000C14
SPI1_RX_DATA_MIRROR	SPI Received Data Mirror Register	0x40000C18
SPI1_ID_NUM	SPI ID number	0x40000CFC

11.7.5.1 SPI_CFG

Bit Field	Read/Write	Field Name	Description
23	RW	TX_DMA_ENABLE	Enable/disable the TX DMA request
22	RW	RX_DMA_ENABLE	Enable/disable the RX DMA request
21	RW	TX_END_INT_ENABLE	Enable/disable the TX interrupt
20	RW	TX_START_INT_ENABLE	Enable/disable the TX interrupt
19	RW	RX_INT_ENABLE	Enable/disable the RX interrupt
18	RW	CS_RISE_INT_ENABLE	Enable/disable the CS rise interrupt (slave mode only)
17	RW	OVERRUN_INT_ENABLE	Enable/disable the overrun interrupt

Bit Field	Read/Write	Field Name	Description
16	RW	UNDERRUN_INT_ENABLE	Enable/disable the underrun interrupt
14:13	RW	MODE	Select the SPI master mode (ignored in slave mode)
12:8	RW	WORD_SIZE	Select the SPI word size (word size = SPI_WORD_SIZE + 1)
7:4	RW	PRESCALE	Prescale the SPI clock for master mode
2	RW	CLK_PHASE	Select the SPI clock phase
1	RW	CLK_POLARITY	Select the SPI clock polarity
0	RW	SLAVE	Use the SPI interface as master or slave

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
23	TX_DMA_ENABLE	SPI_TX_DMA_DISABLE	No TX DMA request is generated	0x0*
		SPI_TX_DMA_ENABLE	A TX DMA request is generated when TX buffer is empty	0x1
22	RX_DMA_ENABLE	SPI_RX_DMA_DISABLE	No RX DMA request is generated	0x0*
		SPI_RX_DMA_ENABLE	An RX DMA request is generated when RX buffer is full	0x1
21	TX_END_INT_ENABLE	SPI_TX_END_INT_DISABLE	No TX interrupt is raised	0x0*
		SPI_TX_END_INT_ENABLE	A TX interrupt is raised when a TX data transmission is finished	0x1
20	TX_START_INT_ ENABLE	SPI_TX_START_INT_DISABLE	No TX interrupt is raised	0x0*
		SPI_TX_START_INT_ENABLE	A TX interrupt is raised when a new TX data send by the SPI interface is started	0x1
19	RX_INT_ENABLE	SPI_RX_INT_DISABLE	No RX interrupt is raised	0x0*
		SPI_RX_INT_ENABLE	An RX interrupt is raised when new data is received by the SPI interface	0x1
18	CS_RISE_INT_ENABLE	SPI_CS_RISE_INT_DISABLE	No interrupt is raised when the CS rises in slave mode	0x0*
		SPI_CS_RISE_INT_ENABLE	A common interrupt is raised when the CS rises in slave mode	0x1
17	OVERRUN_INT_ ENABLE	SPI_OVERRUN_INT_DISABLE	No interrupt is raised when an overrun is detected	0x0*
		SPI_OVERRUN_INT_ENABLE	A common interrupt is raised when an overrun occurs on the SPI interface	0x1
16	UNDERRUN_INT_	SPI_UNDERRUN_INT_DISABLE	No interrupt is raised when an underrun is	0x0*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
	ENABLE		detected	
		SPI_UNDERRUN_INT_ENABLE	A common interrupt is raised when an underrun occurs on the SPI interface	0x1
14:13	MODE	SPI_MODE_SPI	SPI normal mode (separate RX and TX data)	0x0*
		SPI_MODE_DSPI	SPI dual mode (two bideractional data pins)	0x1
		SPI_MODE_QSPI	SPI quad mode (four bidirectional data pins)	0x2
12:8	WORD_SIZE	SPI_WORD_SIZE_1	SPI transfers use 1-bit words	0x0*
		SPI_WORD_SIZE_4	SPI transfers use 4-bit words	0x3
		SPI_WORD_SIZE_8	SPI transfers use 8-bit words	0x7
		SPI_WORD_SIZE_16	SPI transfers use 16-bit words	0xF
		SPI_WORD_SIZE_24	SPI transfers use 24-bit words	0x17
		SPI_WORD_SIZE_32	SPI transfers use 32-bit words	0x1F
7:4	PRESCALE	SPI_PRESCALE_2	Prescale the SPI interface clock by 2	0x0*
		SPI_PRESCALE_4	Prescale the SPI interface clock by 4	0x1
		SPI_PRESCALE_8	Prescale the SPI interface clock by 8	0x2
		SPI_PRESCALE_16	Prescale the SPI interface clock by 16	0x3
		SPI_PRESCALE_32	Prescale the SPI interface clock by 32	0x4
		SPI_PRESCALE_64	Prescale the SPI interface clock by 64	0x5
		SPI_PRESCALE_128	Prescale the SPI interface clock by 128	0x6
		SPI_PRESCALE_256	Prescale the SPI interface clock by 256	0x7
		SPI_PRESCALE_512	Prescale the SPI interface clock by 512	0x8
		SPI_PRESCALE_1024	Prescale the SPI interface clock by 1024	0x9
2	CLK_PHASE	SPI_CLK_PHASE_RISING	In both master and slave modes SERO changes on the falling edge of the SPI clock. The SERI is sampled at the rising edge of the SPI clock	0x0*
		SPI_CLK_PHASE_FALLING	In both master and slave modes SERO changes on the rising edge of the SPI clock. The SERI is sampled at the falling edge of the SPI clock	0x1
1	CLK_POLARITY	SPI_CLK_POLARITY_NORMAL	In Master mode in idle state the spi clock	0x0*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
			polarity is low	
		SPI_CLK_POLARITY_INVERSE	In Master mode in idle state the spi clock polarity is high	0x1
0	SLAVE	SPI_SELECT_MASTER	Use the SPI interface in master mode	0x0*
		SPI_SELECT_SLAVE	Use the SPI interface in slave mode	0x1

11.7.5.2 SPI_CTRL

Bit Field	Read/Write	Field Name	Description
19	R	CS_STATUS	SPI CS status
18:17	R	MODE_STATUS	SPI mode status
16	R	ENABLE_STATUS	SPI enable status
9	W	CS_0	Lower the SPI chip-select line (master mode)
8	W	CS_1	Raise the SPI chip-select line (master mode)
7	W	MODE_NOP	Set mode to no operation
6	W	MODE_WRITE	Set mode to read operation
5	W	MODE_READ	Set mode to write operation
4	W	MODE_READ_WRITE	Set mode to read and write operation
3	W	START	Start a data transfer in master read mode
2	W	RESET	Reset the SPI interface
1	W	DISABLE	Disable the SPI interface
0	W	ENABLE	Enable the SPI interface

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
19	CS_STATUS	SPI_STATUS_CS_0	SPI CS is low	0x0
		SPI_STATUS_CS_1	SPI CS is high	0x1*
18:17	MODE_STATUS	SPI_STATUS_MODE_NOP	Mode is no operation (not allowed in DSPI and QSPI modes)	0x0*
		SPI_STATUS_MODE_WRITE	Mode is write data	0x1
		SPI_STATUS_MODE_READ	Mode is read data	0x2
		SPI_STATUS_MODE_READ_WRITE	Mode is read and write data (not allowed in DSPI and QSPI modes)	0x3

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
16	ENABLE_STATUS	SPI_STATUS_DISABLED	SPI interface is disabled	0x0*
		SPI_STATUS_ENABLED	SPI interface is enabled	0x1
9	CS_0	SPI_CS_0	Set the SPI CS signal low	0x1
8	CS_1	SPI_CS_1	Set the SPI CS signal high	0x1
7	MODE_NOP	SPI_MODE_NOP	Set the mode to no operation	0x1
6	MODE_WRITE	SPI_MODE_WRITE	Set the mode to write operation	0x1
5	MODE_READ	SPI_MODE_READ	Set the mode to read operation	0x1
4	MODE_READ_WRITE	SPI_MODE_READ_WRITE	Set the mode to read and write operation (not allowed in DSPI and QSPI modes)	0x1
3	START	SPI_START_READ	Start a transfer on the SPI interface (master read mode only)	0x1
2	RESET	SPI_RESET	Reset the SPI interface	0x1
1	DISABLE	SPI_DISABLE	Disable the SPI interface	0x1
0	ENABLE	SPI_ENABLE	Enable the SPI interface	0x1

11.7.5.3 SPI_STATUS

Bit Field	Read/Write	Field Name	Description
13	R	BUSY	Indicate that the reception or transmission of the data is ongoing
12	R	TX_REQ	Indicate that TX data can be written
11	R	RX_REQ	Indicate that RX data can be read
10	R	CS_RISE	Indicate that CS has risen in slave mode
9	R	OVERRUN	Indicate that an overrun has occurred when receiving data on the SPI interface
8	R	UNDERRUN	Indicate that an underrun has occurred when transmitting data on the SPI interface
4	W	TX_REQ_SET	Set TX_REQ status flag and clear internal TX buffer status
2	W	CS_RISE_CLEAR	Clear the CS rise status flag
1	W	OVERRUN_CLEAR	Clear the overrun status flag
0	W	UNDERRUN_CLEAR	Clear the underrun status flag

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
13	BUSY	SPI_IDLE	SPI idle	0x0*
		SPI_BUSY	SPI busy	0x1
12	TX_REQ	SPI_TX_NO_REQ	SPI TX data has already been written	0x0
		SPI_TX_REQ	SPI TX data can be written	0x1*
11	RX_REQ	SPI_RX_NO_REQ	No new SPI RX data available	0x0*
		SPI_RX_REQ	New SPI RX data available	0x1
10	CS_RISE	SPI_CS_RISE_FALSE	CS did not rise in slave mode	0x0*
		SPI_CS_RISE_TRUE	CS has risen in slave mode	0x1
9	OVERRUN	SPI_OVERRUN_FALSE	No SPI input overrun detected	0x0*
		SPI_OVERRUN_TRUE	SPI input overrun detected	0x1
8	UNDERRUN	SPI_UNDERRUN_FALSE	No SPI input underrun detected	0x0*
		SPI_UNDERRUN_TRUE	SPI input underrun detected	0x1
4	TX_REQ_SET	SPI_TX_REQ_SET	Set the TX_REQ status flag and clear internal TX buffer status	0x1
2	CS_RISE_CLEAR	SPI_CS_RISE_CLEAR	Clear the SPI CS rise status bit	0x1
1	OVERRUN_CLEAR	SPI_OVERRUN_CLEAR	Clear the SPI overrun status bit	0x1
0	UNDERRUN_CLEAR	SPI_UNDERRUN_CLEAR	Clear the SPI underrun status bit	0x1

11.7.5.4 SPI_TX_DATA

Bit Field	Read/Write	Field Name	Description
31:0	RW	TX_DATA	Single word buffer for data to be transmitted. When in master write or read_write mode, the transaction is started automatically

11.7.5.5 SPI_RX_DATA

Bit Field	Read/Write	Field Name	Description
31:0	R	RX_DATA	Single word buffer for received data. When in master read mode, a new transaction is started automatically

11.7.5.6 SPI_RX_DATA_NO_START

Bit Field	Read/Write	Field Name	Description
31:0	R	RX_DATA	Single word buffer for received data. Does not start a new transaction in master read mode, but does clear the RX_REQ flag

11.7.5.7 SPI_RX_DATA_MIRROR

Bit Field	Read/Write	Field Name	Description
31:0	R	RX_DATA	Single word buffer for received data. Does not start a new transaction and does clear the RX_REQ flag

11.7.5.8 SPI_ID_NUM

Bit Field	Read/Write	Field Name	Description
19:16	R	SPI_NUMBER	SPI Instance number
15:8	R	SPI_MAJOR_REVISION	SPI Major Revision number
7:0	R	SPI_MINOR_REVISION	SPI Minor Revision number

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
15:8	SPI_MAJOR_REVISION	SPI_MAJOR_REVISION	SPI revision 2.0	0x2*
7:0	SPI_MINOR_REVISION	SPI_MINOR_REVISION	SPI revision 2.0	0x0*

11.7.6 UART Registers

Register Name	Register Description	Address
UART0_CFG	UART Configuration Register	0x40000F00
UART0_CTRL	UART Control Register	0x40000F04
UART0_STATUS	UART Status Register	0x40000F08
UART0_TX_DATA	UART Transmit Data Register	0x40000F0C
UART0_RX_DATA	UART Receive Data Register	0x40000F10
UART0_ID_NUM	UART ID number	0x40000FFC

11.7.6.1 UART_CFG

Bit Field	Read/Write	Field Name	Description
21	RW	TX_DMA_ENABLE	Enable/disable the TX DMA request
20	RW	RX_DMA_ENABLE	Enable/disable the RX DMA request
19	RW	TX_END_INT_ENABLE	Enable/disable the TX end interrupt
18	RW	TX_START_INT_ENABLE	Enable/disable the TX start interrupt
17	RW	RX_INT_ENABLE	Enable/disable the RX interrupt
16	RW	OVERRUN_INT_ENABLE	Enable/disable the overrun interrupt
15:0	RW	CNT_STEP	Counter step size that configures the baud rate

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
21	TX_DMA_ENABLE	UART_TX_DMA_DISABLE	No TX DMA request is generated	0x0*
		UART_TX_DMA_ENABLE	A TX DMA request is generated when new data is requested by the UART interface	0x1
20	RX_DMA_ENABLE	UART_RX_DMA_DISABLE	No RX DMA request is generated	0x0*
		UART_RX_DMA_ENABLE	An RX DMA request is generated when new data is received by the UART interface	0x1
19	TX_END_INT_ENABLE	UART_TX_END_INT_DISABLE	No TX interrupt is raised	0x0*
		UART_TX_END_INT_ENABLE	A TX interrupt is raised when data transmission by the UART interface is finished	0x1
18	TX_START_INT_ ENABLE	UART_TX_START_INT_DISABLE	No TX interrupt is raised	0x0*
		UART_TX_START_INT_ENABLE	A TX interrupt is raised when new data is requested by the UART interface	0x1
17	RX_INT_ENABLE	UART_RX_INT_DISABLE	No RX interrupt is raised	0x0*
		UART_RX_INT_ENABLE	An RX interrupt is raised when new data is received by the UART interface	0x1
16	OVERRUN_INT_ ENABLE	UART_OVERRUN_INT_DISABLE	No ERROR interrupt is raised when an overrun is detected	0x0*
		UART_OVERRUN_INT_ENABLE	An ERROR interrupt is raised when an overrun occurs on the UART interface	0x1
15:0	CNT_STEP	UART_9600_BAUD	9600 baud at UARTCLK 1 MHz	0x9D4
		UART_14400_BAUD	14400 baud at UARTCLK 1 MHz	0xEBE
		UART_19200_BAUD	19200 baud at UARTCLK 1 MHz	0x13A8
		UART_38400_BAUD	38400 baud at UARTCLK 1 MHz	0x2751
		UART_57600_BAUD	57600 baud at UARTCLK 1 MHz	0x3AFA
		UART_115200_BAUD	115200 baud at UARTCLK 1 MHz	0x75F6
		UART_120000_BAUD	120000 baud at UARTCLK 1 MHz	0x7AE0
		UART_125000_BAUD	125000 baud at UARTCLK 1 MHz	0x7FFF

11.7.6.2 UART_CTRL

Bit Field	Read/Write	Field Name	Description
8	R	ENABLE_STATUS	UART enable status
2	W	RESET	Reset the UART interface
1	W	DISABLE	Disable the UART interface
0	W	ENABLE	Enable the UART interface

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
8	ENABLE_STATUS	UART_STATUS_DISABLED	UART interface is disabled	0x0*
		UART_STATUS_ENABLED	UART interface is enabled	0x1
2	RESET	UART_RESET	Reset the UART interface	0x1
1	DISABLE	UART_DISABLE	Disable the UART interface	0x1
0	ENABLE	UART_ENABLE	Enable the UART interface	0x1

11.7.6.3 UART_STATUS

Bit Field	Read/Write	Field Name	Description
12	R	TX_BUSY	Indicate that a TX transaction is ongoing
11	R	RX_BUSY	Indicate that a RX transaction is ongoing
10	R	TX_REQ	Indicate that a TX data can be written
9	R	RX_REQ	Indicate that a RX data can be read
8	R	OVERRUN	Indicate that an overrun occurred when receiving data
0	W	OVERRUN_CLEAR	Clear the overrun status flag

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
12	TX_BUSY	UART_TX_IDLE	No UART TX transaction is ongoing	0x0*
		UART_TX_BUSY	UART TX transaction is ongoing	0x1
11	RX_BUSY	UART_RX_IDLE	No UART RX transaction is ongoing	0x0*
		UART_RX_BUSY	UART RX transaction is ongoing	0x1
10	TX_REQ	UART_TX_NO_REQ	UART TX data has already been written	0x0
		UART_TX_REQ	UART TX data can be written	0x1*
9	RX_REQ	UART_RX_NO_REQ	No new UART RX data available	0x0*
		UART_RX_REQ	New UART RX data available	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
8	OVERRUN	UART_OVERRUN_FALSE	No UART RX overrun detected	0x0*
		UART_OVERRUN_TRUE	UART RX overrun detected	0x1
0	OVERRUN_CLEAR	UART_OVERRUN_CLEAR	Clear the RX overrun status bit	0x1

11.7.6.4 UART_TX_DATA

Bit Field	Read/Write	Field Name	Description
7:0	RW	TX_DATA	Transmitted data

11.7.6.5 UART_RX_DATA

Bit Field	Read/Write	Field Name	Description
7:0	R	RX_DATA	Received data

11.7.6.6 UART_ID_NUM

Bit Field	Read/Write	Field Name	Description
19:16	R	UART_NUMBER	UART Instance number
15:8	R	UART_MAJOR_REVISION	UART Major Revision number
7:0	R	UART_MINOR_REVISION	UART Minor Revision number

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
15:8	UART_MAJOR_ REVISION	UART_MAJOR_REVISION	UART revision 1.0	0x1*
7:0	UART_MINOR_ REVISION	UART_MINOR_REVISION	UART revision 1.0	0x0*

CHAPTER 12

Sensor Interfaces

This group of topics shows information for the Sensor interfaces and how they can be used with RSL15. The "Analog Comparator, SAR-ADC and LSAD Multiplexing Diagram" figure (Figure 61) shows the multiplexing scheme for the Analog Comparator, SAR-ADC, and LSAD Sensor interfaces, illustrating that each GPIO can be used for only one Sensor interface.

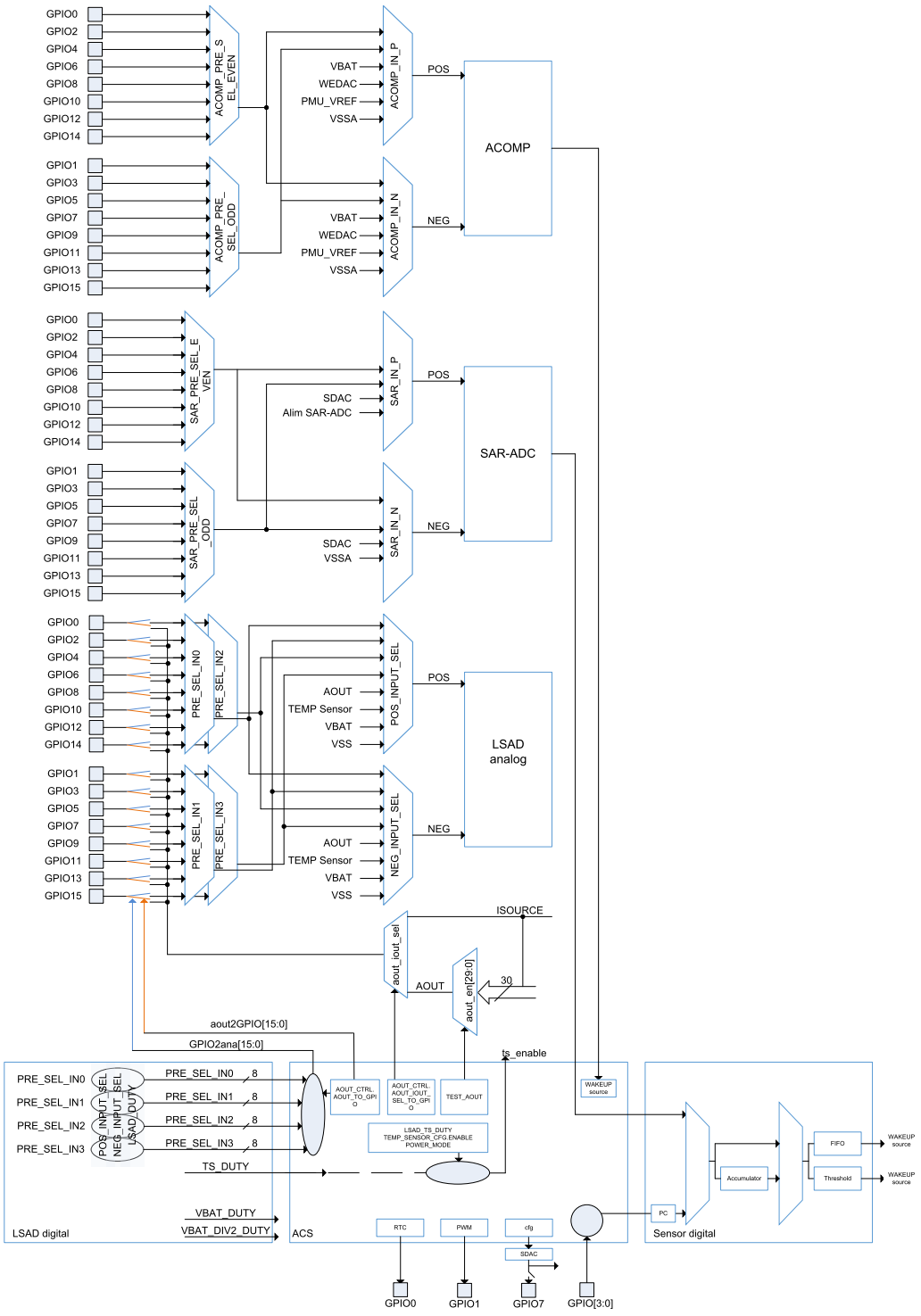


Figure 61. Analog Comparator, SAR-ADC and LSAD Multiplexing Diagram

12.1 Analog Comparator

12.1.1 Functional Description

The analog comparator is an analog general purpose comparator that can be used as analog wakeup, or for any other measurement that can be performed by means of a coarse comparator. The comparator inputs can be routed to any of the GPIO pins. Reference voltage can be selected from WEDAC voltage (the same that is used in SDAC) or PMU reference. The comparator has three modes: low-power, normal, and high-speed. Each mode provides different power consumption and delay trade-offs. The comparator employs a configurable hysteresis window for noisy input signals. It can also operate without any active hysteresis configuration.

12.1.2 Block Diagram

The "Analog Comparator Block Diagram" figure (Figure 62) shows the internal structure of the analog comparator.

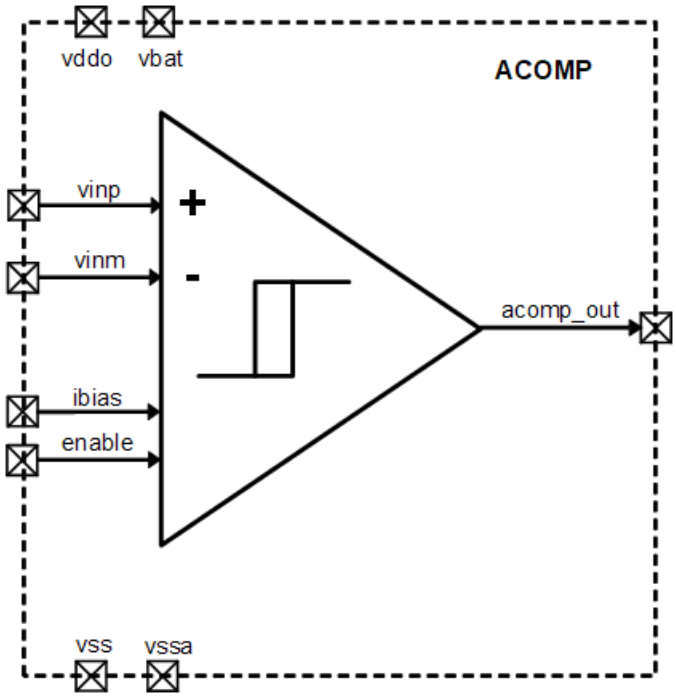


Figure 62. Analog Comparator Block Diagram

Table 29 shows the Analog Comparator's electrical specifications.

Table 29. Analog Comparator Electrical Specifications

Voltage Type	Voltage Range and Details
Input voltage: VBAT rail	Input common mode range: • 0 V to VBAT rail voltage, when VDDO voltage ≥ VBAT • Otherwise, input range is 0 V to VDDO voltage
Input offset voltage	Maximum 10 mV
Low-power mode comparator hysteresis	Adjustable from 40 mV to 90 mV • Delay: mihnimum 120 µs, maximum 800 µs • Power consumption: minimum 10 nA, maximum 50 nA
Normal mode comparator hysteresis	Adjustable from 40 mV to 100 mV • Delay: minimum 1.5 µs, maximum 10 µs • Power consumption: minimum 1.5 µA, maximum 5 µA
High-speed mode comparator hysteresis	Adjustable from 40 mV to 200 mV • Delay: minimum 0.2 µs, maximum 3 µs • Power consumption: minim um 18 µA, maximum 70 µA

For registers, see Section 12.9.1 “Analog Comparator Registers” on page 664.

12.2 SAR-ADC

12.2.1 Power Source

The SAR-ADC — successive approximation analog-to-digital converter — can be supplied by VBAT or via the DIO9, and the differential inputs are configurable via the `SENSOR_SAR_CFG` register.

If DIO9 is used as the SAR supply source, the `DIO_CFG[9].IO_MODE` field must be set to `DIO_MODE_DISABLE`.

12.2.2 Functional Description

The SAR-ADC is based on a standard successive approximation search algorithm. Its core blocks are shown in Figure 63: capacitive DAC, comparator, and digital control.

A conversion starts by sampling the input voltage on the capacitors in the capacitive DAC. The digital control then steps through the DAC codes, starting at the MSB (largest DAC element). The comparator indicates whether the digital control code is higher or lower than the sampled signal.

The SAR-ADC is clocked using SENSOR_CLK, which is configured as described in Section 12.8.1 “Clock Source Configuration” on page 661.

The input of the SAR-ADC is differential.

12.2.3 Block Diagram

The "SAR-ADC Block Diagram" figure (Figure 63) shows the internal structure of the SAR-ADC converter.

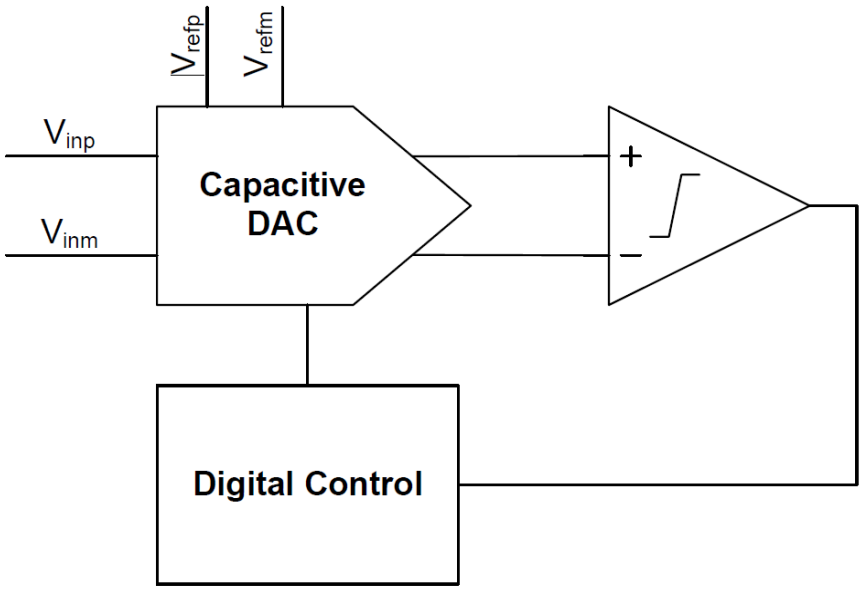


Figure 63. SAR-ADC Block Diagram

To allow for smaller, faster, and lower power analog operation, the calibrated 12-bit SAR-ADC is modified from the standard approach in several ways. First, the DAC capacitors are sized smaller than is necessary for 12-bit accuracy. Only the lowest 7 bits are sized for inherent matching. To achieve 12-bit performance, the upper five bits are implemented with six DAC elements. The sizes of these six elements ensure that with mismatch there are no dead areas in the ADC output. As a result, these elements are not binarily scaled, and the DAC code does not match the ADC output code. To create a 12-bit accurate ADC code, the digital block sums the weight values of the Capacitive DAC elements. The "Typical DAC Element Weights" table (Table 30) shows the typical DAC element weights.

Table 30. Typical DAC Element Weights

Bit	Weight	Component
0	1	Normally only for calibration
1	2	Normally only for calibration
2	4	
3	8	
4	16	
5	32	
6	64	
7	128	
8	256	
9	384	Calibrated

Table 30. Typical DAC Element Weights (Continued)

Bit	Weight	Component
10	704	Calibrated
11	1280	Calibrated
12	2304	Calibrated
13	4096	Calibrated
14	7105	Calibrated
Total	16384	

The nine LSBs are binarily scaled, and their weights are fixed. The weights for the six MSBs are a function of matching, and vary from chip to chip. Once these values are determined, the ADC output is constructed from the final SAR DAC code and the weights as:

$$outcode = \sum_{i=0}^{14} DACcode\,[i] \cdot weight\,[i]$$

The minimum (unsigned) output code is 0, and, based on the weights in Table 30, the maximum output code is 16384, the sum of all weights. The ADC output code requires 15 bits to represent the full range of the ADC.

12.2.4 ADC Calibration

The accuracy of the final output code of the ADC is dependent on the accuracy of the element weights. As noted previously, the weights for the six largest DAC elements vary from chip to chip. A calibration routine is required to determine the correct weights for these elements. This calibration routine is built into the digital control block of the ADC. It measures the proper weight code for each of the six MSB elements eight times. The average of the eight measurements with two additional sub bits is used as the final weight value. The calibration routine requires 650 clock cycles to complete, and must be performed when the ADC reference voltages are settled and with input voltages around mid range. Subsequent changes in supply voltage and reference voltages do not require re-calibration. This allows the calibration to be run once after the system is powered up.

12.2.5 ADC Outputs

SAR-ADC outputs are provided as a fraction of the selected supply range (as defined by the power supply reference).

- The output code can be selected to `signed` or `unsigned` (see the `OUT_SEL` configuration in Section 12.9.2 “SAR-ADC Registers” on page 666).
  - In unsigned mode the output ranges typically from 0 to 16384.
  - In signed mode the output ranges typically from -8192 to 8192.
- In either case (signed or unsigned), the output code can exceed the minimum and maximum values, depending on the calibrated offsets and weights. No saturation is performed inside the block to minimize the gate count. But as the output has one guard MSB bit, there is no risk of result misinterpretation.
- Scaling: the ADC output code is a 15-bit bus. This allows the code to include the effects of the weight sub-bits. The 13 MSBs of the output code can be used to confine the output to 13 bits. If a 12-bit output is needed, additional saturation needs to be implemented to discard the MSB.

- Internal offset: the comparator in the ADC adds a DC offset to the output code. The current ADC design compensates this offset during the calibration process. When the offset is compensated, the result is an output code which is in the middle of the range when the analog inputs are set to the same voltage $(Vrefp - Vrefm)/2$ by register configuration.
- Auto zeroing: another offset compensation method allows compensating external circuits inaccuracies (such as external offset, unbalanced reference voltages, etc.). To measure this offset, the analog inputs have to be set by register configuration to $(Vrefp - Vrefm)/2$ and a conversion started in auto zeroing mode. The result of this conversion is registered and subtracted from subsequent conversions.

A certain number of clock cycles are needed for conversion of each sample, as follows:

- If the `SENSOR_SAR_CTRL_MODE` field of the `SENSOR_SAR_CTRL` register = `SAR_CONV_12BIT` (value 0x0, which is the default), conversion takes 14 clock cycles.
- If the `SENSOR_SAR_CTRL_MODE` field of the `SENSOR_SAR_CTRL` register = `SAR_CONV_14BIT` (value 0x1), conversion takes 16 clock cycles.

12.2.6 Timing Diagrams

The following diagrams illustrate SAR-ADC timing for two different modes. The "Calibration and One Shot Conversion Mode" figure (Figure 64) shows timing for the calibration and one shot conversion mode, while the "Continuous Conversion Mode (Mode = 0x0 or 0x1)" figure (Figure 65) illustrates continuous conversion mode timing.

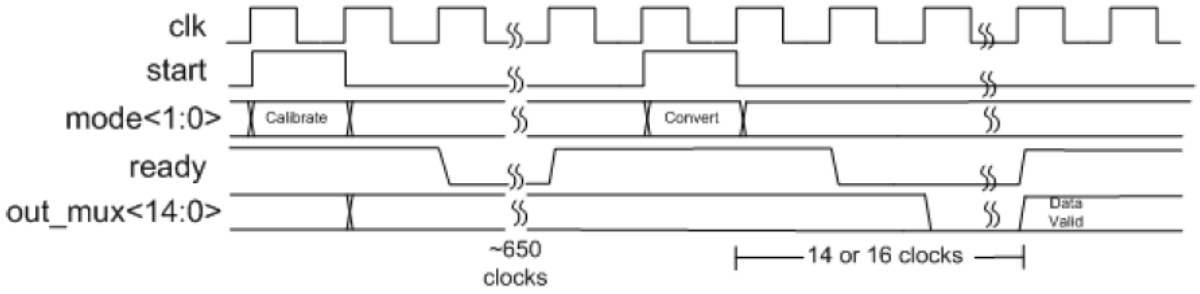


Figure 64. Calibration and One Shot Conversion Mode

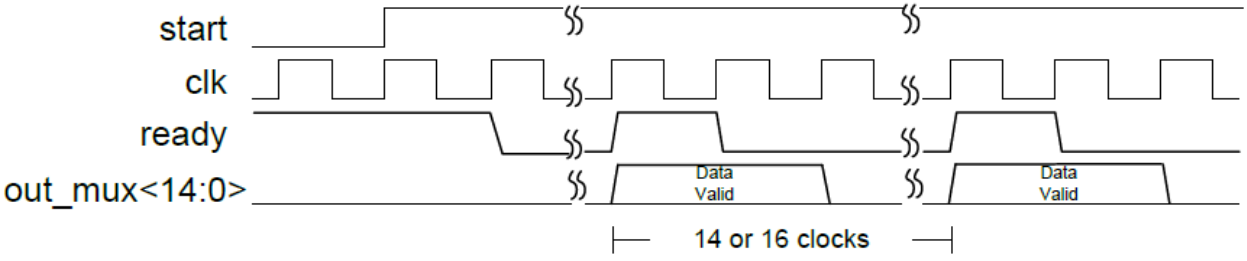


Figure 65. Continuous Conversion Mode (Mode = 0x0 or 0x1)

For registers, see Section 12.9.2 “SAR-ADC Registers” on page 666.

12.3 Pulse Counter

The pulse counter is used to observe an input signal and count the number of clock periods, during either of the following situations:

- A rising edge is observed
- The signal is at a high level

The pulse counter is an input to the ultra-low power data acquisition system (see Section 12.8 “Ultra-Low Power Data Acquisition Subsystem”), and is clocked using the SENSOR_CLK defined by that block. The pulse counter is reset with the SAR-ADC when the `SENSOR_CTRL_RESET` bit in the `SENSOR_CTRL` register is set.

The pulse counter is configured using the `SENSOR_PC_CFG` register, as follows:

- Select between counting rising edges and high-level periods using the `SENSOR_PC_CFG_PC_COUNT_ON` bit.
- Select the input signal to the pulse counter from one of GPIO0 to GPIO4, or from a constant source, using the `SENSOR_PC_CFG_PC_SRC_SEL` bit field.
- Set the measurement duration of between 1 and 1023 cycles using the `SENSOR_PC_CFG_COUNT_INT` bit field.

The "Example counting using the pulse counter" figure (Figure 66) shows how the pulse counter is updated in each mode based on an example input.

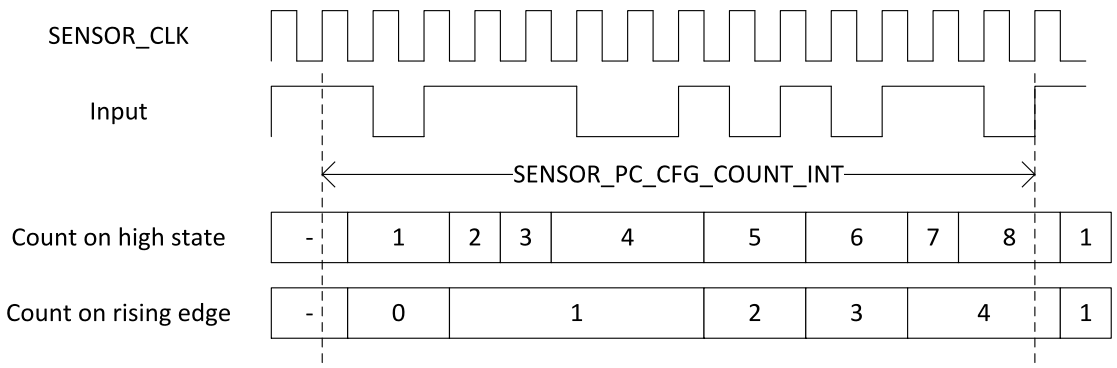


Figure 66. Example counting using the pulse counter

The current value of the pulse counter can be read through the `SENSOR_PC_COUNT` register, and the output from the pulse counter is captured using the ultra-low power data acquisition system's accumulator, FIFO, or threshold blocksSection 12.8 “Ultra-Low Power Data Acquisition Subsystem”

NOTE: If the pulse counter is clocked using the RTC (a clock which is not synchronous to SYSCLK), the current count read from the `SENSOR_PC_COUNT` register can be unstable. Therefore, to maintain data integrity, ensure that the data is read repeatedly until two consecutive reads match.

For registers, see Section 12.9.3 “Pulse Counter Registers” on page 669.

12.4 LSAD

The low speed analog-to-digital converters (LSADs) 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 and restrictions based on sampling frequency. The output code is always 14 bits. For example, the highest frequency divisor (0x08, SLOWCLK/20) results in 8-bit resolution, and the lowest speed (0x7: SLOWCLK/6400) has the full 14 bits of resolution. The LSAD has a conversion range from 0 V to 2 V.

12.4.1 LSAD Input Configuration

The purpose of the LSADs 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 LSAD block are configured using the `NEG_INPUT_SEL` and `POS_INPUT_SEL` bit fields from the `LSAD_INPUT_SEL` register set. The negative and positive signals used for each differential measurement are selected from:

All DIOs

For more information about GPIO configuration for LSADs, see Chapter 10.1 "Overview" on page 511.

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 LSADs to additionally measure a number of other internal power supply outputs and status flags with the `AOUT_*` and `DOUT_*` fields, respectively, of the `ACS_AOUT_CTRL` register. For more information about the internal power supplies, see Section 8.2 "Power Supply Overview" on page 406.

VBAT

Direct measurement of the battery supply voltage. Setting this bit divides the battery supply voltage that the LSAD measures by 2. This allows the battery supply voltage to always stay in the measurable range of the LSAD.

CAUTION: The LSAD measurable input voltage range is 0 V to 2 V. Exceeding this voltage range can damage the device (see the datasheet ABSOLUTE MAXIMUM RATINGS table for details). Therefore, when measuring a GPIO, take care not to exceed the absolute maximum voltage on the selected GPIO. The internal VBAT signal is protected by a voltage divider, allowing safe measurement of all valid VBAT ranges.

Internal Temperature Sensor

Allows measurement of the output of the internal temperature sensor, allowing a simple way to measure internal device temperature.

IMPORTANT: There are gain and offset values available in NVR7 that are intended for use with the LSAD. There is a separate set of gain and offset values in NVR7 for measuring the internal temperature sensor using the LSADs. Usage of gain and offset values from NVR7 is discussed below Section 12.4.3 "LSAD Output Data" on page 654.

GND

Allows measurement of the ground of the device. This input is most often used as a negative input, in order to measure a positive voltage. If both positive and negative inputs are connected to ground, automatic offset compensation is enabled and automatically measured and applied by the device. The offset value measured is placed into LSAD_OFFSET.

12.4.2 LSAD Sampling Configuration

The LSAD signals are sampled at a sampling rate derived from SLOWCLK, and configured using the LSAD_CFG_FREQ bit field from the LSAD_CFG register. Configuration of SLOWCLK is described in Section 7.4.3 “Slow Clock (SLOWCLK)” on page 391, with SLOWCLK typically initialized to 1 MHz. LSAD 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. LSAD 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 LSAD measurement results have a resolution of 14 bits, or up to 50 kHz where LSAD measurement results have a resolution of 8 bits. In addition to selecting the sampling mode, the LSAD_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 LSAD measurement. (See the "LSAD Sampling Configuration Settings" table (Table 31).)

The LSAD 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 LSAD block is configured in Normal Mode, each LSAD channel is sampled in sequence and an LSAD interrupt occurs once every eighth sample. The channel number that triggers the interrupt (or phase offset) can be defined as part of the LSAD_INT_CH_NUM bit field in the LSAD_INT_ENABLE register. Between each interrupt, the data value for each LSAD channel is updated to a new, valid sample.
- If the LSAD is configured in Continuous Mode, only one LSAD channel is sampled and an interrupt occurs every sample. The sampled channel is defined in the LSAD_INT_CH_NUM bit field as part of the LSAD_INT_ENABLE register.

The "LSAD Sampling Configuration Settings" table (Table 31) shows the LSAD sampling configuration settings.

Table 31. LSAD Sampling Configuration Settings

Value	Setting	Fixed Divider	Configurable Division	Native Voltage Range (V)	Effective Division of SLOWCLK (Normal Mode)	Effective Division of SLOWCLK (Continuous Mode)
0x0	LSAD_DISABLE	N/A	LSAD disabled	N/A	LSAD disabled	LSAD disabled
0x1	LSAD_PRESCALE_200	10	Divide by 20	-0.125 to 2.125	Divide by 1600	Divide by 200
0x2	LSAD_PRESCALE_400	10	Divide by 40	-0.063 to 2.063	Divide by 3200	Divide by 400

Table 31. LSAD Sampling Configuration Settings (Continued)

Value	Setting	Fixed Divider	Configurable Division	Native Voltage Range (V)	Effective Division of SLOWCLK (Normal Mode)	Effective Division of SLOWCLK (Continuous Mode)
0x3	LSAD_PRESCALE_640	10	Divide by 64	-0.031 to 2.031	Divide by 5120	Divide by 640
0x4	LSAD_PRESCALE_800	10	Divide by 80	-0.016 to 2.016	Divide by 6400	Divide by 800
0x5	LSAD_PRESCALE_1600	10	Divide by 160	-0.008 to 2.008	Divide by 12800	Divide by 1600
0x6	LSAD_PRESCALE_3200	10	Divide by 320	-0.004 to 2.004	Divide by 25600	Divide by 3200
0x7	LSAD_PRESCALE_6400	10	Divide by 640	-0.002 to 2.002	Divide by 51200	Divide by 6400
0x8	LSAD_PRESCALE_20H	2	Divide by 10	-1.0 to 3.0	Divide by 160	Divide by 20
0x9	LSAD_PRESCALE_40H	2	Divide by 20	-0.125 to 2.125	Divide by 320	Divide by 40
0xA	LSAD_PRESCALE_80H	2	Divide by 40	-0.063 to 2.063	Divide by 512	Divide by 80
0xB	LSAD_PRESCALE_128H	2	Divide by 64	-0.031 to 2.031	Divide by 640	Divide by 128
0xC	LSAD_PRESCALE_160H	2	Divide by 80	-0.016 to 2.016	Divide by 1280	Divide by 160
0xD	LSAD_PRESCALE_320H	2	Divide by 160	-0.008 to 2.008	Divide by 2560	Divide by 320
0xE	LSAD_PRESCALE_640H	2	Divide by 320	-0.004 to 2.004	Divide by 5120	Divide by 640
0xF	LSAD_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 LSAD samples all eight channels in Normal mode at a configurable rate between 6.25 kHz and 19.5 Hz.

12.4.3 LSAD Output Data

The converted output from the most recent conversion of each channel can be found in the LSAD_DATA_TRIM_CH* and LSAD_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.

*LSAD_DATA_TRIM_CH**

The output data value from the LSAD 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.

*LSAD_DATA_AUDIO_CH**

The output data value from the LSAD 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 LSAD sampling configuration extends the range of measured values.

For improved accuracy in the LSAD data, the RSL15 SoC provides an offset correction factor in the LSAD_OFFSET register that automatically applies a compensation value to the measured LSAD data. This value is normally the difference between the expected and measured LSAD 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 LSAD 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). If you are using LSAD gain and offset values from NVR7, we recommend that you disable automatic offset compensation and apply the gain and offset settings in firmware.

12.4.4 Voltage Monitoring

The LSAD block additionally provides resources that can be used to monitor any voltage measured by the LSAD, to provide an alarm if that voltage drops below a certain threshold. Typically, this 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 nearing the end of its 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 LSAD_MONITOR_CFG_MONITOR_THRESHOLD bit field. Select the LSAD channel as the source to monitor by setting the LSAD_MONITOR_CFG_MONITOR_SRC bit. Both of these values are part of the LSAD_MONITOR_CFG register.

When the monitored voltage is measured as being below the specified threshold, the value of the LSAD_MONITOR_COUNT_VAL register is incremented. This counter register is reset when read. The value of this register is compared with the value stored in the LSAD_MONITOR_CFG_ALARM_COUNT_VALUE bit field from the LSAD_MONITOR_CFG register, and if the counter value ever exceeds the alarm value, an alarm is generated.

The status of the monitor and the LSAD module can be read or reset from the LSAD_MONITOR_STATUS register. The status of the monitor alarm, data overruns, and new LSAD sample data availability is all ready to be read and reset.

12.4.5 LSAD and Voltage Monitoring Interrupt

A single interrupt line is shared between the LSAD and the voltage monitoring circuitry. This interrupt can be independently configured to trigger when one or both of the following conditions are met:

- Triggering when a specified LSAD channel is sampled. To enable this trigger condition, use the LSAD_INT_ENABLE bit from the LSAD_INT_ENABLE register. To specify the LSAD channel, use the LSAD_INT_CH_NUM bit field from this same register.
- Triggering when a voltage monitor alarm occurs. To enable this trigger condition, set the ALARM_COUNT_VALUE bit field from the LSAD_MONITOR_CFG register.

NOTE: All three flags in the LSAD_MONITOR_STATUS register are sticky, and each must be separately cleared by setting the appropriate LSAD_MONITOR_STATUS_*_CLEAR bit from the LSAD_MONITOR_STATUS register.

12.4.6 Using LSAD calibration data

Using the LSAD calibration data available in NVR7 is a straightforward process that can result in very good accuracy without any additional calibration on the part of the end user. When using the gain and offset data from NVR7, it is recommended to disable automatic offset compensation for best accuracy. Details for how to configure automatic offset compensation are found in "LSAD Output Data" on the previous page. There are two sets of gain and offset values stored in NVR7, one each for the Low and High Frequency Modes. Both values are stored in signed integer format and must be converted to floating point values. The code below uses the low frequency gain and offset values. The resulting floating point values are in Volts.

```
TRIM_Type *trim = TRIM;
float offset_error = (float)((int16_t)trim->lsad_trim.lf_offset) / 32768.0f;
float gain_error = (float)(trim->lsad_trim.lf_gain) / 65536.0f;
```

Once the gain and offset values have been converted, and the LSAD data has been converted to a floating point voltage (in Volts), the gain and offset can be applied. The equation below demonstrates how to apply the gain and offset.

$$\text{Compensated voltage} = (\text{measured voltage} - \text{offset})/\text{gain}$$

For registers, see Section 12.9.4 “LSAD Registers” on page 670.

12.5 Simple Low-Power DAC

12.5.1 Functional Description

The SDAC (Simple Digital-to-Analog Converter) is a buffer with selectable gain (G=1 or G=2), which buffers the WEDAC voltage generated in the bandgap reference. Its output can be used as reference or sensor supply on the PCB; to do this, the user needs to configure GPIO[7] as GPIO_MODE_DISABLE, and route the output through this AIO. Current drive is limited, because the AIO has a series resistance of 500 Ω and the signal is multiplexed with other analog signals. If a higher current drive is needed, an external buffer has to be used.

To configure the SDAC as output,

1. Set the GPIO[7] IO mode to disabled.
2. Set the ACS_SDAC_CFG_SDAC_TO_GPIO7 field in the ACS_SDAC_CFG register (see Section 12.5 “Simple Low-Power DAC” on page 655).

The SDAC output can also be connected internally to the SAR-ADC reference input using the SENSOR_SAR_CFG_SAR_IN_P or SENSOR_SAR_CFG_SAR_IN_N fields in the SENSOR_SAR_CFG register. This provides a low impedance buffer to the WEDAC reference voltage for the SAR-ADC.

The "Simple DAC Electrical Specifications" table (Table 32) shows the simple DAC's electrical specifications.

Table 32. Simple DAC Electrical Specifications

Voltage Type	Voltage Range
Input voltage VDDA rail	Common input mode range: 0 V to +2.6 V
Input offset voltage with regard to input reference	-15 mV to +15 mV
Output voltage	0.1 V to VDDA-0.2 V NOTE: Cannot exceed VDDO when connected to GPIO.
Output current	Maximum 10 µA
Voltage gains	Can be set to 1 (buffer) or 2 (amplifier) Gain error maximum: 2%
Current consumption	Maximum 5 µA

For more information on the simple DAC's electrical specifications, see the RSL15 Datasheet.

For registers, see Section 12.9.5 "SDAC Registers" on page 679.

12.6 Current Source

12.6.1 Description

The current source is a module designed to provide a consistent, tunable current source via output on a GPIO pin. Outputting this constant current source allows measurement of temperature via the addition of a thermistor in series from the selected GPIO pad to ground.

A trimmable current of 1 µA to 16 µA (default 10 µA) can be sourced to a chosen GPIO pin. The LSAD measures the voltage on the IO pin; this voltage can be used to calculate the actual temperature.

To improve accuracy, it is recommended that you perform a four-point measurement when using the current source. There are many switches in series to connect the current externally, causing some voltage drop. This can easily be improved by using two GPIOs, one to output the current and another one to measure the exact voltage on the positive voltage node. See the "Thermistor Configuration Using Two GPIOs" figure (Figure 67), which demonstrates utilizing GPIO6 as the output of the current source, and using GPIO2 to measure the resulting voltage.

For registers, see Section 12.9.6 "Current Source Registers" on page 680.

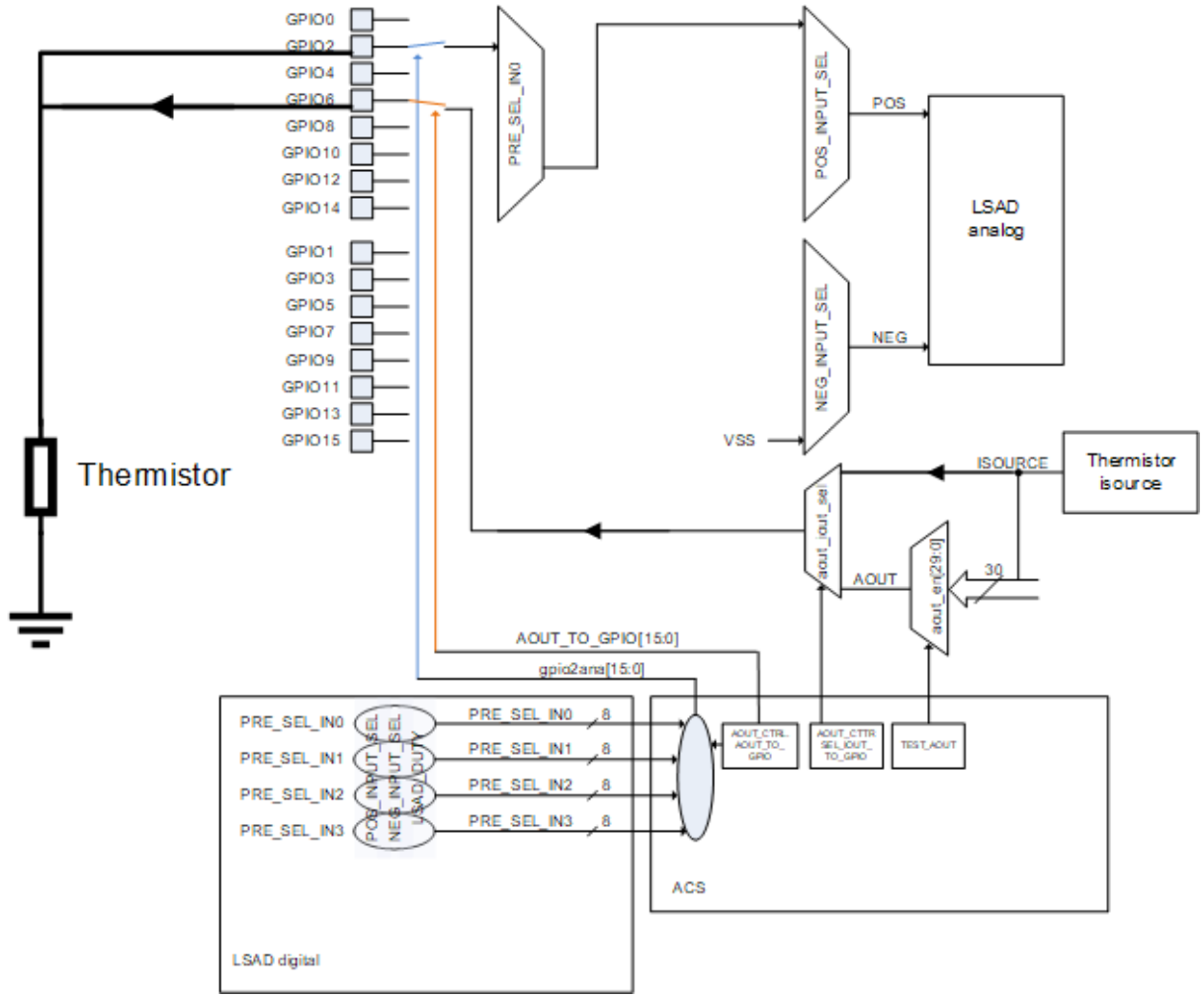


Figure 67. Thermistor Configuration Using Two GPIOs

NOTE: The impedance characteristic of the thermistor can be logarithmic; therefore, to obtain a valid input range for the LSAD such that it can be measured correctly, the scale of the current needs to be changed.

12.6.2 Current Source Configuration

The current source can be configured to output a current between 1 uA and 16 uA. The current is linearly set in the ACS_TEMP_CURR_CURRENT_VALUE bit-field in the ACS_TEMP_CURR_CFG register, starting with a setting of 0 corresponding to 1 uA. The current output can also be trimmed to a maximum of +/- 10% from the default trimming. This can be set in the same register's ACS_TEMP_CURR_CURRENT_TRIM bit-field. In order to lower power consumption, the current source can be dynamically enabled according to the LSAD duty cycle. For example, if GPIO6 is used to measure the voltage generated by the current source, the current source is only enabled when the LSAD conversion on the selected GPIO6 is occurring. The GPIO duty cycle to use is selected via the ACS_TEMP_CURR_CFG_GPIO_IN_USE bit-field in the ACS_TEMP_CURR_CFG register. In this example, GPIO6 is chosen as the GPIO that the LSAD channel is connected to.

12.6.3 Temperature Calculation

The temperature can be calculated using the traditional thermistor formula:

$$T = \frac{B}{ln\left(\frac{R_{thermistor}}{R_0 \times e^{\frac{-B}{T_0}}}\right)}$$

- T [°K]: measured temperature
- B [°K]: thermistor variation coefficient over temperature. It is given by the thermistor component datasheet
- R0 [Ω]: thermistor resistor at T0
- T0 [°K]: reference temperature (25°C)
- Rthermistor [Ω] given by the LSAD output voltage and thermistor current value

$$Rthermistor = V_{LSAD} / I_{thermistor}$$

To obtain maximum accuracy, it is recommended that you adjust the thermistor current to have an LSAD voltage close to the middle of the LSAD range, around 1.0 V. This can be accomplished with firmware, by adjusting the thermistor current ($I_{thermistor}$) using the field `ACS_TEMP_CURR_CFG :CURRENT_VALUE`.

12.7 Internal Temperature Sensor

An on-chip temperature sensor provides a device internal temperature reading to the LSAD. This temperature sensor requires no external components.

The temperature sensor output voltage is read by the LSAD and converted to a digital value. (For information on configuring the LSAD to measure the internal temperature, see Section 12.4 “LSAD” on page 651.) The temperature sensor has offset and gain measurements stored in NVR7, separate from the standard gain and offset values used for regular LSAD measurements.

An offset one point calibration is necessary to reach the specified temperature accuracy. Typically the gain does not need to be calibrated; but for measurements that require higher accuracy, a 2 point gain and offset calibration can be performed.

To calculate the final compensated temperature, the formula below can be used:

$$Temperature = \frac{LSAD\_DATA\_LSB}{TEMP\_SENSOR\_GAIN} - TEMP\_SENSOR\_OFFSET$$

Where `LSAD_DATA_LSB` is the digital value received from the LSAD, `TEMP_SENSOR_GAIN` is the value in LSB/Co for the temperature sensor, typically 21.3 LSB/°Celsius, and `TEMP_SENSOR_OFFSET` is a static offset in Celsius, typically 334° Celsius. However, these values vary across devices, and if higher accuracy is desired, you need to calibrate for the values.

Stored in NVR7 is a set of measurements intended to aid software development, since the accuracy of the temperature measurement at production, when NVR7 is written, is +/- 5° Celsius — this does not refer to the overall accuracy of the temperature sensor. The temperatures used for these data points are 30° Celsius and 90° Celsius.

A higher-accuracy calibration by the user is advised for applications that require more precise temperature measurements.

12.7.1 Enabling the Temperature Sensor

The temperature sensor is enabled or disabled in the ACS_TEMP_SENSOR_CFG register. If the ACS_TEMP_SENSOR_CFG_DUTY_TEMP_SENS bit of the ACS_TEMP_SENSOR_CFG register is set, the temperature sensor is enabled automatically when the appropriate LSAD channel is sampled. If the ACS_TEMP_SENSOR_CFG_DUTY_TEMP_SENS bit is cleared, the temperature sensor is instead enabled via the ACS_TEMP_SENSOR_CFG_ENABLE bit in the ACS_TEMP_SENSOR_CFG register.

12.7.2 Calibration Procedures

A one-point calibration is sufficient to generate the TEMP_SENSOR_OFFSET at a single temperature. A two-point calibration is required to generate the TEMP_SENSOR_GAIN between the two temperature points. Using more calibration points in a curve fitting calibration procedure will result in higher temperature accuracy.

12.7.2.1 One-Point Calibration

To perform a one-point calibration, follow these steps:

1. Place the device in a temperature controlled location. In the equation that follows, the temperature used is called T_{CAL}. The TEMP_SENSOR_GAIN used is the average value, 21.3 LSB/°Celsius.
2. Enable the LSAD and the internal temperature sensor, and configure the LSAD to measure the internal temperature. (See Section 12.4 “LSAD” on page 651 for information on how to perform this configuration.)
3. Retrieve the result from the LSAD measurement, in LSBs. In the equation that follows, this values is called D_{out} (Tcal).
4. Use the following equation to calculate the offset:

$$TEMP\_SENSOR\_OFFSET = \frac{D_{OUT(Tcal)}}{TEMP\_SENSOR\_GAIN} - T_{cal}$$

12.7.2.2 Two-Point Calibration

Two-point calibration allows the calculation of an accurate gain for the device temperature sensor between the two temperature points. Accurate reference temperatures must be used. The two point calibration method can be extended to multiple temperature points using curve fitting for higher accuracy.

The two-point calibration can be performed by following these steps:

1. Place the device in a temperature-controlled location. In the equation that follows, the temperature used is called T_{cal1}.
2. Enable the LSAD and the internal temperature sensor, and configure the LSAD to measure the internal temperature. (See Section 12.4 “LSAD” on page 651 for more information on how to perform this configuration.)
3. Retrieve the result from the LSAD measurement, in LSBs. In the equation that follows, this value is called D_{out} (Tcal1).
4. Repeat steps 1-3 at the second reference temperature. This is referenced as T_{cal2}.
5. The temperature sensor gain can be calculated with the formula below:

$$TEMP\_SENSOR\_GAIN = \frac{D_{OUT(Tcal2)} - D_{OUT(Tcal1)}}{T_{cal2} - T_{cal1}}$$

For registers, see Section 12.9.7 “Internal Temperature Sensor Registers” on page 681.

12.8 Ultra-Low Power Data Acquisition Subsystem

The ultra-low power (ULP) data acquisition subsystem collects data from the SAR-ADC (see Section 12.2 “SAR-ADC ” on page 646) and pulse counter (see Section 12.3 “Pulse Counter” on page 650), optionally routing this data through an accumulator, and distributing the data through a FIFO and threshold detection circuitry for further processing in the rest of the RSL15 system.

An overview of the ULP data acquisition subsystem is provided in the "Top-Level Overview of the Data Acquisition Subsystem" figure (Figure 68).

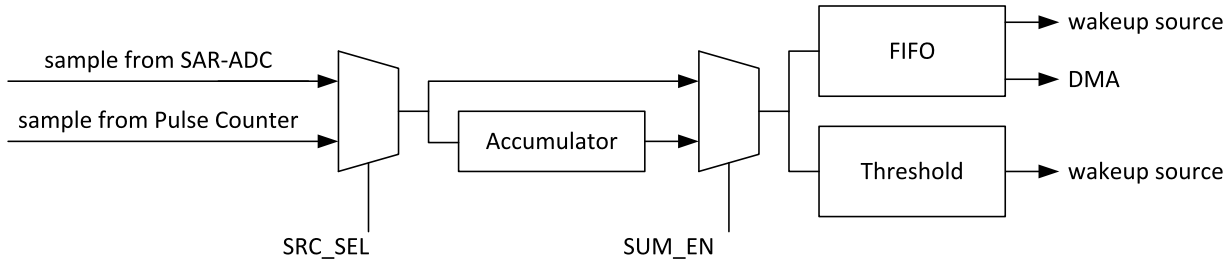


Figure 68. Top-Level Overview of the Data Acquisition Subsystem

The ULP data acquisition subsystem is designed to be used in any power mode, and can be used to trigger a transition between Run Mode and one of the following:

- A low power mode (such as Standby Mode)
- Smart Sense Mode

For more information on power modes, see Section 8.6 “Power Modes” on page 431.

The ULP data acquisition subsystem is controlled by the `SENSOR_CTRL` register:

- Reset the ULP data acquisition subsystem, including the SAR-ADC, by setting the `SENSOR_CTRL_RESET` bit
- The subsystem can be enabled or disabled using the `SENSOR_CTRL_ENABLE` bit.
    - If the ULP data acquisition subsystem is currently sampling from the SAR-ADC when disabled, the sample conversion is completed before the system transitions to an idle state.

IMPORTANT: If the ULP data acquisition subsystem is reset while using the SAR-ADC (whether used with the ULP data acquisition subsystem or separately), the SAR-ADC must be re-calibrated.

The state of the ULP data acquisition subsystem is also available in the `SENSOR_CTRL` register:

- The `SENSOR_CTRL_STATUS` bit indicates if the ULP data acquisition subsystem is idle or active. Since the ULP data acquisition subsystem might not be disabled immediately when the `SENSOR_CTRL_ENABLE` bit is cleared, this status bit is used to indicate when the subsystem actually becomes idle.

- The SENSOR_CTRL_STATE bit indicates if the ULP data acquisition subsystem is currently sampling data using the pulse counter or SAR-ADC, or if it is in an idle/delay state.

12.8.1 Clock Source Configuration

The ULP data acquisition subsystem is clocked using SENSOR_CLK or STANDBYCLK, as selected via the SENSOR_CFG_CLK_SEL bit from the SENSOR_CFG register. If SENSOR_CLK is selected as the clock source, the ULP data acquisition subsystem disables SENSOR_CLK when switching to sleep or standby power modes. For more information on power modes, see Section 8.6 “Power Modes”.

IMPORTANT: The ULP data acquisition subsystem must be disabled when switching between clock sources, as SENSOR_CLK and STANDBYCLK are not synchronous and the resulting clock switch cannot otherwise be guaranteed to be glitch free.

SENSOR_CLK is divided from SYSCLK using the CLK_CFG1_SENSOR_CLK_PRESCALE bit-field from the CLK_CFG1 register. If not used, SENSOR_CLK can be disabled using the CLK_CFG1_SENSOR_CLK_DISABLE bit from the CLK_CFG1 register.

12.8.2 Input Configuration

The input to the ULP data acquisition subsystem is controlled by the SENSOR_CFG_SRC_SEL bit from the SENSOR_CFG register, selecting between sourcing data from the SAR-ADC (see Section 12.2 “SAR-ADC ” on page 646) and the pulse counter (see Section 12.3 “Pulse Counter” on page 650).

NOTE: While the pulse counter is used as the input to the ULP data acquisition subsystem, the SAR-ADC input can continue to be used in parallel using the SENSOR_SAR_DATA register or the DMA block (see Section A.8 “DMA Controller” on page 786) to obtain data from the SAR-ADC.

The time between retrieving input data samples (and the delay before reading the first sample) from the SAR-ADC can be configured using the SENSOR_CFG register:

- Enable the sampling delay by setting the SENSOR_CFG_DLY_EN bit.
- Configure the sampling delay, as a multiple of the selected clock source of between 1 and 1024 cycles, using the SENSOR_CFG_DLY_DIV_EN bit-field.

NOTE: Sampling delays for the SAR-ADC do not apply when accessing SAR-ADC data using the SENSOR_SAR_DATA register or the DMA block.

12.8.3 Sample Accumulation

The input samples can be accumulated before being stored and post-processed. When the accumulator is enabled, samples output from the accumulation block is used by the sample storage and signal threshold detection blocks in place of using the input samples directly.

The accumulator sums the number of samples specified by the SAMPLE_PROCESSING_NBR_SAMPLES bit-field, from the SAMPLE_PROCESSING register whenever it is enabled using the SAMPLE_PROCESSING_SUM_EN bit from the same register. The summation time duration is equivalent to the duration of one sample, multiplied by the number of samples selected in the SAMPLE_PROCESSING_NBR_SAMPLES bit-field. The accumulator can be used to accumulate (n − 1) samples at a time, with the last sample in a set of n samples needing to be manually read from SENSOR_SAR_

DATA and added to the accumulated result when an event occurs. If the accumulator is used with the ULP sensor's FIFO, every n^{th} sample is ignored, because the last sample in each set is not manually accumulated when using the FIFO or threshold detector.

IMPORTANT: The SAMPLE_PROCESSING_NBR_SAMPLES bit-field from the SAMPLE_PROCESSING register is shared with the threshold detection block. If both sample accumulation and threshold detection are enabled, they use the same value for this bit field.

12.8.4 Sample Storage

The samples collected by the ULP data acquisition subsystem can be stored to a FIFO dedicated to the subsystem. The FIFO is based around a 16-entry buffer stored to the SENSOR_FIFO_DATA* registers. The FIFO configuration and status of the FIFO buffer is provided by the SENSOR_FIFO_CFG register:

- To enable sample storage in the FIFO, set the SENSOR_FIFO_CFG_FIFO_STORE_EN bit.
- To set the size of the FIFO used, configure the SENSOR_FIFO_CFG_FIFO_SIZE bit-field.
- To query the current FIFO level, read the SENSOR_FIFO_CFG_FIFO_LEVEL bit-field.

The FIFO supports triggering three different events when it becomes full, as indicated by the FIFO level being equal to or greater than the defined FIFO size. When the FIFO becomes full, the following events are triggered if their conditions are met:

- A FIFO interrupt is triggered when the ULP data acquisition subsystem is clocked using SENSOR_CLK, and the SENSOR_FIFO_CFG_FIFO_RX_INT_EN bit is set.
- A wakeup event and WAKEUP interrupt is triggered, indicating the wakeup has been triggered by the FIFO_FULL wakeup source, when the ULP data acquisition subsystem is clocked using STANDBYCLK. This includes ULP data acquisition subsystem operation in all power modes other than run mode. For more information about power modes, see Section 8.6 "Power Modes" on page 431.
- A DMA trigger is sent to start one or more DMA channel operations when the SENSOR_FIFO_CFG_FIFO_RX_DMA_EN bit is set. For more information about DMA triggers and DMA channel operations, see Section 13.4 "Direct Memory Access (DMA) Controller" on page 690.

IMPORTANT: The FIFO level provided by the SENSOR_FIFO_CFG_FIFO_LEVEL bit-field is updated asynchronously to the rest of the system, when the ULP data acquisition subsystem is clocked using STANDBYCLK. If the FIFO level is not pointing to the last entry in the FIFO buffer, this value needs to be read until the same value is read twice consecutively, to confirm the actual level of the FIFO has been read cleanly.

NOTE: The FIFO does not reset when there is a digital reset, and the FIFO buffer contains undefined data at system start up. The SENSOR_FIFO_CFG_FIFO_LEVEL bit-field indicates which data in the FIFO is valid.

The FIFO level is reset when the first entry in the FIFO buffer is read. When reading the first sample in the FIFO, all other valid data needs to be read as well — before it is overwritten with new samples. Practically, this means that the second and subsequent samples must be read no more than one sampling period of the SAR-ADC or pulse counter (including delays and accumulation) after reading of the previous sample.

The FIFO does not provide any error handling for edge cases:

- The FIFO does not support overflow detection. If additional samples are received when the FIFO is full, new samples continue to be written to the FIFO buffer with the FIFO level saturated at the last entry in the 16-entry buffer, which is overwritten repeatedly until the first value for the FIFO is read and the FIFO level is reset.
- The FIFO does not support underflow detection. Data read from the FIFO buffer past the FIFO level read from the SENSOR_FIFO_CFG_FIFO_LEVEL bit-field prior to reading the first sample in the buffer, is undefined.
- The FIFO does not support protection of received samples after the first entry in the FIFO buffer (the first sample is protected by not allowing overflow of the FIFO buffer).

12.8.5 Signal Threshold Detection

The ULP data acquisition subsystem contains a threshold detection block that can be used alongside the sample storage block. The threshold block is used to compare the values created by the rest of the ULP data acquisition subsystem to determine if a number of consecutive received samples are:

- Above or equal to the maximum threshold detection level
- Below or equal to the minimum threshold detection level
- Outside of the range defined by the minimum and maximum threshold detection levels

Threshold detection is enabled whenever one or both thresholds are enabled:

- The maximum threshold is configured using the SENSOR_THRESHOLD_MAX register. The level for the maximum threshold is set by the SENSOR_THRESHOLD_MAX_THRESHOLD_MAX bit-field, and this threshold is enabled using the SENSOR_THRESHOLD_MAX_THRESHOLD_MAX_EN bit.
- The minimum threshold is configured using the SENSOR_THRESHOLD_MIN register. The level for the minimum threshold is set by the SENSOR_THRESHOLD_MIN_THRESHOLD_MIN bit-field, and this threshold is enabled using the SENSOR_THRESHOLD_MIN_THRESHOLD_MIN_EN bit.

The number of consecutive samples that need to meet a threshold condition before a wakeup event occurs is configured using the SAMPLE_PROCESSING_NBR_SAMPLES bit-field from the SAMPLE_PROCESSING register.

IMPORTANT: The SAMPLE_PROCESSING_NBR_SAMPLES bit-field from the SAMPLE_PROCESSING register is shared with the sample accumulation block. If both sample accumulation and threshold detection are enabled, they use the same value for this bit field.

The threshold block is used to trigger a wakeup event and WAKEUP interrupt, indicating the wakeup has been triggered by the THRESHOLD wakeup source. This event is triggered when one or both of the enabled threshold conditions is met over the specified number of consecutive samples.

For registers, see Section 12.9.8 “ULP Registers” on page 682.

12.9 Sensor Interfaces Registers

For the user’s convenience, the registers for the sensor interfaces are grouped in separate sections according to interface.

12.9.1 Analog Comparator Registers

Register Name	Register Description	Address
ACS_ACOMP_CFG	ACOMP configuration register	0x40001B2C
ACS_ACOMP_OUT	ACOMP out	0x40001B30

12.9.1.1 ACS_ACOMP_CFG

Bit Field	Read/Write	Field Name	Description
22:20	RW	ACOMP_PRE_SEL_ODD	Pre selection of odd GPIOs
18:16	RW	ACOMP_PRE_SEL_EVEN	Pre selection of even GPIOs
14:12	RW	ACOMP_IN_P	Defines the positive input signal
10:8	RW	ACOMP_IN_N	Defines the negative input signal
6:4	RW	ACOMP_HYST	Hysteresis level
2:1	RW	MODE	Power mode selection
0	RW	ENABLE	Enable signal

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
22:20	ACOMP_PRE_SEL_ ODD	ACOMP_PRE_SEL_GPIO_1	Select GPIO[1] for "pre_sel_odd"	0x0*
		ACOMP_PRE_SEL_GPIO_3	Select GPIO[3] for "pre_sel_odd"	0x1
		ACOMP_PRE_SEL_GPIO_5	Select GPIO[5] for "pre_sel_odd"	0x2
		ACOMP_PRE_SEL_GPIO_7	Select GPIO[7] for "pre_sel_odd"	0x3
		ACOMP_PRE_SEL_GPIO_9	Select GPIO[9] for "pre_sel_odd"	0x4
		ACOMP_PRE_SEL_GPIO_11	Select GPIO[11] for "pre_sel_odd"	0x5
		ACOMP_PRE_SEL_GPIO_13	Select GPIO[13] for "pre_sel_odd"	0x6
		ACOMP_PRE_SEL_GPIO_15	Select GPIO[15] for "pre_sel_odd"	0x7
18:16	ACOMP_PRE_SEL_ EVEN	ACOMP_PRE_SEL_GPIO_0	Select GPIO[0] for "pre_sel_even"	0x0*
		ACOMP_PRE_SEL_GPIO_2	Select GPIO[2] for "pre_sel_even"	0x1
		ACOMP_PRE_SEL_GPIO_4	Select GPIO[4] for "pre_sel_even"	0x2
		ACOMP_PRE_SEL_GPIO_6	Select GPIO[6] for "pre_sel_even"	0x3
		ACOMP_PRE_SEL_GPIO_8	Select GPIO[8] for "pre_sel_even"	0x4
		ACOMP_PRE_SEL_GPIO_10	Select GPIO[10] for "pre_sel_even"	0x5
		ACOMP_PRE_SEL_GPIO_12	Select GPIO[12] for "pre_sel_even"	0x6

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		ACOMP_PRE_SEL_GPIO_14	Select GPIO[14] for "pre_sel_even"	0x7
14:12	ACOMP_IN_P	ACOMP_IN_P_SRC_PRE_SEL_EVEN	Select "pre_sel_even" as source	0x0
		ACOMP_IN_P_SRC_PRE_SEL_ODD	Select "pre_sel_odd" as source	0x1
		ACOMP_IN_P_SRC_VSSA	Select VSSA as source	0x2*
		ACOMP_IN_P_SRC_WEDAC	Select WEDAC as source	0x3
		ACOMP_IN_P_SRC_PMU_VREF	Select PMU Vref as source	0x4
		ACOMP_IN_P_SRC_VBAT	Select VBAT as source	0x5
		ACOMP_IN_P_SRC_NOT_CONNECTED	Not connect the source	0x6
10:8	ACOMP_IN_N	ACOMP_IN_N_SRC_PRE_SEL_EVEN	Select "pre_sel_even" as source	0x0
		ACOMP_IN_N_SRC_PRE_SEL_ODD	Select "pre_sel_odd" as source	0x1
		ACOMP_IN_N_SRC_VSSA	Select VSSA as source	0x2*
		ACOMP_IN_N_SRC_WEDAC	Select WEDAC as source	0x3
		ACOMP_IN_N_SRC_PMU_VREF	Select PMU Vref as source	0x4
		ACOMP_IN_N_SRC_VBAT	Select VBAT as source	0x5
		ACOMP_IN_N_SRC_NOT_CONNECTED	Not connect the source	0x6
6:4	ACOMP_HYST	HYST_0	No hysteresis	0x0*
		HYST_10MV	10 mV hysteresis	0x1
		HYST_20MV	20 mV hysteresis	0x2
		HYST_30MV	30 mV hysteresis	0x3
		HYST_40MV	40 mV hysteresis	0x4
		HYST_50MV	50 mV hysteresis	0x5
		HYST_60MV	60 mV hysteresis	0x6
		HYST_70MV	70 mV hysteresis	0x7
2:1	MODE	ACOMP_LOW_POWER_MODE	Low power mode	0x0
		ACOMP_NORMAL_MODE	Normal mode	0x1*
		ACOMP_HIGH_SPEED_MODE	High Speed mode	0x2
0	ENABLE	ACOMP_DIS	ACOMP disabled	0x0*
		ACOMP_EN	ACOMP enabled	0x1

12.9.1.2 ACS_ACOMP_OUT

Bit Field	Read/Write	Field Name	Description
0	R	ACOMP_OUT	Comparison result

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
0	ACOMP_OUT	ACOMP_OUT_0	Vinn > Vinp	0x0*
		ACOMP_OUT_1	Vinn < Vinp	0x1

12.9.2 SAR-ADC Registers

Register Name	Register Description	Address
SENSOR_SAR_CFG	SAR configuration	0x40001D00
SENSOR_SAR_CTRL	SAR control register	0x40001D04
SENSOR_SAR_DATA	SAR data out	0x40001D08

12.9.2.1 SENSOR_SAR_CFG

Bit Field	Read/Write	Field Name	Description
22:20	RW	SAR_PRE_SEL_ODD	Pre-selection of odd GPIOs
18:16	RW	SAR_PRE_SEL_EVEN	Pre-selection of even GPIOs
14:12	RW	SAR_IN_P	Defines the positive input signal
10:8	RW	SAR_IN_N	Defines the negative input signal
3	RW	SAR_DATA_OUT_RX_DMA_EN	Enable / disable the DMA trigger when SAR data available
2	RW	SAR_DATA_OUT_UPDATE	Enable / disable the SAR data update
1	RW	SAR_SUPPLY_EN	Enable / disable the SAR supply
0	RW	SAR_SUPPLY_SELECT	SAR supply selection

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
22:20	SAR_PRE_SEL_ODD	SAR_PRE_SEL_GPIO_1	Select GPIO[1] for "pre_sel_odd"	0x0*
		SAR_PRE_SEL_GPIO_3	Select GPIO[3] for "pre_sel_odd"	0x1
		SAR_PRE_SEL_GPIO_5	Select GPIO[5] for "pre_sel_odd"	0x2
		SAR_PRE_SEL_GPIO_7	Select GPIO[7] for "pre_sel_odd"	0x3
		SAR_PRE_SEL_GPIO_9	Select GPIO[9] for "pre_sel_odd"	0x4
		SAR_PRE_SEL_GPIO_11	Select GPIO[11] for "pre_sel_odd"	0x5

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		SAR_PRE_SEL_GPIO_13	Select GPIO[13] for "pre_sel_odd"	0x6
		SAR_PRE_SEL_GPIO_15	Select GPIO[15] for "pre_sel_odd"	0x7
18:16	SAR_PRE_SEL_EVEN	SAR_PRE_SEL_GPIO_0	Select GPIO[0] for "pre_sel_even"	0x0*
		SAR_PRE_SEL_GPIO_2	Select GPIO[2] for "pre_sel_even"	0x1
		SAR_PRE_SEL_GPIO_4	Select GPIO[4] for "pre_sel_even"	0x2
		SAR_PRE_SEL_GPIO_6	Select GPIO[6] for "pre_sel_even"	0x3
		SAR_PRE_SEL_GPIO_8	Select GPIO[8] for "pre_sel_even"	0x4
		SAR_PRE_SEL_GPIO_10	Select GPIO[10] for "pre_sel_even"	0x5
		SAR_PRE_SEL_GPIO_12	Select GPIO[12] for "pre_sel_even"	0x6
		SAR_PRE_SEL_GPIO_14	Select GPIO[14] for "pre_sel_even"	0x7
14:12	SAR_IN_P	SAR_IN_P_SRC_PRE_SEL_EVEN	Select "pre_sel_even" as source	0x0
		SAR_IN_P_SRC_PRE_SEL_ODD	Select "pre_sel_odd" as source	0x1
		SAR_IN_P_SRC_SDAC	Select SDAC as source	0x2
		SAR_IN_P_SRC_SAR_SUPPLY	Select SAR-ADC supply as source	0x3
		SAR_IN_P_SRC_SAR_SUPPLY_DIV2	Select SAR-ADC supply divided by 2 as source	0x4*
10:8	SAR_IN_N	SAR_IN_N_SRC_PRE_SEL_EVEN	Select "pre_sel_even" as source	0x0
		SAR_IN_N_SRC_PRE_SEL_ODD	Select "pre_sel_odd" as source	0x1
		SAR_IN_N_SRC_SDAC	Select SDAC as source	0x2
		SAR_IN_N_SRC_VSSA	Select VSSA as source	0x3
		SAR_IN_N_SRC_SAR_SUPPLY_DIV2	Select SAR-ADC supply divided by 2 as source	0x4*
3	SAR_DATA_OUT_RX_DMA_EN	SAR_DATA_OUT_RX_DMA_DISABLED	No RX DMA triggered	0x0*
		SAR_DATA_OUT_RX_DMA_ENABLED	RX DMA triggered when SAR data available	0x1
2	SAR_DATA_OUT_UPDATE	SAR_DATA_OUT_UPDATE_AUTO	The SAR data buffer updated at the end of a conversion	0x0*
		SAR_DATA_OUT_UPDATE_ENABLE	The SAR data buffer updated at each clock cycle	0x1
1	SAR_SUPPLY_EN	SAR_SUPPLY_AUTO	The SAR supply is in auto mode	0x0*
		SAR_SUPPLY_ENABLE	The SAR supply is enabled.	0x1
0	SAR_SUPPLY_SELECT	SAR_SUPPLY_BY_VBAT	SAR ADC supplied by VBAT	0x0*
		SAR_SUPPLY_BY_GPIO9	SAR ADC supplied by GPIO9	0x1

12.9.2.2 SENSOR_SAR_CTRL

Bit Field	Read/Write	Field Name	Description
14	W	START_SINGLE	Start single conversion command
13	RW	START_CONTINUOUS	Start continuous conversions command
12	R	BUSY	Conversion in progess flag
9:8	RW	MODE	Conversion mode selection
6:4	RW	NUM_SAMPLE	Defines the number of clk cycles for data sampling
3:0	RW	OUT_SEL	Output selection (undefined values return 0)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
14	START_SINGLE	SAR_START_SINGLE	Start single conversion	0x1
13	START_CONTINUOUS	SAR_START_CONTINUOUS	Start continuous conversions	0x1
		SAR_STOP_CONTINUOUS	Stop continuous conversions	0x0*
12	BUSY	SAR_IDLE	Idle or conversion done	0x0*
		SAR_BUSY	Conversion in progress	0x1
9:8	MODE	SAR_CONV_12BIT	Convert 12 bits	0x0*
		SAR_CONV_14BIT	Convert 14 bits	0x1
		SAR_CAL_WEIGHT	Gain compensation (calibration)	0x2
		SAR_AUTO_ZERO	Offset compensation (calibration)	0x3
6:4	NUM_SAMPLE	SAR_NSAMPLING_CYCLE_1	1 cycle	0x0*
		SAR_NSAMPLING_CYCLE_2	2 cycles	0x1
		SAR_NSAMPLING_CYCLE_3	3 cycles	0x2
		SAR_NSAMPLING_CYCLE_4	4 cycles	0x3
		SAR_NSAMPLING_CYCLE_5	5 cycles	0x4
		SAR_NSAMPLING_CYCLE_6	6 cycles	0x5
		SAR_NSAMPLING_CYCLE_7	7 cycles	0x6
		SAR_NSAMPLING_CYCLE_8	8 cycles	0x7
3:0	OUT_SEL	SAR_SEL_GATED_SIGNED_COMPENSATED	Gated 2's complement offset compensated output	0x0*
		SAR_SEL_GATED_SIGNED	Gated 2's complement output	0x1
		SAR_SEL_GATED_UNSIGNED_COMPENSATED	Gated unsigned offset compensated output	0x2
		SAR_SEL_GATED_UNSIGNED	Gated unsigned output	0x3

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		SAR_SEL_ACCUMULATOR	Accumulator output (test purposes)	0x4
		SAR_SEL_DAC	DAC signal (to analog) selection (test purposes)	0x5
		SAR_SEL_COMP_CNT	Comparator output (from analog) and counters selection (test purposes)	0x6
		SAR_SEL_OFFSET	Select offset compensation register (calibration)	0x8
		SAR_SEL_WEIGHT_BIT9	Select bit 9 weight gain compensation register (calibration)	0x9
		SAR_SEL_WEIGHT_BIT10	Select bit 10 weight gain compensation register (calibration)	0xA
		SAR_SEL_WEIGHT_BIT11	Select bit 11 weight gain compensation register (calibration)	0xB
		SAR_SEL_WEIGHT_BIT12	Select bit 12 weight gain compensation register (calibration)	0xC
		SAR_SEL_WEIGHT_BIT13	Select bit 13 weight gain compensation register (calibration)	0xD
		SAR_SEL_WEIGHT_BIT14	Select bit 14 weight gain compensation register (calibration)	0xE

12.9.2.3 SENSOR_SAR_DATA

Bit Field	Read/Write	Field Name	Description
15:0	R	SAR_OUT	15 bit SAR conversion result (signed)

12.9.3 Pulse Counter Registers

Register Name	Register Description	Address
SENSOR_PC_CFG	Sensor Pulse Counter Configuration	0x40001D0C
SENSOR_PC_COUNT	Sensor Pulse Counter Current Value	0x40001D10

12.9.3.1 SENSOR_PC_CFG

Bit Field	Read/Write	Field Name	Description
16	RW	PC_COUNT_ON	Pulse counter count on rising edge or high state
14:12	RW	PC_SRC_SEL	Pulse count source selection
9:0	RW	COUNT_INT	Duration of count state

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
16	PC_COUNT_ON	PC_COUNT_ON_HIGH_LEVEL	Count on high level	0x0*
		PC_COUNT_ON_RISING_EDGE	Count on rising edge	0x1
14:12	PC_SRC_SEL	PC_SRC_GPIO_0	Select GPIO[0] as source	0x0
		PC_SRC_GPIO_1	Select GPIO[1] as source	0x1
		PC_SRC_GPIO_2	Select GPIO[2] as source	0x2
		PC_SRC_GPIO_3	Select GPIO[3] as source	0x3
		PC_SRC_CONST_LOW	Select constant low as source	0x4
		PC_SRC_CONST_HIGH	Select constant high as source	0x5*
9:0	COUNT_INT	COUNT_INT_1	Number of SENSOR_CLK periods for "Count Sample State": 1	0x0
		COUNT_INT_2	Number of SENSOR_CLK periods for "Count Sample State": 2	0x1
		COUNT_INT_3	Number of SENSOR_CLK periods for "Count Sample State": 3	0x2
		COUNT_INT_4	Number of SENSOR_CLK periods for "Count Sample State": 4	0x3*
		COUNT_INT_256	Number of SENSOR_CLK periods for "Count Sample State": 256	0xFF
		COUNT_INT_1023	Number of SENSOR_CLK periods for "Count Sample State": 1023	0x3FE

12.9.3.2 SENSOR_PC_COUNT

Bit Field	Read/Write	Field Name	Description
9:0	R	VALUE	Sensor pulse counter current value

12.9.4 LSAD Registers

Register Name	Register Description	Address
LSAD_DATA_TRIM_CH	LSAD conversion result for channel 0 to 7 in trimmer mode	0x40001C00-0x40001C1C
LSAD_DATA_AUDIO_CH	LSAD conversion result for channel 0 to 7 in audio mode (signed)	0x40001C20-0x40001C3C
LSAD_INPUT_SEL	LSAD input selection for channel 0 to 7	0x40001C40-0x40001C5C

Register Name	Register Description	Address
LSAD_CFG	LSAD Configuration Register	0x40001C60
LSAD_OFFSET	LSAD conversion result for LSAD GND	0x40001C64
LSAD_INT_ENABLE	LSAD Interrupt Mask Register	0x40001C68
LSAD_MONITOR_CFG	Monitoring Configuration Register	0x40001C6C
LSAD_MONITOR_COUNT_VAL	Monitoring Status Register	0x40001C70
LSAD_MONITOR_STATUS	LSAD / MONITOR Status Register	0x40001C74
LSAD_PRE_SEL_INPUT	LSAD pre-selection for input 0	0x40001C78
LSAD_DUTY	LSAD duty config	0x40001C7C
LSAD_ID_NUM	LSAD ID Number	0x40001CFC

12.9.4.1 LSAD_DATA_TRIM_CH

Bit Field	Read/Write	Field Name	Description
13:0	R	DATA	14-bit LSAD conversion result

12.9.4.2 LSAD_DATA_AUDIO_CH

Bit Field	Read/Write	Field Name	Description
31:0	R	DATA	14-bit LSAD conversion result (sign extended to 32 bits)

12.9.4.3 LSAD_INPUT_SEL

Bit Field	Read/Write	Field Name	Description
6:4	RW	POS_INPUT_SEL	Positive input selection
2:0	RW	NEG_INPUT_SEL	Negative input selection

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
6:4	POS_INPUT_SEL	LSAD_POS_INPUT_SEL0	Select pre-selection input 0 as positive input	0x0
		LSAD_POS_INPUT_SEL1	Select pre-selection input 1 as positive input	0x1
		LSAD_POS_INPUT_SEL2	Select pre-selection input 2 as positive input	0x2
		LSAD_POS_INPUT_SEL3	Select pre-selection input 3 as positive input	0x3
		LSAD_POS_INPUT_AOUT	Select AOUT as positive input	0x4

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		LSAD_POS_INPUT_TEMP	Select temperature sensor as positive input	0x5
		LSAD_POS_INPUT_VBAT_DIV2	Select VBAT/2 as positive input	0x6*
		LSAD_POS_INPUT_GND	Select LSAD internal ground as positive input	0x7
2:0	NEG_INPUT_SEL	LSAD_NEG_INPUT_SEL0	Select pre-selection input 0 as negative input	0x0
		LSAD_NEG_INPUT_SEL1	Select pre-selection input 1 as negative input	0x1
		LSAD_NEG_INPUT_SEL2	Select pre-selection input 2 as negative input	0x2
		LSAD_NEG_INPUT_SEL3	Select pre-selection input 3 as negative input	0x3
		LSAD_NEG_INPUT_AOUT	Select AOUT as negative input	0x4
		LSAD_NEG_INPUT_TEMP	Select temperature as negative input	0x5
		LSAD_NEG_INPUT_VBAT_DIV2	Select VBAT/2 as negative input	0x6
		LSAD_NEG_INPUT_GND	Select LSAD internal ground as negative input	0x7*

12.9.4.4 LSAD_CFG

Bit Field	Read/Write	Field Name	Description
4	RW	CONTINUOUS_MODE	LSAD continuously sampling the channel selected as interrupt source (LSAD_INT_CH_NUM)
3:0	RW	FREQ	Defines the sampling frequency of the LSAD channels

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
4	CONTINUOUS_MODE	LSAD_NORMAL	Normal mode, all 8 channels sampled	0x0*
		LSAD_CONTINUOUS	Continuous mode: only one channel sampled (for test purpose)	0x1
3:0	FREQ	LSAD_DISABLE	LSAD disabled	0x0*
		LSAD_PRESCALE_200	Sample rate is SLOWCLK/200 (low speed mode)	0x1
		LSAD_PRESCALE_400	Sample rate is SLOWCLK/400 (low speed mode)	0x2
		LSAD_PRESCALE_640	Sample rate is SLOWCLK/640 (low	0x3

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
			speed mode)	
		LSAD_PRESCALE_800	Sample rate is SLOWCLK/800 (low speed mode)	0x4
		LSAD_PRESCALE_1600	Sample rate is SLOWCLK/1600 (low speed mode)	0x5
		LSAD_PRESCALE_3200	Sample rate is SLOWCLK/3200 (low speed mode)	0x6
		LSAD_PRESCALE_6400	Sample rate is SLOWCLK/6400 (low speed mode)	0x7
		LSAD_PRESCALE_20H	Sample rate is SLOWCLK/20 (high speed mode)	0x8
		LSAD_PRESCALE_40H	Sample rate is SLOWCLK/40 (high speed mode)	0x9
		LSAD_PRESCALE_80H	Sample rate is SLOWCLK/80 (high speed mode)	0xA
		LSAD_PRESCALE_128H	Sample rate is SLOWCLK/128 (high speed mode)	0xB
		LSAD_PRESCALE_160H	Sample rate is SLOWCLK/160 (high speed mode)	0xC
		LSAD_PRESCALE_320H	Sample rate is SLOWCLK/320 (high speed mode)	0xD
		LSAD_PRESCALE_640H	Sample rate is SLOWCLK/640 (high speed mode)	0xE
		LSAD_PRESCALE_1280H	Sample rate is SLOWCLK/1280 (high speed mode)	0xF

12.9.4.5 LSAD_OFFSET

Bit Field	Read/Write	Field Name	Description
14:0	RW	DATA	15-bit LSAD signed offset

12.9.4.6 LSAD_INT_ENABLE

Bit Field	Read/Write	Field Name	Description
3:1	RW	LSAD_INT_CH_NUM	Channel number triggering the LSAD interrupt
0	RW	LSAD_INT_ENABLE	The LSAD new sample ready interrupt mask

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
3:1	LSAD_INT_CH_NUM	LSAD_INT_CH0	The LSAD interrupt is triggered when the LSAD_DATA_CH0 register is updated	0x0*
		LSAD_INT_CH1	The LSAD interrupt is triggered when the LSAD_DATA_CH1 register is updated	0x1
		LSAD_INT_CH2	The LSAD interrupt is triggered when the LSAD_DATA_CH2 register is updated	0x2
		LSAD_INT_CH3	The LSAD interrupt is triggered when the LSAD_DATA_CH3 register is updated	0x3
		LSAD_INT_CH4	The LSAD interrupt is triggered when the LSAD_DATA_CH4 register is updated	0x4
		LSAD_INT_CH5	The LSAD interrupt is triggered when the LSAD_DATA_CH5 register is updated	0x5
		LSAD_INT_CH6	The LSAD interrupt is triggered when the LSAD_DATA_CH6 register is updated	0x6
		LSAD_INT_CH7	The LSAD interrupt is triggered when the LSAD_DATA_CH7 register is updated	0x7
0	LSAD_INT_ENABLE	LSAD_INT_DIS	This source cannot set an interrupt	0x0*
		LSAD_INT_EN	This source can set the LSAD interrupt line	0x1

12.9.4.7 LSAD_MONITOR_CFG

Bit Field	Read/Write	Field Name	Description
23:16	RW	ALARM_COUNT_VALUE	An Alarm Status bit is set and an interrupt generated when SUPPLY_COUNT_VALUE = ALARM_COUNT_VALUE
15:8	RW	MONITOR_THRESHOLD	Low voltage detection threshold (7.8 mV steps)
2:0	RW	MONITOR_SRC	Selects the source channel to be monitored

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
23:16	ALARM_COUNT_VALUE	MONITOR_ALARM_NONE	No Alarm is triggered	0x0*
		MONITOR_ALARM_COUNT1	Alarm count value is 1	0x1
		MONITOR_ALARM_COUNT255	Alarm count value is 255	0xFF
15:8	MONITOR_THRESHOLD	MONITOR_THRESHOLD_LOW	Lowest voltage threshold: 7.8 mV	0x0
		MONITOR_THRESHOLD_MID	Mid voltage threshold: 1.4 V	0xB3*
		MONITOR_THRESHOLD_HIGH	Highest voltage threshold: ~2.0 V	0xFF

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
2:0	MONITOR_SRC	MONITOR_CH0	Channel 0 is monitored	0x0
		MONITOR_CH1	Channel 1 is monitored	0x1
		MONITOR_CH2	Channel 2 is monitored	0x2
		MONITOR_CH3	Channel 3 is monitored	0x3
		MONITOR_CH4	Channel 4 is monitored	0x4
		MONITOR_CH5	Channel 5 is monitored	0x5
		MONITOR_CH6	Channel 6 is monitored	0x6
		MONITOR_CH7	Channel 7 is monitored	0x7*

12.9.4.8 LSAD_MONITOR_COUNT_VAL

Bit Field	Read/Write	Field Name	Description
7:0	R	MONITOR_COUNT_VALUE	Number of times the voltage has fallen below the monitor voltage threshold. The counter is reset when read.

12.9.4.9 LSAD_MONITOR_STATUS

Bit Field	Read/Write	Field Name	Description
12	W	MONITOR_ALARM_CLEAR	Monitoring alarm status bit
9	W	LSAD_OVERRUN_CLEAR	LSAD Overrun condition
8	W	LSAD_READY_CLEAR	LSAD new sample Ready status bit
4	R	MONITOR_ALARM_STAT	Monitoring alarm status bit
1	R	LSAD_OVERRUN_STAT	LSAD Overrun condition
0	R	LSAD_READY_STAT	LSAD new sample Ready status bit

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
12	MONITOR_ALARM_ CLEAR	MONITOR_ALARM_CLEAR	Writing a 1 clears the MONITOR Alarm status bit	0x1
9	LSAD_OVERRUN_ CLEAR	LSAD_OVERRUN_CLEAR	Writing a 1 clears the LSAD Overrun status bit	0x1
8	LSAD_READY_CLEAR	LSAD_READY_CLEAR	Writing a 1 clears the LSAD Ready status bit	0x1
4	MONITOR_ALARM_ STAT	MONITOR_ALARM_FALSE	MONITOR Alarm flag not set	0x0*
		MONITOR_ALARM_TRUE	MONITOR Alarm has been triggered	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
1	LSAD_OVERRUN_STAT	LSAD_OVERRUN_FALSE	No LSAD Overrun detected	0x0*
		LSAD_OVERRUN_TRUE	LSAD Overrun detected	0x1
0	LSAD_READY_STAT	LSAD_READY_FALSE	No new LSAD samples available	0x0*
		LSAD_READY_TRUE	New LSAD samples are ready	0x1

12.9.4.10 LSAD_PRE_SEL_INPUT

Bit Field	Read/Write	Field Name	Description
14:12	RW	LSAD_PRE_SEL_IN3	LSAD input pre-selection
10:8	RW	LSAD_PRE_SEL_IN2	LSAD input pre-selection
6:4	RW	LSAD_PRE_SEL_IN1	LSAD input pre-selection
2:0	RW	LSAD_PRE_SEL_IN0	LSAD input pre-selection

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
14:12	LSAD_PRE_SEL_IN3	IN3_PRE_SEL_GPIO1	GPIO1 pre-selected for lsad input 3	0x0*
		IN3_PRE_SEL_GPIO3	GPIO3 pre-selected for lsad input 3	0x1
		IN3_PRE_SEL_GPIO5	GPIO5 pre-selected for lsad input 3	0x2
		IN3_PRE_SEL_GPIO7	GPIO7 pre-selected for lsad input 3	0x3
		IN3_PRE_SEL_GPIO9	GPIO9 pre-selected for lsad input 3	0x4
		IN3_PRE_SEL_GPIO11	GPIO11 pre-selected for lsad input 3	0x5
		IN3_PRE_SEL_GPIO13	GPIO13 pre-selected for lsad input 3	0x6
		IN3_PRE_SEL_GPIO15	GPIO15 pre-selected for lsad input 3	0x7
10:8	LSAD_PRE_SEL_IN2	IN2_PRE_SEL_GPIO0	GPIO0 pre-selected for lsad input 2	0x0*
		IN2_PRE_SEL_GPIO2	GPIO2 pre-selected for lsad input 2	0x1
		IN2_PRE_SEL_GPIO4	GPIO4 pre-selected for lsad input 2	0x2
		IN2_PRE_SEL_GPIO6	GPIO6 pre-selected for lsad input 2	0x3
		IN2_PRE_SEL_GPIO8	GPIO8 pre-selected for lsad input 2	0x4
		IN2_PRE_SEL_GPIO10	GPIO10 pre-selected for lsad input 2	0x5
		IN2_PRE_SEL_GPIO12	GPIO12 pre-selected for lsad input 2	0x6
		IN2_PRE_SEL_GPIO14	GPIO14 pre-selected for lsad input 2	0x7
6:4	LSAD_PRE_SEL_IN1	IN1_PRE_SEL_GPIO1	GPIO1 pre-selected for lsad input 1	0x0*
		IN1_PRE_SEL_GPIO3	GPIO3 pre-selected for lsad input 1	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		IN1_PRE_SEL_GPIO5	GPIO5 pre-selected for lsad input 1	0x2
		IN1_PRE_SEL_GPIO7	GPIO7 pre-selected for lsad input 1	0x3
		IN1_PRE_SEL_GPIO9	GPIO9 pre-selected for lsad input 1	0x4
		IN1_PRE_SEL_GPIO11	GPIO11 pre-selected for lsad input 1	0x5
		IN1_PRE_SEL_GPIO13	GPIO13 pre-selected for lsad input 1	0x6
		IN1_PRE_SEL_GPIO15	GPIO15 pre-selected for lsad input 1	0x7
2:0	LSAD_PRE_SEL_IN0	IN0_PRE_SEL_GPIO0	GPIO0 pre-selected for lsad input 0	0x0*
		IN0_PRE_SEL_GPIO2	GPIO2 pre-selected for lsad input 0	0x1
		IN0_PRE_SEL_GPIO4	GPIO4 pre-selected for lsad input 0	0x2
		IN0_PRE_SEL_GPIO6	GPIO6 pre-selected for lsad input 0	0x3
		IN0_PRE_SEL_GPIO8	GPIO8 pre-selected for lsad input 0	0x4
		IN0_PRE_SEL_GPIO10	GPIO10 pre-selected for lsad input 0	0x5
		IN0_PRE_SEL_GPIO12	GPIO12 pre-selected for lsad input 0	0x6
		IN0_PRE_SEL_GPIO14	GPIO14 pre-selected for lsad input 0	0x7

12.9.4.11 LSAD_DUTY

Bit Field	Read/Write	Field Name	Description
15:14	RW	CH7_DUTY_CFG	Input to channel 7 duty config
13:12	RW	CH6_DUTY_CFG	Input to channel 6 duty config
11:10	RW	CH5_DUTY_CFG	Input to channel 5 duty config
9:8	RW	CH4_DUTY_CFG	Input to channel 4 duty config
7:6	RW	CH3_DUTY_CFG	Input to channel 3 duty config
5:4	RW	CH2_DUTY_CFG	Input to channel 2 duty config
3:2	RW	CH1_DUTY_CFG	Input to channel 1 duty config
1:0	RW	CH0_DUTY_CFG	Input to channel 0 duty config

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
15:14	CH7_DUTY_CFG	INPUT_TO_CH7_PERMANENT	Input always connected to channel	0x0*
		INPUT_TO_CH7_DUTY	Input only connected to the channel during this conversion	0x1
		INPUT_TO_CH7_RESERVED	reserved - same as 0x0	0x2

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		INPUT_TO_CH7_PRE_CONV	Input connected to the channel during this conversion and the previous conversion	0x3
13:12	CH6_DUTY_CFG	INPUT_TO_CH6_PERMANENT	Input always connected to channel	0x0*
		INPUT_TO_CH6_DUTY	Input only connected to the channel during this conversion	0x1
		INPUT_TO_CH6_RESERVED	reserved - same as 0x0	0x2
		INPUT_TO_CH6_PRE_CONV	Input connected to the channel during this conversion and the previous conversion	0x3
11:10	CH5_DUTY_CFG	INPUT_TO_CH5_PERMANENT	Input always connected to channel	0x0*
		INPUT_TO_CH5_DUTY	Input only connected to the channel during this conversion	0x1
		INPUT_TO_CH5_RESERVED	reserved - same as 0x0	0x2
		INPUT_TO_CH5_PRE_CONV	Input connected to the channel during this conversion and the previous conversion	0x3
9:8	CH4_DUTY_CFG	INPUT_TO_CH4_PERMANENT	Input always connected to channel	0x0*
		INPUT_TO_CH4_DUTY	Input only connected to the channel during this conversion	0x1
		INPUT_TO_CH4_RESEREVED	reserved - same as 0x0	0x2
		INPUT_TO_CH4_PRE_CONV	Input connected to the channel during this conversion and the previous conversion	0x3
7:6	CH3_DUTY_CFG	INPUT_TO_CH3_PERMANENT	Input always connected to channel	0x0*
		INPUT_TO_CH3_DUTY	Input only connected to the channel during this conversion	0x1
		INPUT_TO_CH3_RESERVED	reserved - same as 0x0	0x2
		INPUT_TO_CH3_PRE_CONV	Input connected to the channel during this conversion and the previous conversion	0x3
5:4	CH2_DUTY_CFG	INPUT_TO_CH2_PERMANENT	Input always connected to channel	0x0*
		INPUT_TO_CH2_DUTY	Input only connected to the channel during this conversion	0x1
		INPUT_TO_CH2_RESERVED	reserved - same as 0x0	0x2
		INPUT_TO_CH2_PRE_CONV	Input connected to the channel during	0x3

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
			this conversion and the previous conversion	
3:2	CH1_DUTY_CFG	INPUT_TO_CH1_PERMANENT	Input always connected to channel	0x0*
		INPUT_TO_CH1_DUTY	Input only connected to the channel during this conversion	0x1
		INPUT_TO_CH1_RESERVED	reserved - same as 0x0	0x2
		INPUT_TO_CH1_PRE_CONV	Input connected to the channel during this conversion and the previous conversion	0x3
1:0	CH0_DUTY_CFG	INPUT_TO_CH0_PERMANENT	Input always connected to channel	0x0*
		INPUT_TO_CH0_DUTY	Input only connected to the channel during this conversion	0x1
		INPUT_TO_CH0_RESERVED	reserved - same as 0x0	0x2
		INPUT_TO_CH0_PRE_CONV	Input connected to the channel during this conversion and the previous conversion	0x3

12.9.4.12 LSAD_ID_NUM

Bit Field	Read/Write	Field Name	Description
15:8	R	MAJOR_REVISION	LSAD Major Revision number
7:0	R	MINOR_REVISION	LSAD Minor Revision number

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
15:8	MAJOR_REVISION	LSAD_MAJOR_REVISION	LSAD revision 1.0	0x1*
7:0	MINOR_REVISION	LSAD_MINOR_REVISION	LSAD revision 1.0	0x0*

12.9.5 SDAC Registers

Register Name	Register Description	Address
ACS_SDAC_CFG	SDAC configuration register	0x40001B34

12.9.5.1 ACS_SDAC_CFG

Bit Field	Read/Write	Field Name	Description
2	RW	SDAC_TO_GPIO7	GPIO7 drive by SDAC
1	RW	SDAC_GAIN	Sets the SDAC buffer gain
0	RW	SDAC_EN	Enable signal for SDAC buffer

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
2	SDAC_TO_GPIO7	GPIO7_NOT_DRIVE_BY_SDAC	GPIO[7] not driven by SDAC	0x0*
		GPIO7_DRIVE_BY_SDAC	GPIO[7] drives by SDAC	0x1
1	SDAC_GAIN	SDAC_GAIN_X1	Gain = 1	0x0*
		SDAC_GAIN_X2	Gain = 2	0x1
0	SDAC_EN	SDAC_DISABLE	SDAC disabled	0x0*
		SDAC_ENABLE	SDAC enabled	0x1

12.9.6 Current Source Registers

Register Name	Register Description	Address
ACS_TEMP_CURR_CFG	Source Current Configuration Register	0x40001B84

12.9.6.1 ACS_TEMP_CURR_CFG

Bit Field	Read/Write	Field Name	Description
19:16	RW	CURRENT_VALUE	Temperature current value
13:8	RW	CURRENT_TRIM	Temperature current trimming
7:4	RW	GPIO_IN_USE	GPIO from which the duty cycle is used
1	RW	DUTY_CURR	Duty cycling current enable
0	RW	ENABLE	Temperature current enable

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
19:16	CURRENT_VALUE	TEMP_CURR_1UA	Temperature current set to 1 uA	0x0
		TEMP_CURR_10UA	Temperature current set to 10 uA	0x9*
		TEMP_CURR_16UA	Temperature current set to 16 uA	0xF
13:8	CURRENT_TRIM	TEMP_CURR_TRIM_M32	-10% trimming	0x0*
		TEMP_CURR_TRIM_0	Default trimming	0x20

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		TEMP_CURR_TRIM_P31	+10% trimming	0x3F
7:4	GPIO_IN_USE	TEMP_CURR_GPIO0	Duty cycle of GPIO0 is used	0x0*
		TEMP_CURR_GPIO1	Duty cycle of GPIO1 is used	0x1
		TEMP_CURR_GPIO2	Duty cycle of GPIO2 is used	0x2
		TEMP_CURR_GPIO3	Duty cycle of GPIO3 is used	0x3
		TEMP_CURR_GPIO4	Duty cycle of GPIO4 is used	0x4
		TEMP_CURR_GPIO5	Duty cycle of GPIO5 is used	0x5
		TEMP_CURR_GPIO6	Duty cycle of GPIO6 is used	0x6
		TEMP_CURR_GPIO7	Duty cycle of GPIO7 is used	0x7
		TEMP_CURR_GPIO8	Duty cycle of GPIO8 is used	0x8
		TEMP_CURR_GPIO9	Duty cycle of GPIO9 is used	0x9
		TEMP_CURR_GPIO10	Duty cycle of GPIO10 is used	0xA
		TEMP_CURR_GPIO11	Duty cycle of GPIO11 is used	0xB
		TEMP_CURR_GPIO12	Duty cycle of GPIO12 is used	0xC
		TEMP_CURR_GPIO13	Duty cycle of GPIO13 is used	0xD
		TEMP_CURR_GPIO14	Duty cycle of GPIO14 is used	0xE
		TEMP_CURR_GPIO15	Duty cycle of GPIO15 is used	0xF
1	DUTY_CURR	LSAD_CURR_NORMAL	Normal mode: current enable only controlled by CURR_ENABLE	0x0*
		LSAD_CURR_DUTY	Duty cycling current enable (enabled only during LSAD conversion)	0x1
0	ENABLE	TEMP_CURR_DISABLE	Disable the temperature current	0x0*
		TEMP_CURR_ENABLE	Enable the temperature current	0x1

12.9.7 Internal Temperature Sensor Registers

Register Name	Register Description	Address
ACS_TEMP_SENSOR_CFG	Temperature Sensor Configuration Register	0x40001B80

12.9.7.1 ACS_TEMP_SENSOR_CFG

Bit Field	Read/Write	Field Name	Description
1	RW	DUTY_TEMP_SENS	Duty cycling temperature sensor
0	RW	ENABLE	Internal temperature sensor enable

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
1	DUTY_TEMP_SENS	LSAD_TEMP_SENS_NORMAL	Normal mode: temperature sensor enable only controlled by TEMP_SENS_ENABLE	0x0*
		LSAD_TEMP_SENS_DUTY	Duty cycling temperature sensor (enabled only during LSAD conversion)	0x1
0	ENABLE	TEMP_SENS_DISABLE	Disable the internal Temperature Sensor	0x0*
		TEMP_SENS_ENABLE	Enable the internal Temperature Sensor	0x1

12.9.8 ULP Registers

Register Name	Register Description	Address
SENSOR_CFG	Sensor Configuration	0x40001D18
SENSOR_CTRL	Sensor Control	0x40001D1C
SENSOR_FIFO_CFG	Sensor ADC data FIFO configuration	0x40001D20
SENSOR_PROCESSING	Sensor ADC wake-up threshold	0x40001D24
SENSOR_FIFO_DATA	Sensor ADC output data FIFO	0x40001D30-0x40001D6C

12.9.8.1 SENSOR_CFG

Bit Field	Read/Write	Field Name	Description
24	RW	CLK_SEL	Clock source selection
20	RW	SRC_SEL	Sample source selection
13	RW	DLY_EN	Delay state enable
12	RW	DLY_DIV_EN	Delay divider selection
9:0	RW	DLY	Absolute Value of main counter to trigger the change of delay state

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
24	CLK_SEL	SENSOR_CLK_STANDBY	Standby clock is used as sensor clock	0x0
		SENSOR_CLK_SYSCLK_DIV	Divided SYSCLK is used as sensor clock	0x1*
20	SRC_SEL	SAR_ADC_SELECTED	SAR ADC sample selected	0x0*
		PC_SELECTED	Pulse Counter sample selected	0x1
13	DLY_EN	DLY_NOT_USED	Delay state is not used	0x0
		DLY_USED	Delay state is used	0x1*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
12	DLY_DIV_EN	DLY_DIV_DISABLED	Delay runs at sensor clock	0x0*
		DLY_DIV_ENABLED	Delay runs at sensor clock divided by 32	0x1
9:0	DLY	DLY_1	Number of SENSOR_CLK periods for "Delay": 1	0x0*
		DLY_2	Number of SENSOR_CLK periods for "Delay": 2	0x1
		DLY_256	Number of SENSOR_CLK periods for "Delay": 256	0xFF
		DLY_1024	Number of SENSOR_CLK periods for "Delay": 1024	0x3FF

12.9.8.2 SENSOR_CTRL

Bit Field	Read/Write	Field Name	Description
31	W	RESET	Reset the Sensor interface (SAR and Pulse Counter)
12	R	STATE	Sensor state
8	R	STATUS	Sensor status
0	RW	ENABLE	Sensor

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31	RESET	SENSOR_RESET	The sensor timer counter, the SAR-ADC analog, the sensor enable and the pulse counter are reset.	0x1
12	STATE	DELAY_STATUS	Sensor in delay state or in idle state	0x0*
		COUNT_STATUS	Sensor in count state	0x1
8	STATUS	SENSOR_IDLE	Sensor state machine in idle mode	0x0*
		SENSOR_RUNNING	Sensor state machine in active mode	0x1
0	ENABLE	SENSOR_DISABLE	Sensor disables	0x0*
		SENSOR_ENABLE	Sensor enables	0x1

12.9.8.3 SENSOR_FIFO_CFG

Bit Field	Read/Write	Field Name	Description
12:8	R	FIFO_LEVEL	Number of samples stored in FIFO
6	RW	FIFO_RX_INT_EN	Enable / disable the interrupt when FIFO is full
5	RW	FIFO_RX_DMA_EN	Enable / disable the DMA trigger when FIFIO is full
4	RW	FIFO_STORE_EN	Enable the storage of sample on the FIFO
3:0	RW	FIFO_SIZE	Number of samples stored before waking up the core

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
12:8	FIFO_LEVEL	SENSOR_FIFO_EMPTY	FIFO data level is 0	0x0*
		SENSOR_FIFO_FULL	FIFO data level is 16	0x10
6	FIFO_RX_INT_EN	FIFO_RX_INT_DISABLED	No RX interrupt	0x0*
		FIFO_RX_INT_ENABLED	RX interrupt when FIFO is full	0x1
5	FIFO_RX_DMA_EN	FIFO_RX_DMA_DISABLED	No RX DMA triggered	0x0*
		FIFO_RX_DMA_ENABLED	RX DMA triggered when FIFO is full	0x1
4	FIFO_STORE_EN	SENSOR_FIFO_STORE_ DISABLED	Samples are not stored in the FIFO	0x0
		SENSOR_FIFO_STORE_ENABLED	Samples are stored in the FIFO	0x1*
3:0	FIFO_SIZE	SENSOR_FIFO_SIZE1	FIFO size 1; core wakes up after first sample arrives	0x0*
		SENSOR_FIFO_SIZE2	FIFO size 2	0x1
		SENSOR_FIFO_SIZE3	FIFO size 3	0x2
		SENSOR_FIFO_SIZE4	FIFO size 4	0x3
		SENSOR_FIFO_SIZE5	FIFO size 5	0x4
		SENSOR_FIFO_SIZE6	FIFO size 6	0x5
		SENSOR_FIFO_SIZE7	FIFO size 7	0x6
		SENSOR_FIFO_SIZE8	FIFO size 8	0x7
		SENSOR_FIFO_SIZE9	FIFO size 9	0x8
		SENSOR_FIFO_SIZE10	FIFO size 10	0x9
		SENSOR_FIFO_SIZE11	FIFO size 11	0xA
		SENSOR_FIFO_SIZE12	FIFO size 12	0xB
		SENSOR_FIFO_SIZE13	FIFO size 13	0xC

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		SENSOR_FIFO_SIZE14	FIFO size 14	0xD
		SENSOR_FIFO_SIZE15	FIFO size 15	0xE
		SENSOR_FIFO_SIZE16	FIFO size 16	0xF

12.9.8.4 SENSOR_PROCESSING

Bit Field	Read/Write	Field Name	Description
24	RW	SUM_EN	Summation enable
19:16	RW	NBR_SAMPLES	Number of samples used for wakeup in sensor detect mode

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
24	SUM_EN	SENSOR_SUMMATION_DISABLED	Summation is disabled	0x0*
		SENSOR_SUMMATION_ENABLED	Summation is enabled	0x1
19:16	NBR_SAMPLES	SENSOR_NBR_SAMPLES_1	1 sample used	0x0*
		SENSOR_NBR_SAMPLES_2	2 samples used	0x1
		SENSOR_NBR_SAMPLES_16	16 samples used	0xF

12.9.8.5 SENSOR_FIFO_DATA

Bit Field	Read/Write	Field Name	Description
31:0	R	VALUE	Sensor FIFO output data (signed)

CHAPTER 13

Peripherals

RSL15's functionality is compatible with numerous peripherals to enhance user applications. These peripherals are described in this group of topics.

13.1 Activity Counters

The activity counters are used for accurately measuring the execution time of processes between user-defined start and stop points. This, in turn, helps firmware developers to analyze how actively the Arm Cortex-M33 processor and the flash memory are being used by a user application. This information helps when estimating and optimizing 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 of cycles on the Arm Cortex-M33 processor is recorded in the SYSCTRL_CPU_CNT register. This counter only counts when the Arm Cortex-M33 processor is running, and does not increment when the core is sleeping or otherwise inactive.
- The number of CBus read instructions is recorded in the SYSCTRL_CBUS_CNT register.
- 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 writing 1 to the SYSCTRL_CNT_CTRL_CNT_CLEAR bit
- Started by writing 1 to the SYSCTRL_CNT_CTRL_CNT_START bit
- Stopped by writing 1 to the SYSCTRL_CNT_CTRL_CNT_STOP bit. The write to SYSCTRL_CNT_CTRL_CNT_STOP is ignored when a 1 is written to the SYSCTRL_CNT_CTRL_CNT_START bit at the same time.

NOTE: The SYSCTRL_SYSCLK_CNT register is used by the program ROM during the boot to store error flags, so we recommend that you clear it before use.

When the counters are tracking activity, the SYSCTRL_CNT_CTRL_CNT_STATUS bit in the SYSCTRL_CNT_CTRL register is set to CNT_RUNNING.

When the SYSCTRL_SYSCLK_CNT register is saturated (0xFFFF_FFFF), the SYSCTRL_CNT_CTRL_CNT_STATUS bit of the SYSCTRL_CNT_CTRL register is reset and all the counters are stopped automatically.

For registers, see Section 13.9.1 "Activity Counters Registers" on page 711.

13.2 Asynchronous Clock Counter

The asynchronous clock counter (ASCC) can be used to measure the timing of a clock through the internal STANDBYCLK, or the timing of a clock in a connected external device. In addition, the ASCC can be used to measure the frequency of the internal 32 kHz RC oscillator using the 48 MHz crystal oscillator.

The following ASCC uses are supported:

- Generation of control information to synchronize the RSL15 system's sampling frequency with the signal of an external connected device
- Measurement of the internal STANDBYCLK oscillator relative to the SYSCLK frequency through a GPIO pad
- Measurement of the internal 32 kHz RC oscillator

IMPORTANT: The asynchronous clock being measured is sourced from a GPIO or from STANDBYCLK, as specified by the GPIO configuration. For more information on GPIO configuration for use as the asynchronous clock source, see Section 10.2 “Functional Configuration” on page 512.

The timing diagram in the "Asynchronous Clock Counters Timing Diagram" figure (Figure 69) highlights the measurement registers in the Asynchronous Clock module.

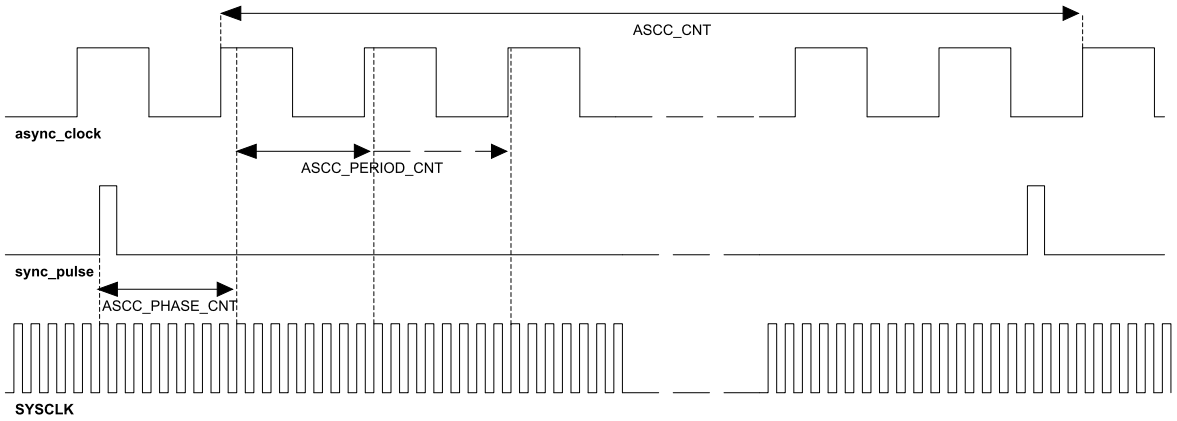


Figure 69. Asynchronous Clock Counters Timing Diagram

The measurement registers are defined as follows:

- The ASCC_CNT register holds the integer number of cycles of the asynchronous clock (saturated to 0x0FFF) being measured between consecutive synchronization pulses.
- The ASCC_PERIOD_CNT register contains the period count of the asynchronous clock (saturated to 0x0FFF) being measured in terms of SYSCLK cycles, which is typically derived from the 48 MHz crystal oscillator (See Section 7.4.1 “System Clock (SYSCLK)” on page 390 for more information.) The period can be measured over 1-16 asynchronous clock cycles. The number of asynchronous clock cycles measured is controlled by the ASCC_CFG_PERIODS_CFG bit field in the ASCC_CFG register. By using the value stored in the ASCC_PERIOD_CNT register, the period can be calculated as:

$$period = \frac{ASCC\_PERIOD\_CNT}{(ASCC\_CFG\_PERIODS+1)} \times period_{SYSCLK}$$

NOTE: The accuracy of the period measurement is improved by increasing the ASCC_CFG_PERIODS_CFG value and/or the SYSCLK frequency.

This period counter is supported by:

- The ASCC_CTRL_PERIOD_CNT_START bit from the ASCC_CTRL register, which clears and starts the period counter when a rising edge of the asynchronous clock is detected

- The ASCC_CTRL_PERIOD_CNT_STOP bit from the ASCC_CTRL register, which stops the period counter mechanism manually
- The ASCC_CTRL_PERIOD_STATUS bit, which indicates whether the period counter mechanism is active or idle
- The ASCC_PERIOD interrupt, which is triggered when the asynchronous clock period counter finishes counting the defined number of asynchronous clock periods

The ASCC_PHASE_CNT register measures the time from the synchronization pulse to the first detected rising edge of the asynchronous clock, in terms of SYSCLK cycles. This measurement is used to improve the resolution of the calculated number of asynchronous clock cycles per frame. This counter is supported by:

- The ASCC_CTRL_PHASE_CNT_START bit (in the ASCC_CTRL register), which clears and starts the phase counter the next time a synchronization pulse is generated
- The same register's ASCC_CTRL_PHASE_CNT_START_NO_WAIT bit, which clears and starts the phase counter immediately
- The same register's ASCC_CTRL_PHASE_CNT_STOP bit, which stops the phase counter mechanism manually
- The same register's ASCC_CTRL_PHASE_STATUS bit, which indicates whether the phase counter mechanism is currently active or idle
- The ASCC_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 asynchronous clock (saturated to 0xFFFF), at which point the counter is stopped automatically
- The ASCC_PHASE interrupt, which is triggered when the phase counter is running and a rising edge occurs on the asynchronous clock, at which point the counter mechanism is stopped automatically

Using the values obtained from the ASCC_CNT and ASCC_PHASE_CNT registers plus the previous period and phase count values, you can calculate the number of asynchronous clock cycles between the previous (i-1) and the current (i) synchronization pulses as follows:

$$\text{asynchronous clock cycles} = ASCC\_CNT[i] + \frac{ASCC\_PHASE\_CNT[i-1] - ASCC\_PHASE\_CNT[i]}{ASCC\_PERIOD\_CNT[i-1]/(ASCC\_PERIODS\_CFG+1)}$$

The recommended process for using the measurement registers is as follows:

- Before a synchronization pulse is expected to occur, start the phase counter mechanism.
- When the ASCC_PHASE interrupt occurs:
  a. Save the ASCC_PHASE_CNT counter value from the previous frame.
  b. Record the ASCC_PHASE_CNT counter value.
  c. Record the value in the ASCC_CNT register.
  d. Reset the ASCC_CNT register/counter.
  e. Start the period counter.
- When the ASCC_PERIOD interrupt occurs:
  a. Record the ASCC_PERIOD_CNT counter value.
  b. Calculate the number of SYSCLK cycles per asynchronous clock period.

For registers, see Section 13.9.2 "Asynchronous Clock Counter Registers" on page 712.

13.3 Cyclic Redundancy Check (CRC) Generator

The RSL15 system includes 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 flash memory words is 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_CFG_CRC_TYPE bit from the CRC_CFG register to the CRC_CCITT bit setting. The parameters associated with the CRC-CCITT algorithm implementation are provided in the "CRC_CCITT Algorithm Parameters" table (Table 33).

Table 33. 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_CFG_CRC_TYPE bit from the CRC_CFG register to the CRC_32 bit setting. The parameters associated with the CRC-32 algorithm implementation are provided in the "CRC_CCITT Algorithm Parameters" table (Table 33).

Table 34. 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 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_CFG_BIT_ORDER bit from the CRC_CFG register.
- To use non-standard CRC ordering of the final result, set the CRC_CFG_FINAL_CRC_REVERSE bit from the CRC_CFG register.

- To use non-standard CRC XOR of the final result, set the CRC_CFG_FINAL_CRC_XOR bit from the CRC_CFG 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_CFG_BYTE_ORDER bit from the CRC_CFG 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.

For registers, see Section 13.9.3 “CRC Registers” on page 715.

13.4 Direct Memory Access (DMA) Controller

13.4.1 Block Overview

The direct memory access controller (DMA) module allows background transfers between components (referred to in this section as peripherals) on the peripheral bus, and memories, without any processor intervention. This allows the processors to be used for other computational needs while enabling high speed sustained transfers to and from the peripherals/memories.

The DMA has 4 independent configurable channels. Each channel can be configured for one of four modes:

- Data transferred from peripheral-to-memory (PM)
- Data transferred from memory-to-peripheral (MP)
- Data transferred between peripherals (PP)
- Data transferred between memory locations (MM)

The "DMA Overview" figure (Figure 70) shows an overview of the module.

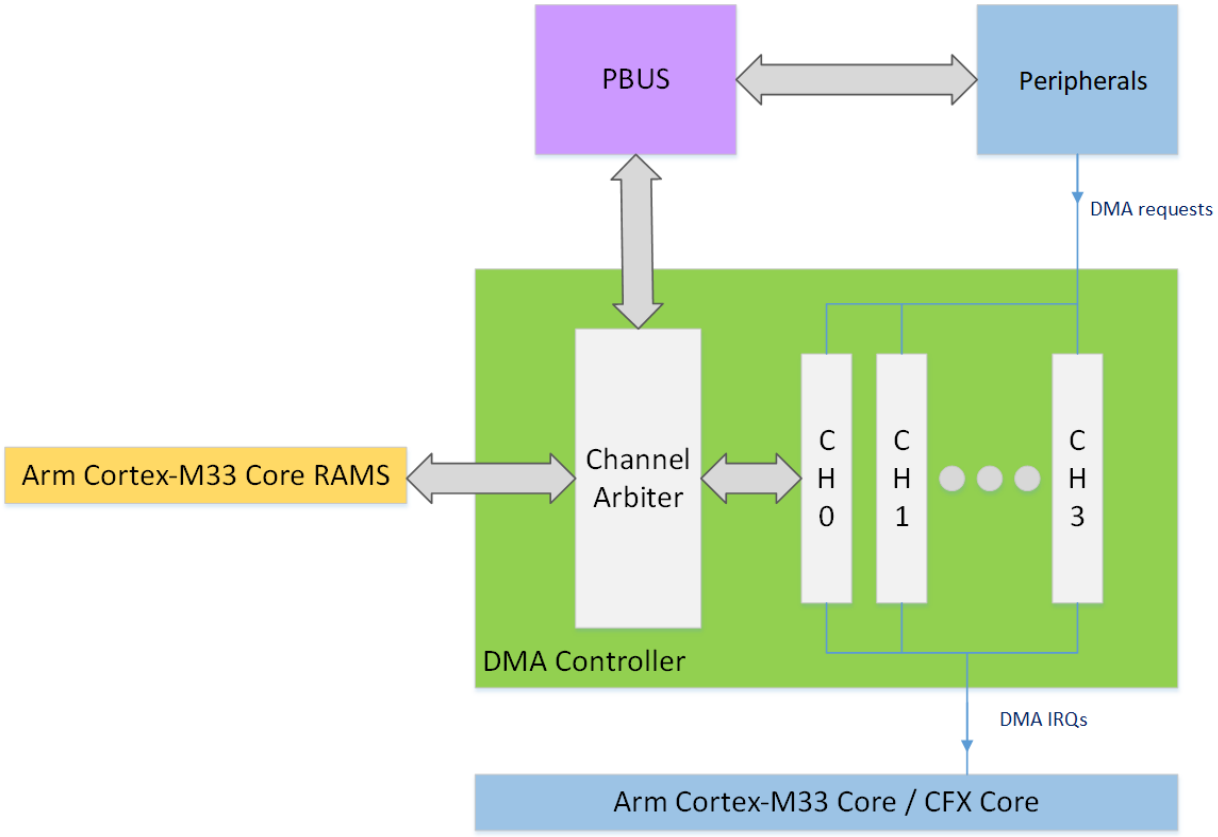


Figure 70. DMA Overview

There is a separate interrupt line available to the Arm Cortex-M33 core for each of the DMA channels. Each DMA channel can be configured to assert its interrupt line for several independent conditions, such as when a transfer, or a specified portion of it, is complete.

13.4.2 Functional Description

13.4.2.1 DMA to Peripherals Interface

The DMA supports several types of data transfers between the memories of the Arm Cortex-M33 processor 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

Each peripheral that supports DMA transfers is connected to the DMA controller through one or more DMA request signals. In most cases, a peripheral must be configured explicitly to operate in DMA mode to enable requests as required. The DMA supports 16 request lines from various peripherals included in RSL15, and is connected independently to each channel.

The interfaces and peripherals that are mapped onto the peripheral bus, and that are valid data sources and destinations for DMA data transfers, are listed in the "DMA Request Lines and their Associated Peripherals" table (Table 35) along with their associated request lines. DMA trigger lines and sources are shown in the "DMA Trigger Lines and Sources" table (Table 36). For sources or destinations that do not need to wait for data (typically memories), no request timing is required so the transfer uses an always on (`DMA_SRC_ALWAYS_ON`, `DMA_DEST_ALWAYS_ON`) configuration. To select the desired source and destination, set the `DMA_CFG0_SRC_SELECT` and `DMA_CFG0_DEST_SELECT` bit-fields from the `DMA_CFG0` register.

Table 35. DMA Request Lines and their Associated Peripherals

DMA Request Line	Peripheral	Source	Destination
0	Always_on	DMA_SRC_ALWAYS_ON	DMA_DEST_ALWAYS_ON
1	SPI0	DMA_SRC_SPI0	DMA_DEST_SPI0
2	SPI1	DMA_SRC_SPI1	DMA_DEST_SPI1
3	I2C0	DMA_SRC_I2C0	DMA_DEST_I2C0
4	I2C1	DMA_SRC_I2C1	DMA_DEST_I2C1
5	UART0	DMA_SRC_UART0	DMA_DEST_UART0
6	PCM0	DMA_SRC_PCM0	DMA_DEST_PCM0
7	TOF	DMA_SRC_TOF	DMA_DEST_TOF
8	RF_IQ_READY	DMA_SRC_RF_IQ_READY	DMA_DEST_REQ_8
9	R_ Phase_ ADC	DMA_SRC_RF_PHASE_ADC_READY	DMA_DEST_REQ_9
10		Reserved	
11		Reserved	
12		Reserved	
13		Reserved	
14		Reserved	
15		Reserved	

Table 36. DMA Trigger Lines and Sources

DMA Trigger Line	Source
0	DMA Channel 0 completed
1	DMA Channel 1 completed
2	DMA Channel 2 completed
3	DMA Channel 3 completed
4	Sensor interface FIFO full
5	LIN interface RX completed
6	SAR interface RX completed

Table 36. DMA Trigger Lines and Sources (Continued)

DMA Trigger Line	Source
7	Timer 0 interrupt
8	Timer 1 interrupt
9	Timer 2 interrupt
10	Timer 3 interrupt
11	Reserved
12	Reserved
13	Reserved
14	Reserved
15	Reserved

Each source interface or peripheral, when configured for DMA operation, asserts its DMA request signal when data can be read from the source. This signal is cleared automatically when a data value is read from the source using the peripheral bus.

CAUTION: When the DMA interface is used to control transfers that use an interface or peripheral, any access 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 a processor access, the underlying DMA transfer becomes corrupted.

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.

13.4.2.2 Configuration and Status

The "DMA Module Specification" table (Table 3) describes DMA supported configurations and specifications:

Table 37. DMA Module Specification

Item	Specification
Number of DMA Channels	4
Priority Levels	4
Maximum Transfer Length	65536
Counter Interrupt Trigger Range	1 – 65535
Configurable Modes	Increment or Circular

DMA Module Specification (Continued)

Item	Specification
Types of Transfers	Memory to Peripheral (MP) Peripheral to Memory (PM) Memory to Memory (MM) Peripheral to Peripheral (PP)
Interrupt Lines	8 (one per channel)
Interrupt Source per Channel	3: Transfer length counter, Complete, Word counter
Word Sizes	4-bit, 8-bit, 16-bit, and 32-bit

Each DMA channel can be configured independently through its associated memory mapped registers. Each DMA channel has one bit field, the DMA*_CTRL_MODE field of the DMA_CTRL register, which is used to enable one of the seven different modes in which the channel can be active (see the "DMA Channel Modes of Operation" table (Table 4), below). The DMA channel can be deactivated at any time by setting the DMA_CTRL_MODE_ENABLE bit to DMA_DISABLE. If the DMA channel is in the midst of an operation when it is disabled, the current read or write transaction is completed. All pending bus requests are subsequently aborted when the channel is disabled.

Table 38. DMA Channel Modes of Operation

MODE_ ENABLE Bit Value	Constant	Remarks
0	DMA_DISABLE	Disable the DMA channel (channel waits on current READ or WRITE access to finish before the active bit is cleared)
1	DMA_ENABLE	Immediately enable the DMA channel, and when the transfer is completed, disable the DMA channel
2	DMA_ENABLE_WRAP	Immediately enable the DMA channel, and when the transfer is completed, wrap addresses and disable the DMA channel
3	DMA_ENABLE_WRAP_RESTART	Immediately enable the DMA channel, and when the transfer is completed, wrap addresses and restart the transfer
4	DMA_TRIGGER	When the source trigger occurs, enable the DMA channel; when the transfer is completed, disable the DMA channel
5	DMA_TRIGGER_WRAP	When the source trigger occurs, enable the DMA channel; when the transfer is completed, wrap addresses and disable the DMA channel
6	DMA_TRIGGER_WRAP_RESTART	When the source trigger occurs, enable the DMA channel; when the transfer is completed, wrap addresses and restart the DMA channel
7	DMA_TRIGGER_WRAP_TRIGGER_ RESTART	When the source trigger occurs, enable the DMA channel; when the transfer is completed, wrap addresses, and upon next trigger event, restart the DMA channel

The DMA interface has eight modes of operation. These modes can be roughly divided into two groups: the manual modes (DMA_ENABLE, DMA_ENABLE_WRAP, and DMA_ENABLE_WRAP_RESTART) and the trigger modes (DMA_TRIGGER, DMA_TRIGGER_WRAP, DMA_TRIGGER_WRAP_RESTART, and DMA_TRIGGER_WRAP_TRIGGER_RESTART). The

two mode groups differ in how a transfer is started in each group. In a manual (non-trigger) mode, the transfer start is controlled by the user application. One such application would be a memory-to-memory buffer copy operation or a memory-to-peripheral transfer to a communication interface or a peripheral. In a trigger mode, a transfer is started by a trigger event configured through the DEST_SELECT and SRC_SELECT bits in the DMA[0:3]_CFG0 register. Typical applications of this mode are peripheral-to-memory transfers.

The DMA_ENABLE and DMA_TRIGGER modes are one-shot modes that are used when the user application controls the start and address of the DMA channel. The application has to set up the addresses after each transfer.

The DMA_ENABLE_WRAP and DMA_TRIGGER_WRAP are also one-shot modes. In these modes, the user application needs to set up the addresses for the first transfer. After completing a transfer, the addresses are wrapped by the hardware and left ready for the next transfer. This is useful for implementing circular buffers.

In the DMA_ENABLE_WRAP_RESTART and DMA_TRIGGER_WRAP_RESTART modes, the DMA interface is run in a continuous mode. It keeps running until manually stopped by the user application. The addresses are automatically wrapped by the hardware at the end of each transfer.

The DMA_TRIGGER_WRAP_TRIGGER_RESTART mode differs from the two previous modes in that the interface is not continuously run. Each transfer is automatically started by a trigger event without user intervention, and the addresses are wrapped by the hardware.

Trigger modes are further explained in Section 13.4 “Direct Memory Access (DMA) Controller” on page 690.

Other DMA channel configurations include:

Source and Destination Address Configuration

The DMA_SRC_ADDR registers define the base address of the source data for a DMA channel. The DMA_DEST_ADDR registers define the base address of the destination data for a DMA channel. The DMA supports byte addressing for source and destination addresses. Smaller word sizes are supported indirectly by selecting the source and destination word sizes, and through packing and unpacking. The DMA calculates the next source address and the next destination address by using the current counter value.

These addresses need to be word-aligned, a requirement for the 32-bit accesses that this block performs. If a non-aligned address is configured, the DMA ignores the two LSBs, making the transfer word-aligned.

Each DMA channel can be configured for two address modes: linear and circular. These address modes are selected while defining the mode of operation in the DMA control registers.

- In linear address mode, the DMA operation occurs and then the DMA channel is automatically disabled once the transfer is complete. The source and destination addresses are not reset to predefined addresses.
- In wrap restart mode (circular address mode), the DMA operation occurs and then the DMA channel configuration is reset to the initial configuration once the transfer is completed. This configuration includes current source address, current destination address, and length of transfer. The DMA channel remains enabled, continuing to run until explicitly stopped by the firmware when it sets the appropriate enabling bit to a disabled status.
- In wrap linear mode, the addresses are wrapped at the end before the DMA channel is disabled. This sets up the memory block for the next transfer.

- Each DMA channel has independent increment and decrement controls: between -8 and +7, with a step size of 1, for both source and destination addresses. The required step size can be configured using the DMA_CFG0_SRC_ADDR_STEP and DMA_CFG0_DEST_ADDR_STEP bit fields in the DMA_CFG0 registers, for the source and destination addresses respectively. When the addressing step size is set to DMA_DEST_ADDR_STATIC, the address is neither incremented nor decremented after each word is transferred, i.e., it remains static. The address step size is applied to the address after each word is transferred. For example, in Memory-to-Peripheral operations, typically the source has the increment enabled, while the destination has the increment disabled.

NOTE: The source and destination address registers are not modified during a transfer; the increment setting only affects the calculation of the next address. When disabled, the next address is always equal to the base address.

Source and Destination Data

The source and destination data can be configured to be interpreted as being packed into a variety of different word sizes and endian ordering. For more information, see Section 13.4.2.6 “Packing and Unpacking” on page 699.

Transfer Length and Counter Interrupt 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_CFG1_TRANSFER_LENGTH bit field in the DMA_CFG1 registers. When the DMA channel completes transferring this length of data to the destination, the DMA_STATUS_COMPLETE_INT flag is set, and an interrupt is generated if the DMA_CFG0_COMPLETE_INT_ENABLE bit is set in the DMA_CFG0 registers.

The application can set a re-configurable trigger level for 1 to 2^{16} words in the DMA_CFG1_INT_TRANSFER_LENGTH bit field of the DMA_CFG1 register for each DMA channel; set the field to 0 to disable it. When a channel’s current transfer counter reaches the trigger level, an interrupt is generated if the counter interrupt is enabled, and all the data inside the source buffer register DMA_SRC_BUFFER is written to the destination.

Interrupt Configuration.

The interrupts used to coordinate with, and control, a DMA transfer can be defined using the DMA_CFG*_COMPLETE_INT_ENABLE and DMA_CFG*_CNT_INT_ENABLE bits from the DMA channel's DMA_CFG* registers. For more information, see Section 13.4.2.3 “DMA Interrupt Configuration” on page 697.

Transfer Priority.

The relative priority of this DMA channel's transfer. For more information, see Section 1.1.2.8 “DMA Channel Arbiter” on page 1.

IMPORTANT: The DMA channel uses counters to calculate the next source address and next destination address. The base source address, base destination address, and starting length remain unchanged. Only the counter value is modified during a transfer. This allows the DMA channel configuration to be reused (either explicitly through firmware or in circular mode) without rewriting the configuration register for multiple transfers.

The DMA channels each contain a status register (DMA_STATUS), indicating the completion status of the transfer (active, idle or complete), and the channel's interrupt status. The firmware can use this to quickly assess the status of a DMA channel.

IMPORTANT: When a DMA channel is enabled, the DMA_STATUS_ACTIVE bit in the DMA_STATUS register stays low for one cycle. This can cause a problem when polling the DMA_STATUS_ACTIVE bit directly after enabling the channel, as the DMA status is incorrect for the first cycle (this only affects optimized C or assembly code). Make sure there is at least one operation (such as a NOP) between enabling the DMA channel and polling the DMS_STATUS_ACTIVE bit.

The DMA channel can be paused by disabling it. If the current transaction is a read, the interface fills up the source buffer and is ready for a write transfer. The channel configuration is not cleared unless the DMA_CTRL_BUFFER_CLEAR and DMA_CTRL_CNTS_CLEAR bits in the DMA_CTRL register are set.

In some cases, you might wish to re-assign a DMA channel that is already in use. In this case, you probably need to save the current state of the DMA and restore and restart it later. To do so, perform the following steps:

For saving the state of the DMA:

1. Disable the selected DMA channel by issuing DMA_DISABLE (clearing the DMA_CTRL_MODE_ENABLE bit-field) to the corresponding DMA_CTRL register, without affecting any other settings in that register.
2. Wait until the value of the DMA_STATUS_ACTIVE bit in the DMA_STATUS register becomes zero.
3. Save state of the DMA_CTRL, DMA_CFG0, DMA_CFG1, DMA_STATUS, DMA_SRC_ADDR, DMA_DEST_ADDR, and DMA_SRC_BUFFER registers.

For restoring and restarting the DMA:

1. Write all of the above saved data back to the desired DMA channel’s registers.
2. When restoring DMA_STATUS, ensure that the saved value is logical OR’ed with DMA_SRC_BUFFER_FILL_LVL_WR to enable writing the buffer fill level.
3. Once this is done, write the appropriate DMA_ENABLE command to the DMA_CTRL_MODE_ENABLE bit-field of the DMA_CTRL register.

13.4.2.3 DMA Interrupt Configuration

The DMA provides separate interrupts to the core for each channel independently. Each DMA channel can be configured using its DMA_CFG0 register to assert an interrupt for several independent conditions:

A transfer has reached the counter interrupt value

Set the DMA_CFG1_INT_TRANSFER_LENGTH bit field in the channel's DMA_CFG1 register, as defined by the DMA_CFG1_TRANSFER_LENGTH bit field in the channel's DMA_CFG1 register, to a non zero value to configure a trigger level, and set the DMA_CFG0_COUNTER_INT_ENABLE bit to enable it. If the counter is set to **n**, the transfer reaches the counter interrupt value when it writes the **n**th memory word to the DMA channel's destination. This counter can be cleared using the DMA_CTRL_BUFFER_CLEAR and DMA_CTRL_CNTS_CLEAR bits in the DMA_CTRL register. This interrupt can be used to allow the firmware to operate on partial blocks of data as they are available.

A transfer is complete

Set the DMA_CFG0_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. For wrap restart transfers (circular transfers), a complete interrupt is generated each time the circular transfer is completed.

The interrupts for DMA are configurable on a per-channel basis using the DMA_CFG0 registers. A user application can read sticky bits DMA_STATUS_CNT_INT and DMA_STATUS_COMPLETE_INT in the channel status registers (DMA_STATUS) to know which interrupts have been triggered. Consequently, the user application can clear sticky status bits of the interrupts by writing to the DMA_STATUS_COMPLETE_INT_CLEAR or DMA_STATUS_COMPLETE_INT_CLEAR bit fields.

NOTE: If a status bit for a particular event is not cleared, all subsequent interrupts for that same event are masked.

13.4.2.4 Circular Mode

By enabling DMA_CFG*_SRC_ADDR_LSB_TOGGLE, a DMA transfer can have its source address effective LSB toggled after each read, resulting in a circular buffer of 2 words. DMA_CFG0*_SRC_ADDR_STEP_SIZE must be set to zero in this case. For a source word size of 4 or 8 bits, address bit 0 is toggled; for a source word size of 16 bits, address bit 1 is toggled; and for a source word size of 32 bits, address bit 2 is toggled.

By enabling DMA_CFG*_DEST_ADDR_LSB_TOGGLE, a DMA transfer can have its destination address's effective LSB toggled after each write, resulting in a circular buffer of 2 words. DMA_CFG0*_DEST_ADDR_STEP_SIZE must be set to zero in this case. For a destination word size of 4 or 8 bits, address bit 0 is toggled; for a destination word size of 16 bits, address bit 1 is toggled; and for a destination word size of 32 bits, address 2 is toggled.

13.4.2.5 Word Size

Each DMA channel can have different word sizes set for both the source and the destination. The word sizes for the source and destination data can be configured as 4-bit, 8-bit, 16-bit, or 32-bit, using the DMA_CFG0_SRC_DEST_WORD_SIZE bit field from the DMA_CFG0* register. Smaller word sizes can be used to optimize memory utilization when retrieving or storing data. Both 4-bit and 8-bit words are byte addressable, 16-bit words are half word addressable, and 32-bit words are word addressable in memory.

The "Organization of a 32-Bit memory instance" figure (Figure 71) shows the organization of a 32-bit memory instance. From the DMA perspective, it can be comprised of the following:

- One 32-bit word (word 0), or
- Two 16-bit words (word 0, word 1), or
- Four 8-bit words (word 0, word 1, word 2, word 3), or
- Eight 4-bit words (word 0, word 1, word 2, word 3, word 4, word 5, word 6, word 7)

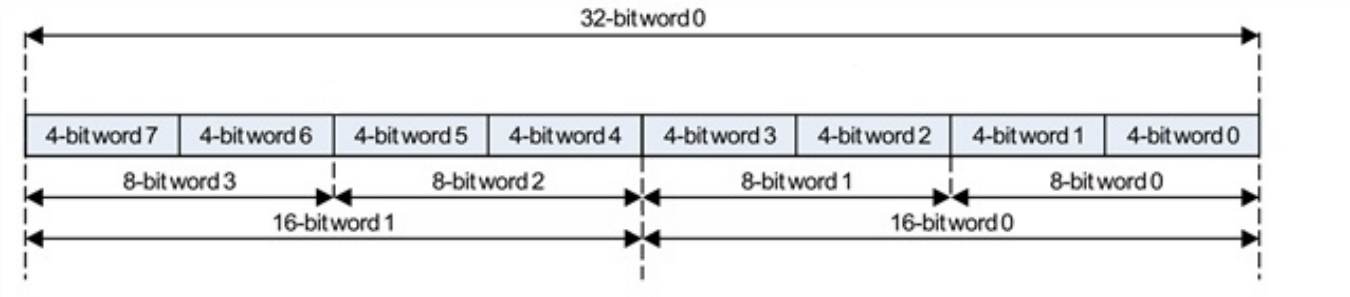


Figure 71. Organization of a 32-Bit memory instance

Read bus operations are always based on 32-bit words. However, the DMA can utilize only a portion of the data if the selected word sizes are smaller. Write bus operations can be 8-bit, 16-bit, or 32-bit words, as long as the memory system that is addressed supports this. The counter in the DMA channel always operates on the configured source word size. If the transfer type is memory-peripheral, the length sets the number of destination words transferred; otherwise it sets the number of source words transferred.The next address for reading or writing can be determined from the configured word size, the current counter, and the base address. This might result in an actual read or write, or use of the temporary register with the DMA.

13.4.2.6 Packing and Unpacking

When the source or destination word sizes are different, the DMA performs packing and/or unpacking (shown in the "DMA Packing and Unpacking" figure (Figure 72)) as required. When packing data, the DMA channel reads and packs input data into an internal buffer that is accessible through the `DMA_SRC_BUFFER` registers. When the DMA channel is unpacking data, it uses the same buffer as the source of packed data. This access is provided to the application so that it can recover from a partial data transfer. In the case of a partial transfer from a serial interface, for example, the data is available in the internal buffer.

Packing and unpacking are performed automatically, depending on the word size configuration, the byte address access, and the DMA operation mode. Packing and unpacking can be used for all types of DMA transfers, to minimize bus utilization and maximize memory use efficiency. For example, when writing four bytes to an 8-bit destination register, only one 32-bit read is required from the source memory instance. In this case, the memory instance word size is configured as 32-bit, and the peripheral word size is configured as 8-bit. When both source and destination are configured for the native word size (32-bit), no special processing is necessary (direct mode).

The data is always right aligned and can be byte addressable. When the destination word size is less than 32 bits and the destination address is not byte addressed, the data is stored in the lower portion of the 32-bit word. The upper bits are always written as zero (padded mode). When the source word size is less than 32 bits, the data is read from the lower portion of the 32-bit word. The upper bits are ignored. In addition to packing/unpacking, the DMA supports both big and little endian transfers.

NOTE: Transfers that cannot be evenly packed into the destination word size (i.e., transfers that include an incomplete final word) are only supported for little endian transfers.

If byte ordering is enabled, the DMA can also order the bytes when the source word size and destination word size are the same. In the case of a word size of 8 bits for both source and destination, the nibble is ordered instead. No ordering occurs for word sizes of 4 bits. This is shown in the two examples that follow.

- Example 1 (see also the "DMA Transfer with Src Word Size < Dest Word Size (Example 1)" table (Table 39), below):

- Transfer Type: Peripheral to Memory
- Source Word Size: 8 bits (Peripheral, for instance SPI output)
- Destination Word Size: 32 bits (Data Memory)
- Transfer Length: 5
- Byte Ordering: Little Endian

Table 39. DMA Transfer with Src Word Size < Dest Word Size (Example 1)

SOURCE	DESTINATION (Data Memory)
WordInput_8_0	
WordInput_8_1	
WordInput_8_2	
WordInput_8_3	WordOutput_32_0 = {WI8_3, WI8_2, WI8_1, WI8_0}
WordInput_8_4	WordOutput_32_1 = {{0x00, 0x00, 0x00, WI8_4}

- Example 2 (see also the "DMA Transfer with Src Word Size < Dest Word Size (Example 2)" table (Table 40), below):
  - Transfer Type: Peripheral to Memory
  - Source Word Size: 8 bits (Peripheral, for instance SPI output)
  - Destination Word Size: 16 bits (Data Memory)
  - Transfer Length: 5
  - Byte Ordering: Big Endian

Table 40. DMA Transfer with Src Word Size < Dest Word Size (Example 2)

SOURCE	DESTINATION (Data Memory)
WordInput_8_0	
WordInput_8_1	WordOutput_32_0 = {0x00, 0x00, WI8_0, WI8_1}
WordInput_8_2	
WordInput_8_3	WordOutput_32_1 = {0x00, 0x00, WI8_2, WI8_3}
WordInput_8_4	WordOutput_32_2 = {0x00, 0x00, WI8_4, 0x00}

When packing data, the DMA channel reads and packs input data into an internal buffer that is accessible through the DMA_SRC_BUFFER registers. When the DMA channel is unpacking data, it uses the same buffer as the source of packed data. The "DMA Packing and Unpacking" figure (Figure 72) shows the DMA packing and unpacking format.

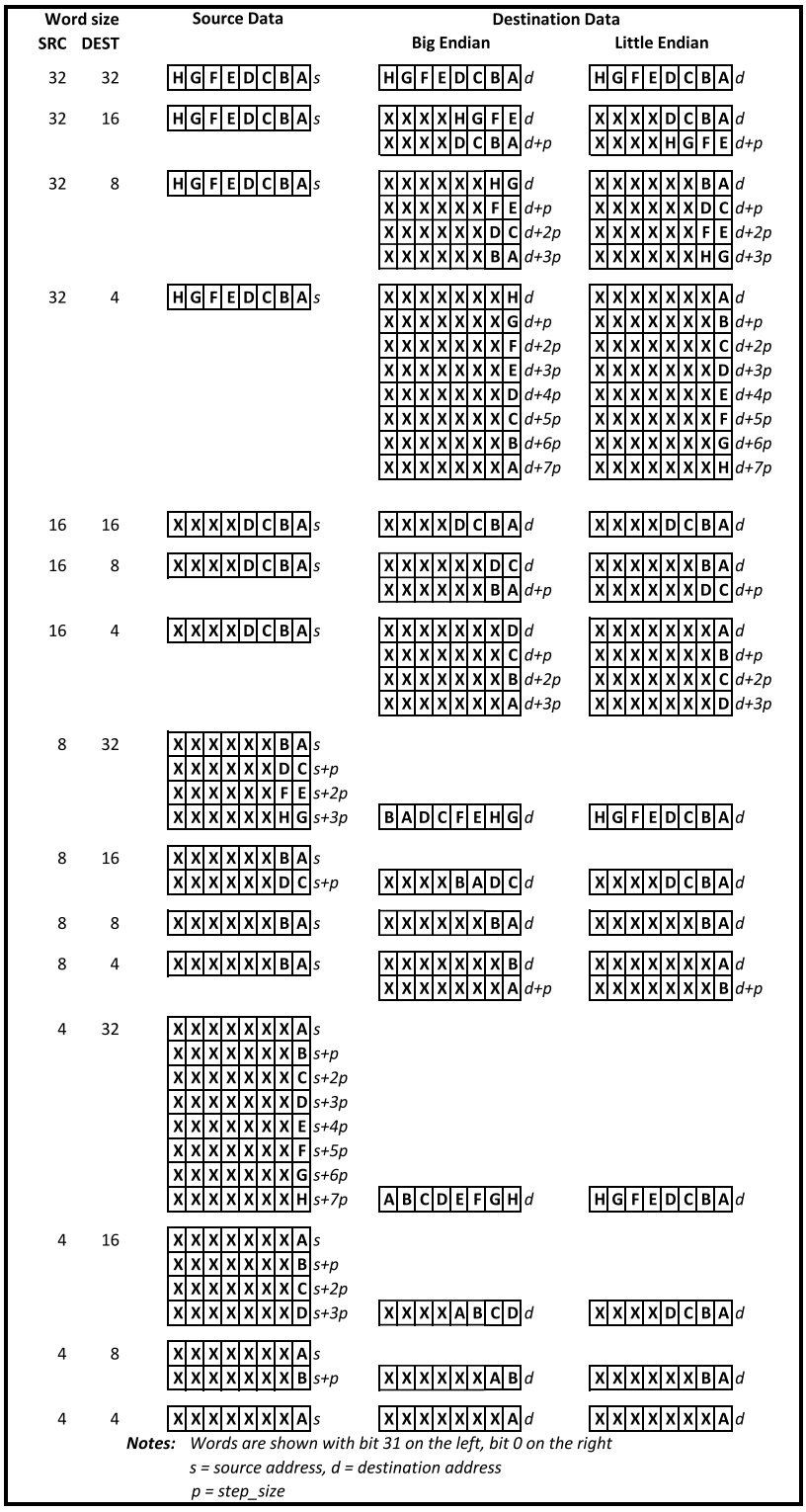


Figure 72. DMA Packing and Unpacking

DMA channels can be triggered to start without firmware intervention, based on another DMA channel completing or a specified system interrupt occurring. This can be used to off-load interrupts and data transfer tasks from the Arm Cortex-M33 processor.

The start of the DMA transfer is defined by the triggering enable modes. For example, one DMA channel can send data to the CRC module, while a second DMA channel can pull data from the CRC module without the Arm Cortex-M33 core's intervention. In any triggering mode, the DMA channel waits until a configured triggering event occurs. This triggering event is defined by the DMA_CTRL_TRIGGER_SOURCE field in the DMA_CTRL registers.

Each DMA channel needs to be configured independently (i.e., different transfer length, word sizes, increment steps, source, destination, etc.).

IMPORTANT: A DMA channel must not be configured to be triggered by itself, or else the channel remains in pending state (**DMA_STATUS_DMA_STATUS_ACTIVE**).

NOTE: If two or more DMA channels are linked to the same trigger source, they run in parallel and start new DMA threads. Any conflict associated with cross-selecting DMA channels needs to be controlled by the firmware.

The destination counter value indicates the number of transfers sent to the destination address during each DMA transfer. Upon completion, the complete interrupt of the corresponding DMA channel is triggered if enabled.

At the end of the DMA transfer, the selected triggering mode can define how the channel behaves when the transfer length is reached:

- If DMA_TRIGGER has been selected, the channel is disabled automatically, with no address wrapping.
- If DMA_TRIGGER_WRAP has been selected, the addresses are wrapped to the configured base addresses and the channel is disabled.
- If DMA_TRIGGER_WRAP_RESTART has been selected, the addresses are wrapped to the configured base addresses, and the channel restarts with another DMA transfer in a circular addressing mode with the same channel configuration. In this mode, the loop is broken when the DMA channel is disabled with the firmware's intervention.
- If DMA_TRIGGER_WRAP_TRIGGER_RESTART has been selected, the addresses are wrapped to the configured base addresses, the transfer counter is reset, and the DMA channel becomes pending, waits to be triggered again, and restarts with another DMA transfer.

NOTE: Source or destination addresses (depending on SRC_DEST_TRANS_LENGTH_SEL) can also be wrapped on every interrupt transfer length counter, as configured using the DMA_CTRL_INT_CNT_ADDR_WRAP_ENABLE bit from the DMA_CTRL register.

13.4.2.7 DMA Operation

The DMA controller works as follows:

1. The DMA channel is configured and enabled by the firmware.
    a. The operation sequence varies slightly if the transfer length depends on the destination or source word count. This can be selected using the bit location DMA_CFG0_SRC_DEST_TRANS_LENGTH_SEL in the DMA_CFG0 register.
    b. If bit DMA_CFG0_SRC_DEST_TRANS_LENGTH_SEL is set to DEST_TRANS_LENGTH_SEL, the transfer length depends on the destination transfer count. When SRC_TRANS_LENGTH_SEL is set, the transfer length depends on the source word count.
2. The operation sequence is as follows when the transfer length depends on the destination word count:

a. The DMA channel reads from the source until its source buffer is filled with enough bits to write to its destination, in accordance with its word size.
    i. If the DMA source request line is configured to be connected to a peripheral with DMA access, the channel reads from the source address when a request is issued by the peripheral.
    ii. If DMA access is not available, or if the source is a memory location, then DMA_SRC_ALWAYS_ON can be selected, and the DMA channel reads immediately.
    iii. The DMA channel reads from the source until its source buffer is filled with enough bits to write to its destination, in accordance with its word size.
b. As the source buffer is read by DMA, the DMA_STATUS_SRC_BUFFER_FILL_LVL bit-field in the DMA_STATUS register is updated to reflect the new source buffer status. The DMA channel writes to its destination, as long as the source buffer contains enough bits for the writing task and the destination word counter is less than the configured transfer length.
    i. If the DMA destination request line is configured to be connected to a peripheral with DMA access, the channel writes to the destination when this request is made by the peripheral.
    ii. If DMA access is not available, or if the source is a memory location, then DMA_DEST_ALWAYS_ON is selected as the destination and the DMA channel writes immediately.
    iii. When data is read by the DMA from the source or destination address, the data is copied into the DMA_SRC_BUFFER register. The data in this register is shifted to the right in accordance with the destination word size, and zeroes are shifted in. The DMA_STATUS_SRC_BUFFER_FILL_LVL field in the DMA0_STATUS register is updated to reflect the new source buffer status.
    iv. The DMA continues to transfer data from source to destination, until the destination transfer length (configured through the DMA_CFG1_TRANSFER_LENGTH field in the DMA_CFG1 registers) is reached. The DMA increments a destination word counter which can be read by the DMA_CNTS_TRANSFER_WORD_CNT field in the DMA_CNTS registers.
c. An interrupt can be generated for a transfer word count by writing to the DMA_CFG1_INT_TRANSFER_LENGTH field in the DMA_CFG1 registers. This transfer word count is incremented with the destination word counter of the DMA. If a non zero value for the interrupt transfer word count is written in the register, and the interrupt transfer word count in the register value reaches the configured value, an interrupt triggers, the interrupt status bit is set, and the counter resets and begins to count again. The transfer word counter continues counting and generating interrupts in the same manner until the DMA operation is complete.
d. When the destination word count equals the transfer word count, the DMA channel triggers a completed interrupt (when it is enabled), its interrupt state bit is set, and the counter resets only in continuous modes.

3. The operation is as follows when the transfer length depends on the source word count:
    a. The source word counter is incremented until it reaches the transfer word count. The counter resets only in continuous modes.
    b. When the destination word count equals the transfer word count, the DMA channel triggers a completed interrupt when it is enabled and its interrupt state bit is set. However, in this mode, if the DMA transfer is not completed, the channel restarts its operation.

13.4.2.8 DMA Channel Arbiter

Only one transfer using the DMA channels can be actively serviced at a time. The DMA contains a channel arbiter, which is responsible for determining which DMA channel to activate. Each DMA channel is assigned a channel priority in the range of 0 - 3, using the DMA_CFG0_CHANNEL_PRIORITY bit-field from the DMA_CFG0 registers. This channel priority is used by the arbiter to determine which DMA channel to service when multiple requests are pending.

Each DMA channel receives respective requests from peripherals connected to the DMA request lines. Once the DMA channel is ready to begin an operation, it indicates to the arbiter that it is ready by making a request. The DMA arbiter selects between multiple ready DMA channels based on configured channel priority.

The following situations describe scenarios when multiple requests might be pending:

- A DMA request is asserted by two or more peripherals (or a single peripheral serving two DMA channels) 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 have 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.

For registers, see "DMA Registers" on page 718.

13.5 Flash Copier

This module copies data from the flash into any DMA-accessible memory or the CRC block. This module can also run in comparison mode. In this mode, the 38-bit data read from flash is verified with a reference value, but not written to any memory. This is useful to verify that a sector has been properly erased, and for other test purposes.

The flash copier is configured using the following registers:

- `FLASH_COPY_SRC_ADDR_PTR`: defines the flash source address (byte oriented). In 32-bit copier mode or in comparator mode, this pointer must point to the beginning of a word (two LSBs are ignored).
- `FLASH_COPY_DST_ADDR_PTR`: defines the destination address (byte oriented). Addressing corresponds to logical memory instances in normal order, as viewed by the Arm Cortex-M33 processor and must point to the beginning of a word (two LSBs are ignored). This pointer is not used in comparator mode or when the flash copier destination is the CRC block.
- `FLASH_COPY_WORD_CNT`: indicates how many words are to be written in the destination (in copier mode) or the number of words to be read and verified (in comparator mode).
- `FLASH_COPY_CFG`: configures `MODE` (copier or comparator), flash access `FLASH_COPY_CFG_PRIORITY`, `FLASH_COPY_CFG_COPY_DEST` (memory or CRC), `FLASH_COPY_CFG_COMP_MODE` (constant or checkboard), `FLASH_COPY_CFG_COMP_ADDR_DIR` (up or down), and `FLASH_COPY_CFG_COMP_ADDR_STEP` (increment `FLASH_COPY_SRC_ADDR_PTR` by one or two words).

NOTE: While the flash 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.

In copier mode, the flash is read with or without error correction coding (ECC), based on the FLASH_ECC_CTRL_COPIER_ECC_CTRL bit in the FLASH_ECC_CTRL register. In comparator mode, the flash is always read without ECC. For more information on ECC, see Section 1.0.1 “Error-Correction Coding” on page 1.

The flash copier execution state and status is controlled using the FLASH_COPY_CTRL register. This includes the following control and status bits:

- The FLASH_COPY_CTRL_START bit is used to start the flash copier.
- The FLASH_COPY_CTRL_STOP bit is used to stop the flash copier; if the flash copier receives both start and stop requests at the same time, the flash copier is stopped.
    - The copy or comparator operation can be stopped using this bit before the copy has been completed. The copy operation can be continued by giving the FLASH_COPY_CTRL_START command, as the address pointers and word counter retain the values required to continue the copy operation.
- The FLASH_COPY_CTRL_BUSY status bit indicates when the flash copier is busy copying data.
- The FLASH_COPY_CTRL_ERROR status bit indicates if the flash copier has observed an error; this status bit is cleared when the flash copier is started using the FLASH_COPY_CTRL_START bit.

While the copier is busy, the following behaviors occur in each flash copier mode:

When the word counter reaches zero or when a write error occurs (copier mode only), or when a verification error occurs (comparator mode only), then the FLASH_COPY_CTRL_BUSY status bit is cleared and a FLASH0_COPY interrupt is generated. In the case of an error, the FLASH_COPY_CTRL_ERROR bit in the FLASH_COPY_CTRL register is set.

When the flash copier is running, the flash access priority is defined by the FLASH_COPY_CFG_PRIORITY bit in the FLASH_COPY_CFG register:

1. When FLASH_COPY_CFG_PRIORITY = FLASH_CPU_PRIORITY, the Arm Cortex-M33 processor has priority over the flash copier.
2. When FLASH_COPY_CFG_PRIORITY = FLASH_COPIER_PRIORITY, the flash copier has priority over the Arm Cortex-M33 processor.

When writing to its destination memory, memory arbitration priority is handled between the flash copier and any DMA memory access, as explained in Section 9.1.1 “Memory Instances” on page 466. The flash copier acts as a DMA access regarding the priority handling between the flash copier, the Arm Cortex-M33 processor, and the baseband controller.

NOTE: While the flash copier is running, it constantly tries to read from the flash memory, in case the destination memory is not available due to an access conflict with the Arm Cortex-M33 processor or DMA. Therefore, we recommend that you avoid such memory access conflicts, as this results in additional reads from flash which increase the power consumption

For registers, see Section 13.9.5 “Flash Copier Registers” on page 730.

13.6 Time of Flight

The time of flight module is designed to enable measurement of the time it takes for certain operations to complete, especially RF operations. The time of flight module accomplishes this measurement by implementing a 20-bit counter that counts SYSCLK cycles; therefore, the resolution and accuracy is determined by the SYSCLK and the TOF_CFG_CLK_PRESCALE field of the TOF_CFG register. The TOF_CLK settings higher than 8 MHz have no practical value, as the radio interrupts are generated synchronously at 8 MHz. Care must be taken if the SYSCLK is not an even multiple of 8 MHz; for instance, if the internal RC oscillator is used as the SYSCLK. The TOF_CLK is determined by:

$$TOF\_CLK = \frac{SYSCLK}{TOF\_CFG\_CLK\_PRESCALE+1} \leq 8\ MHz$$

The timer can be started or stopped using software control via the TOF_CTRL_START and TOF_CTRL_STOP bits of the TOF_CTRL register. The starting and stopping of the timer can also be controlled in hardware by the radio frequency front end's interrupts. The available hardware interrupt triggers are as configured as the start or stop trigger in the TOF_CFG_START_SRC and TOF_CFG_STOP_SRC fields, respectively, of the TOF_CFG register. The hardware interrupts available are the interrupts generated by the radio

Once the timer has been started and is counting, the counter continues to increment internally and the the TOF_STATUS_BUSY bit of the TOF_STATUS register is set. Once a stop trigger occurs, the TOF_STATUS_BUSY bit is cleared and the TOF_DATA_REQ bit in the TOF_DATA register is set, indicating a new data sample is available in that register.

The time of flight module also has an averaging feature to increase accuracy. The averaging feature can be configured by the TOF_CFG_AVG_CFG field from the TOF_CFG register, for 1 to 128 samples. The default setting is to use 1 sample for the average data; therefore, no averaging is performed, and the data in the TOF_AVG_DATA register is the same as that in the TOF_DATA register. For cases where the TOF_CFG_AVG_CFG field from the TOF_CFG register is higher than 0 (more than 1 data point), the average is calculated from the number of samples specified, and the result is placed in the TOF_AVG_DATA register. The TOF_AVG_DATA register itself is split into two sections, the integer portion located in the TOF_AVG_DATA_AVG_DATA_INT field, and a decimal portion in the TOF_AVG_DATA_AVG_DATA_DEC field.

The module also selects the minimum and maximum data points from the dataset and places these in the TOF_MIN_DATA and TOF_MAX_DATA registers respectively.

The application can check the number of samples left before the sample is computed and available in the TOF_STATUS_AVG_DATA_STATUS bit of the TOF_STATUS register.

The TOF_STATUS_AVG_DATA_REQ bit from the TOF_STATUS register functions similarly to the TOF_STATU_DATA_REQ bit in the same register, in that the bit is set high when a new average data sample is ready.

The TOF_STATUS_AVG_DATA_CLEAR, TOF_STATUS_MAX_DATA_CLEAR and the TOF_STATUS_MIN_DATA_CLEAR bits from the TOF_STATUS register all clear their respective data registers. The TOF_STATUS_AVG_DATA_CLEAR bit has the extra function of restarting the average computation.

There are three bits in the TOF_STATUS register that indicate errors and are sticky bits:

TOF_STATUS_ERROR
: Indicates that two start conditions have occurred in a row; can be cleared via the TOF_STATUS_ERROR_CLEAR bit from the TOF_STATUS register.

TOF_STATUS_AVG_DATA_OVERRUN
TOF_STATUS_DATA_OVERRUN
: Indicates that two TOF_DATA samples have been calculated without being read; can be cleared via the TOF_STATUS_DATA_OVERRUN_CLEAR bit from the TOF_STATUS register.

The time of flight module has one interrupt that can be generated by 4 sources:

- Error state, enabled by the TOF_CFG_ERROR_INT_ENABLE bit from the TOF_CFG register
- Overrun state, enabled by the TOF_CFG_OVERRUN_INT_ENABLE bit from the TOF_CFG register
- Average data available in TOF_AVG_DATA, enabled by the TOF_CFG_AVG_DATA_INT_ENABLE bit from the TOF_CFG register

- Data available in TOF_DATA, enabled by the TOF_CFG_DATA_INT_ENABLE bit from the TOF_CFG register

The time of flight module can automatically transfer its data via the DMA (see Section 13.4 “Direct Memory Access (DMA) Controller” on page 690) for configuring the DMA. One extra piece of configuration is required in the time of flight module for DMA transfers to work correctly. The TOF_CFG_AVG_DATA_DMA_ENABLE or TOF_CFG_DATA_DMA_ENABLE bits from the TOF_CFG register must be enabled to request DMA transfers whenever either TOF_AVG_DATA or TOF_DATA, respectively, is available.

The timer can be disabled by setting the TOF_CTRL_DISABLE bit or synchronously reset by setting the TOF_CTRL_RESET bit, both from the TOF_CTRL register.

- Disabling the timer resets all registers except the two configuration registers.
- Disabling has priority over any start condition.
- Resetting the timer synchronously resets all the registers to default value.

13.6.1 BLE Link Filtering

The time of flight module also offers the ability to filter unwanted radio interrupts from the baseband, based on the link format and link label information read from the baseband. All control of this feature is available through the TOF_LINK_CFG register. The TOF_LINK_CFG_LINK_FILTER_EN bit from this register enables the filter. The TOF_LINK_CFG_LINK_LABEL and TOF_LINK_CFG_LINK_FORMAT fields must be filled with the label and format information that matches the data reported from the baseband. Notably, the bit positioning of the TOF_LINK_CFG_LINK_LABEL and TOF_LINK_CFG_LINK_FORMAT fields relative to the register is identical to that of the BBIF_STATUS_LINK_LABEL and BBIF_STATUS_LINK_FORMAT fields in the BBIF_STATUS register.

For registers, see Section 13.9.6 “Time of Flight Registers” on page 732.

13.7 TIMERS

The RSL15 system provides five timers, including:

- The SysTick timer from the Arm Cortex-M33 core, which is described in Section 3.3.1 “SysTick” on page 57
- Four general-purpose timers

Each general-purpose timer provides:

- A 24-bit counter
- A 3-bit prescale factor that increases by a factor of 2 at each step, scaling between prescalars of 1 through 128.
- Three 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

NOTE: Throughout this section, the asterisk * indicates that any of the 4 timer channels, numbered 0-3, can be substituted.

The general-purpose timers are clocked from SLOWCLK, divided by either 2 or 32. The SLOWCLK divisor can be selected using the CLK_SRC bit in the appropriate TIMER*_CFG1 register. The available power of 2 prescaling divisions results in the timers achieving a wider range of timing; however, the granularity at which the timer can be configured to trigger increases in parallel. Use the TIMER_CFG0_PRESCALE bit field to set the power-of-two scaling factor.

After prescaling the timer clock, each timer can be configured to trigger after 1 to 2^{24} cycles of the prescaled clock, by setting the TIMER_CFG0_TIMEOUT_VALUE bit field in the appropriate TIMER*_CFG0 register. The resulting timer delay is equal to:

$$\mathrm{DELAY} = \frac{2^{(\mathrm{TIMER\_CFG0\_PRESCALE})} \times (\mathrm{TIMER\_CFG0\_TIMEOUT\_VALUE}+1)}{f_{\mathrm{SLOWCLK\_DIV}}}$$

Where

$f_{SLOWCLK\_DIV}$

is equal to SLOWCLK divided by 2 if the CLK_SRC bit in TIMER*_CFG1 is 1, or else it is equal to SLOWCLK divided by 32.

13.7.1 Starting or Stopping Timers

The state of a timer can be read from the TIMER_CTRL_BUSY bit of its TIMER_CTRL register.

If the timer is not running, it can be started by setting the TIMER_CTRL_START bit of its TIMER_CTRL register. If the timer is running, setting this same bit restarts the timer by reloading the TIMER_CFG0_TIMEOUT_VALUE field in the TIMER_CFG0 register with the initial value.

Each timer can be stopped at any time, by setting the TIMER_CTRL_STOP bit of its TIMER_CTRL register.

In a case where the START and STOP bits are set at the same time, the STOP bit takes precedence over the START bit and the timer is not started.

13.7.2 Mode Selection

The timers support selection between free-run and single-/multi-shot mode, using the TIMER_CFG1_MODE bit of the TIMER_CFG1 register.

- In Free-Run Mode, the timer loads the initial time-out value and counts down to 0. When the value reaches 0, the timer issues an interrupt, reloads the time-out value, and restarts the countdown timer. This process is repeated indefinitely until the timer is explicitly stopped by writing a 1 to the TIMER_CTRL_STOP bit in the TIMER_CTRL register.
- In Multi-Shot Mode, the timer loads the initial time-out value and counts down to 0. When the value reaches 0, the timer issues an interrupt, and checks the TIMER_CFG1_MULTI_COUNT bit field in the TIMER_CFG1 register to determine if the countdown timer must be restarted. This process repeats (TIMER_CFG1_MULTI_COUNT + 1) times before the timer is disabled, unless explicitly stopped by the TIMER_CTRL_STOP bit in the TIMER_CTRL register. Single-shot mode is a special case of Multi-shot mode, where the timer is configured to trigger an interrupt only one time.

13.7.3 GPIO Interrupt Capture Mode

The timers can be used to indicate when a GPIO interrupt occurs. If the timer is configured to monitor GPIO interrupts, and a GPIO interrupt is triggered, the timer stores the current timer value to the TIMER_VAL_CAPTURE register to indicate when the interrupt has occurred.

NOTE: Security levels must be aligned for the GPIO interrupt to be useable by the timer.

To enable GPIO interrupt capture mode, set the TIMER_CFG1_GPIO_INT_ENABLE bit in the TIMER_CFG1 register. To select the GPIO interrupt source used as the trigger for the timer, use the TIMER_CFG1_GPIO_INT_SRC bit-field in the TIMER_CFG1 register.

NOTE: All four GPIO interrupts can be selected as sources for the timer triggers. (See Section 10.4.1 "GPIO Interrupts" on page 518 for more information about GPIO interrupts.)

GPIO interrupt mode can be configured to operate in single or continuous mode, using the TIMER_CFG1_GPIO_INT_MODE bit in the TIMER_CFG1 register.

- When operating in single mode, the timer captures the timing of one GPIO interrupt. The GPIO interrupt capture mechanism is automatically disabled once the GPIO interrupt is generated. Setting the TIMER_CFG1_GPIO_INT_ENABLE bit high again schedules another single shot capture.
- When operating in continuous mode, the timer constantly runs and updates the TIMER_VAL_CAPTURE register when a GPIO interrupt is triggered. The TIMER_CFG1_GPIO_INT_ENABLE bit holds its value in this mode and must be cleared to disable this mode.

When the GPIO-interrupt-based capture is enabled, but the timer is off, the state of the TIMER_VAL_CAPTURE register is preserved.

For more information on configuring GPIO interrupts, see Section 10.4 "Direct Control" on page 518.

For registers, see Section 13.9.7 "Timers Registers" on page 738.

13.8 Watchdog Timer

The watchdog timer is a safety mechanism that resets a system that has malfunctioned. This safety system uses a timer that must be periodically renewed by writing WATCHDOG_REFRESH to the WATCHDOG_CTRL register before it reaches its timeout value (WATCHDOG_CTRL_TIMEOUT). The system assumes that the application's failure to acknowledge this timer before it reaches WATCHDOG_CTRL_TIMEOUT twice indicates that the system is malfunctioning and must be reset. The timer value for the watchdog timer is not visible to the core.

IMPORTANT: The watchdog counter is cleared when the DEBUG_HALT_CTRL_C_DEBUGEN bit in the DEBUG_HALT_CTRL register is set. This prevents watchdog timeouts during code development for the RSL15. The watchdog timer is disabled as long as the debug port is powered up.

The watchdog timer runs on a prescaled clock that has been derived from the SLOWCLK using a fixed division of 1024. This clock is used to control 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 reset and the prescaling counter is reset. The watchdog times out when it reaches the value contained by the WATCHDOG_CTRL_TIMEOUT bit field in the WATCHDOG_CTRL register. The number of cycles that must elapse between refresh events to trigger a watchdog timeout event is defined by the following equation:

$$TIMEOUT = \frac{1024}{f_{SLOWCLK}} * 2^{(WATCHDOG\_CTRL\_TIMEOUT+1)}$$

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 follows. For more information about resets, see Section 8.4 “Resets” on page 427.

NOTE: In standby power mode (see Section 8.6 “Power Modes” on page 431) the watchdog timer is not clocked. Despite this status, to prevent resets when returning from standby mode to run mode, we recommend refreshing the watchdog timer immediately before transitioning to standby mode.

IMPORTANT: For best practices in error, fault, and watchdog interrupt handling see Chapter 1 "Diagnostic Strategies" on page 1 from the *RSL15 Developer's Guide*.

For registers, see Section 13.9.8 “Watchdog Timer Registers” on page 741.

13.9 Peripherals Registers

For the user’s convenience, the registers for the peripherals are grouped in separate sections according to peripheral.

13.9.1 Activity Counters Registers

Register Name	Register Description	Address
SYSCTRL_CNT_CTRL	Activity Counters Control	0x4000007C
SYSCTRL_SYSCLK_CNT	System Clock Counter Value	0x40000080
SYSCTRL_SYSCLK_CNT	System Clock Counter Value	0x40000080
SYSCTRL_CBUS_CNT	CBus Activity Counter Value	0x40000088
SYSCTRL_FLASH_READ_CNT	Flash Read Access Counter Value	0x4000008C

13.9.1.1 SYSCTRL_CNT_CTRL

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

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

13.9.1.2 SYSCTRL_SYSCLK_CNT

Bit Field	Read/Write	Field Name	Description
31:0	RW	SYSCLK_CNT	System clock counter value

13.9.1.3 SYSCTRL_CM33_CNT

Bit Field	Read/Write	Field Name	Description
31:0	RW	CM33_CNT	CM33 activity counter value

13.9.1.4 SYSCTRL_CBUS_CNT

Bit Field	Read/Write	Field Name	Description
31:0	RW	CBUS_CNT	CBus-instruction activity counter value

13.9.1.5 SYSCTRL_FLASH_READ_CNT

Bit Field	Read/Write	Field Name	Description
31:0	RW	FLASH_READ_CNT	Flash read access counter value

13.9.2 Asynchronous Clock Counter Registers

Register Name	Register Description	Address
ASCC_CTRL	ASCC Control Register	0x40001700
ASCC_CFG	ASCC Configuration Register	0x40001704
ASCC_CNT	ASCC Counter Register	0x40001708
ASCC_PHASE_CNT	ASCC Phase Counter Register	0x4000170C
ASCC_PERIOD_CNT	ASCC Period Counter Register	0x40001710
ASCC_ID_NUM	ASCC ID number	0x400017FC

13.9.2.1 ASCC_CTRL

Bit Field	Read/Write	Field Name	Description
8	W	PHASE_CNT_START_NO_WAIT	Start the asynchronous clock phase counter mechanism without waiting on a sync pulse
7	R	PERIOD_CNT_STATUS	Asynchronous clock period counter status
6	W	PERIOD_CNT_STOP	Stop the asynchronous clock period counter mechanism
5	W	PERIOD_CNT_START	Start the asynchronous clock period counter mechanism
4	R	PHASE_CNT_MISSED_STATUS	Asynchronous clock phase counter missed status

Bit Field	Read/Write	Field Name	Description
3	R	PHASE_CNT_STATUS	Asynchronous clock phase counter status
2	W	PHASE_CNT_STOP	Stop the asynchronous clock phase counter mechanism
1	W	PHASE_CNT_START	Start the asynchronous clock phase counter mechanism and wait for sync pulse
0	W	CNT_RESET	Reset asynchronous clock counter

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
8	PHASE_CNT_START_NO_WAIT	PHASE_CNT_START_NO_WAIT	Reset PHASE_CNT and start the asynchronous clock phase counter mechanism without waiting on a sync pulse	0x1
7	PERIOD_CNT_STATUS	PERIOD_CNT_IDLE	The asynchronous clock period counter mechanism is idle	0x0*
		PERIOD_CNT_BUSY	The asynchronous clock period counter mechanism is busy	0x1
6	PERIOD_CNT_STOP	PERIOD_CNT_STOP	Stop the asynchronous clock period counter mechanism	0x1
5	PERIOD_CNT_START	PERIOD_CNT_START	Reset PERIOD_CNT and start the asynchronous clock period counter mechanism	0x1
4	PHASE_CNT_MISSED_STATUS	PHASE_CNT_NORMAL	The asynchronous clock phase counter mechanism was not stopped due to a missed synchronization signal	0x0*
		PHASE_CNT_MISSED	The asynchronous clock phase counter mechanism was stopped due to a missed synchronization signal	0x1
3	PHASE_CNT_STATUS	PHASE_CNT_IDLE	The asynchronous clock phase counter mechanism is idle	0x0*
		PHASE_CNT_BUSY	The asynchronous clock phase counter mechanism is busy	0x1
2	PHASE_CNT_STOP	PHASE_CNT_STOP	Stop the asynchronous clock phase counter mechanism	0x1
1	PHASE_CNT_START	PHASE_CNT_START	Reset PHASE_CNT and start the asynchronous clock phase counter mechanism and wait on a sync pulse	0x1
0	CNT_RESET	CNT_RESET	Reset asynchronous clock counter	0x1

13.9.2.2 ASCC_CFG

Bit Field	Read/Write	Field Name	Description
3:0	RW	PERIODS_CFG	Defines over how many asynchronous clock periods the period counter measures

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
3:0	PERIODS_CFG	ASCC_PERIODS_1	Measure 1 asynchronous clock period	0x0*
		ASCC_PERIODS_2	Measure 2 asynchronous clock periods	0x1
		ASCC_PERIODS_3	Measure 3 asynchronous clock periods	0x2
		ASCC_PERIODS_4	Measure 4 asynchronous clock periods	0x3
		ASCC_PERIODS_5	Measure 5 asynchronous clock periods	0x4
		ASCC_PERIODS_6	Measure 6 asynchronous clock periods	0x5
		ASCC_PERIODS_7	Measure 7 asynchronous clock periods	0x6
		ASCC_PERIODS_8	Measure 8 asynchronous clock periods	0x7
		ASCC_PERIODS_9	Measure 9 asynchronous clock periods	0x8
		ASCC_PERIODS_10	Measure 10 asynchronous clock periods	0x9
		ASCC_PERIODS_11	Measure 11 asynchronous clock periods	0xA
		ASCC_PERIODS_12	Measure 12 asynchronous clock periods	0xB
		ASCC_PERIODS_13	Measure 13 asynchronous clock periods	0xC
		ASCC_PERIODS_14	Measure 14 asynchronous clock periods	0xD
		ASCC_PERIODS_15	Measure 15 asynchronous clock periods	0xE
		ASCC_PERIODS_16	Measure 16 asynchronous clock periods	0xF

13.9.2.3 ASCC_CNT

Bit Field	Read/Write	Field Name	Description
11:0	R	CNT	Asynchronous clock counter value

13.9.2.4 ASCC_PHASE_CNT

Bit Field	Read/Write	Field Name	Description
15:0	RW	PHASE_CNT	Asynchronous clock phase counter value

13.9.2.5 ASCC_PERIOD_CNT

Bit Field	Read/Write	Field Name	Description
15:0	RW	PERIOD_CNT	Asynchronous clock period counter value

13.9.2.6 ASCC_ID_NUM

Bit Field	Read/Write	Field Name	Description
15:8	R	ASCC_MAJOR_REVISION	ASCC Major Revision number
7:0	R	ASCC_MINOR_REVISION	ASCC Minor Revision number

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
15:8	ASCC_MAJOR_ REVISION	ASCC_MAJOR_REVISION	ASCC revision 1.0	0x1*
7:0	ASCC_MINOR_ REVISION	ASCC_MINOR_REVISION	ASCC revision 1.0	0x0*

13.9.3 CRC Registers

Register Name	Register Description	Address
CRC_CFG	CRC Generator Configuration Register	0x40001600
CRC_VALUE	CRC Generator Current Value Register	0x40001604
CRC_ADD_1	CRC Generator - Add 1 Bit	0x40001608
CRC_ADD_8	CRC Generator - Add 1 Byte	0x4000160C
CRC_ADD_16	CRC Generator - Add 1 Half-word	0x40001610
CRC_ADD_24	CRC Generator - Add 3 Bytes	0x40001614
CRC_ADD_32	CRC Generator - Add 1 Word	0x40001618
CRC_FINAL	CRC Generator Final Value	0x4000161C
CRC_ID_NUM	CRC ID number	0x400016FC

13.9.3.1 CRC_CFG

Bit Field	Read/Write	Field Name	Description
4	RW	FINAL_CRC_XOR	Selects the final CRC XOR mode
3	RW	FINAL_CRC_REVERSE	Selects the final CRC reversal mode
2	RW	BIT_ORDER	Selects the bit order for bytes added to the CRC
1	RW	CRC_TYPE	Selects the CRC type
0	RW	BYTE_ORDER	Selects the endianness for bytes added to the CRC

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
4	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
3	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
2	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
1	CRC_TYPE	CRC_CCITT	CRC-CCITT algorithm selected	0x0*
		CRC_32	CRC-32 (IEEE 802.3) algorithm selected	0x1
0	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

13.9.3.2 CRC_VALUE

Bit Field	Read/Write	Field Name	Description
31:0	RW	CURRENT_CRC	CRC generator value: Write 0xFFFFFFF (32) or 0xFFFF (CCITT) to initialize the CRC, read provides the current CRC value.

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:0	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

13.9.3.3 CRC_ADD_1

Bit Field	Read/Write	Field Name	Description
0	W	CRC_ADD_1	Add 1 bit to the CRC calculation

13.9.3.4 CRC_ADD_8

Bit Field	Read/Write	Field Name	Description
7:0	W	CRC_ADD_8	Add 1 byte (8 bits) to the CRC calculation

13.9.3.5 CRC_ADD_16

Bit Field	Read/Write	Field Name	Description
15:0	W	CRC_ADD_16	Add 1 half-word (16 bits) to the CRC calculation

13.9.3.6 CRC_ADD_24

Bit Field	Read/Write	Field Name	Description
23:0	W	CRC_ADD_24	Add 3 bytes (24 bits) to the CRC calculation

13.9.3.7 CRC_ADD_32

Bit Field	Read/Write	Field Name	Description
31:0	W	CRC_ADD_32	Add 1 word (32 bits) to the CRC calculation

13.9.3.8 CRC_FINAL

Bit Field	Read/Write	Field Name	Description
31:0	R	FINAL_CRC	CRC generator final value: After XOR for CCITT or byte reversal for CRC-32

13.9.3.9 CRC_ID_NUM

Bit Field	Read/Write	Field Name	Description
15:8	R	CRC_MAJOR_REVISION	CRC Major Revision number
7:0	R	CRC_MINOR_REVISION	CRC Minor Revision number

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
15:8	CRC_MAJOR_ REVISION	CRC_MAJOR_REVISION	CRC revision 1.0	0x1*
7:0	CRC_MINOR_REVISION	CRC_MINOR_REVISION	CRC revision 1.0	0x0*

13.9.4 DMA Registers

Register Name	Register Description	Address
DMA0_CFG0	DMA Channel Configuration Register	0x40001200
DMA0_CFG1	DMA Channel Transfer Configuration Register	0x40001204
DMA0_CTRL	DMA Channel Control	0x40001208
DMA0_STATUS	DMA Channel Status	0x4000120C
DMA0_SRC_ADDR	DMA Channel Source Address	0x40001210
DMA0_DEST_ADDR	DMA Channel Destination Address	0x40001214
DMA0_CNTS	DMA Channel Counters	0x40001218
DMA0_SRC_BUFFER	DMA Channel Source buffered data after packing	0x4000121C
DMA0_ID_NUM	DMA ID number	0x400012FC
DMA1_CFG0	DMA Channel Configuration Register	0x40001300
DMA1_CFG1	DMA Channel Transfer Configuration Register	0x40001304
DMA1_CTRL	DMA Channel Control	0x40001308
DMA1_STATUS	DMA Channel Status	0x4000130C
DMA1_SRC_ADDR	DMA Channel Source Address	0x40001310
DMA1_DEST_ADDR	DMA Channel Destination Address	0x40001314
DMA1_CNTS	DMA Channel Counters	0x40001318
DMA1_SRC_BUFFER	DMA Channel Source buffered data after packing	0x4000131C
DMA1_ID_NUM	DMA ID number	0x400013FC
DMA2_CFG0	DMA Channel Configuration Register	0x40001400
DMA2_CFG1	DMA Channel Transfer Configuration Register	0x40001404
DMA2_CTRL	DMA Channel Control	0x40001408
DMA2_STATUS	DMA Channel Status	0x4000140C
DMA2_SRC_ADDR	DMA Channel Source Address	0x40001410
DMA2_DEST_ADDR	DMA Channel Destination Address	0x40001414
DMA2_CNTS	DMA Channel Counters	0x40001418

Register Name	Register Description	Address
DMA2_SRC_BUFFER	DMA Channel Source buffered data after packing	0x4000141C
DMA2_ID_NUM	DMA ID number	0x400014FC
DMA3_CFG0	DMA Channel Configuration Register	0x40001500
DMA3_CFG1	DMA Channel Transfer Configuration Register	0x40001504
DMA3_CTRL	DMA Channel Control	0x40001508
DMA3_STATUS	DMA Channel Status	0x4000150C
DMA3_SRC_ADDR	DMA Channel Source Address	0x40001510
DMA3_DEST_ADDR	DMA Channel Destination Address	0x40001514
DMA3_CNTS	DMA Channel Counters	0x40001518
DMA3_SRC_BUFFER	DMA Channel Source buffered data after packing	0x4000151C
DMA3_ID_NUM	DMA ID number	0x400015FC

13.9.4.1 DMA_CFG0

Bit Field	Read/Write	Field Name	Description
31	RW	COMPLETE_INT_ENABLE	Raise an interrupt when the DMA transfer completes
30	RW	CNT_INT_ENABLE	Raise an interrupt when the DMA transfer reaches the counter value
29	RW	DEST_ADDR_LSB_TOGGLE	Enable an address LSB toggling for the destination
28	RW	SRC_ADDR_LSB_TOGGLE	Enable an address LSB toggling for the source
27:24	RW	DEST_ADDR_STEP	Configure whether the destination address increments/decrements in terms of destination word size
23:20	RW	SRC_ADDR_STEP	Configure whether the source address increments/decrements in terms of source word size
19:14	RW	SRC_DEST_WORD_SIZE	Select the source and destination word sizes for the transfer
13:10	RW	DEST_SELECT	Select the request line for the destination
8:5	RW	SRC_SELECT	Select the request line for the source
3:2	RW	CHANNEL_PRIORITY	Select the priority level for this channel
1	RW	SRC_DEST_TRANS_LENGTH_SEL	Selects whether the transfer length counter depends on either the source word counts or the destination word count
0	RW	BYTE_ORDER	Select the byte ordering of the DMA channel

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31	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
30	CNT_INT_ENABLE	DMA_CNT_INT_DISABLE	Disable counter interrupts for DMA channel	0x0*
		DMA_CNT_INT_ENABLE	Enable counter interrupts for DMA channel	0x1
29	DEST_ADDR_LSB_ TOGGLE	DMA_DEST_ADDR_LSB_TOGGLE_ DISABLE	No toggling on the destination address LSB	0x0*
		DMA_DEST_ADDR_LSB_TOGGLE_ ENABLE	Toggle the destination address LSB	0x1
28	SRC_ADDR_LSB_ TOGGLE	DMA_SRC_ADDR_LSB_TOGGLE_ DISABLE	No toggling on the source address LSB	0x0*
		DMA_SRC_ADDR_LSB_TOGGLE_ ENABLE	Toggle the source address LSB	0x1
27:24	DEST_ADDR_STEP	DMA_DEST_ADDR_STATIC	Do not increment the destination address used by DMA channel	0x0*
		DMA_DEST_ADDR_INCR_1	Set the step size of DMA channel destination address to 1	0x1
		DMA_DEST_ADDR_INCR_2	Set the step size of DMA channel destination address to 2	0x2
		DMA_DEST_ADDR_INCR_3	Set the step size of DMA channel destination address to 3	0x3
		DMA_DEST_ADDR_INCR_4	Set the step size of DMA channel destination address to 4	0x4
		DMA_DEST_ADDR_INCR_5	Set the step size of DMA channel destination address to 5	0x5
		DMA_DEST_ADDR_INCR_6	Set the step size of DMA channel destination address to 6	0x6
		DMA_DEST_ADDR_INCR_7	Set the step size of DMA channel destination address to 7	0x7
		DMA_DEST_ADDR_DECR_8	Set the step size of DMA channel destination address to minus 8	0x8
		DMA_DEST_ADDR_DECR_7	Set the step size of DMA channel destination address to minus 7	0x9
		DMA_DEST_ADDR_DECR_6	Set the step size of DMA channel	0xA

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
			destination address to minus 6	
		DMA_DEST_ADDR_DECR_5	Set the step size of DMA channel destination address to minus 5	0xB
		DMA_DEST_ADDR_DECR_4	Set the step size of DMA channel destination address to minus 4	0xC
		DMA_DEST_ADDR_DECR_3	Set the step size of DMA channel destination address to minus 3	0xD
		DMA_DEST_ADDR_DECR_2	Set the step size of DMA channel destination address to minus 2	0xE
		DMA_DEST_ADDR_DECR_1	Set the step size of DMA channel destination address to minus 1	0xF
23:20	SRC_ADDR_STEP	DMA_SRC_ADDR_STATIC	Do not increment the source address used by DMA channel	0x0*
		DMA_SRC_ADDR_INCR_1	Set the step size of DMA channel source address to 1	0x1
		DMA_SRC_ADDR_INCR_2	Set the step size of DMA channel source address to 2	0x2
		DMA_SRC_ADDR_INCR_3	Set the step size of DMA channel source address to 3	0x3
		DMA_SRC_ADDR_INCR_4	Set the step size of DMA channel source address to 4	0x4
		DMA_SRC_ADDR_INCR_5	Set the step size of DMA channel source address to 5	0x5
		DMA_SRC_ADDR_INCR_6	Set the step size of DMA channel source address to 6	0x6
		DMA_SRC_ADDR_INCR_7	Set the step size of DMA channel source address to 7	0x7
		DMA_SRC_ADDR_DECR_8	Set the step size of DMA channel source address to minus 8	0x8
		DMA_SRC_ADDR_DECR_7	Set the step size of DMA channel source address to minus 7	0x9
		DMA_SRC_ADDR_DECR_6	Set the step size of DMA channel source address to minus 6	0xA
		DMA_SRC_ADDR_DECR_5	Set the step size of DMA channel source address to minus 5	0xB
		DMA_SRC_ADDR_DECR_4	Set the step size of DMA channel source address to minus 4	0xC

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		DMA_SRC_ADDR_DECR_3	Set the step size of DMA channel source address to minus 3	0xD
		DMA_SRC_ADDR_DECR_2	Set the step size of DMA channel source address to minus 2	0xE
		DMA_SRC_ADDR_DECR_1	Set the step size of DMA channel source address to minus 1	0xF
19:14	SRC_DEST_WORD_ SIZE	WORD_SIZE_32BITS_TO_32BITS	source data uses 32-bit word and destination data uses 32-bit word	0x0*
		WORD_SIZE_32BITS_TO_4BITS	source data uses 32-bit word and destination data uses 4-bit word	0x1
		WORD_SIZE_32BITS_TO_8BITS	source data uses 32-bit word and destination data uses 8-bit word	0x2
		WORD_SIZE_32BITS_TO_16BITS	source data uses 32-bit word and destination data uses 16-bit word	0x4
		WORD_SIZE_4BITS_TO_32BITS	source data uses 4-bit word and destination data uses 32-bit word	0x8
		WORD_SIZE_4BITS_TO_4BITS	source data uses 4-bit word and destination data uses 4-bit word	0x9
		WORD_SIZE_4BITS_TO_8BITS	source data uses 4-bit word and destination data uses 8-bit word	0xA
		WORD_SIZE_4BITS_TO_16BITS	source data uses 4-bit word and destination data uses 16-bit word	0xC
		WORD_SIZE_8BITS_TO_32BITS	source data uses 8-bit word and destination data uses 32-bit word	0x10
		WORD_SIZE_8BITS_TO_4BITS	source data uses 8-bit word and destination data uses 4-bit word	0x11
		WORD_SIZE_8BITS_TO_8BITS	source data uses 8-bit word and destination data uses 8-bit word	0x12
		WORD_SIZE_8BITS_TO_16BITS	source data uses 8-bit word and destination data uses 16-bit word	0x14
		WORD_SIZE_16BITS_TO_32BITS	source data uses 16-bit word and destination data uses 32-bit word	0x20
		WORD_SIZE_16BITS_TO_4BITS	source data uses 16-bit word and destination data uses 4-bit word	0x21
		WORD_SIZE_16BITS_TO_8BITS	source data uses 16-bit word and destination data uses 8-bit word	0x22
		WORD_SIZE_16BITS_TO_16BITS	source data uses 16-bit word and	0x24

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
			destination data uses 16-bit word	
13:10	DEST_SELECT	DMA_DEST_ALWAYS_ON	Data writes are always triggered	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_I2C0	Data writes are triggered by the I2C0 request line	0x3
		DMA_DEST_I2C1	Data writes are triggered by the I2C1 request line	0x4
		DMA_DEST_UART0	Data writes are triggered by the UART0 request line	0x5
		DMA_DEST_PCM0	Data writes are triggered by the PCM0 request line	0x6
		DMA_DEST_TOF	Data writes are triggered by the TOF request line	0x7
		DMA_DEST_NOT_USED_8	Data writes are triggered by the DEST_NOT_USED_8 request line	0x8
		DMA_DEST_NOT_USED_9	Data writes are triggered by the DEST_NOT_USED_9 request line	0x9
		DMA_DEST_NOT_USED_10	Data writes are triggered by the DEST_NOT_USED_10 request line	0xA
		DMA_DEST_NOT_USED_11	Data writes are triggered by the DEST_NOT_USED_11 request line	0xB
		DMA_DEST_NOT_USED_12	Data writes are triggered by the DEST_NOT_USED_12 request line	0xC
		DMA_DEST_NOT_USED_13	Data writes are triggered by the DEST_NOT_USED_13 request line	0xD
		DMA_DEST_NOT_USED_14	Data writes are triggered by the DEST_NOT_USED_14 request line	0xE
		DMA_DEST_NOT_USED_15	Data writes are triggered by the DEST_NOT_USED_15 request line	0xF
8:5	SRC_SELECT	DMA_SRC_ALWAYS_ON	Data reads are always triggered	0x0*
		DMA_SRC_SPI0	Data reads are triggered by the SPI0 request line	0x1
		DMA_SRC_SPI1	Data reads are triggered by the SPI1	0x2

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
			request line	
		DMA_SRC_I2C0	Data reads are triggered by the I2C0 request line	0x3
		DMA_SRC_I2C1	Data reads are triggered by the I2C1 request line	0x4
		DMA_SRC_UART0	Data reads are triggered by the UART0 request line	0x5
		DMA_SRC_PCM0	Data reads are triggered by the PCM0 request line	0x6
		DMA_SRC_TOF	Data reads are triggered by the TOF request line	0x7
		DMA_SRC_RF_IQ_READY	Data reads are triggered by the RF_IQ_ READY request line	0x8
		DMA_SRC_RF_PHASE_ADC_READY	Data reads are triggered by the RF_ PHASE_ADC_READY request line	0x9
		DMA_SRC_NOT_USED_10	Data reads are triggered by the SRC_ NOT_USED_10 request line	0xA
		DMA_SRC_NOT_USED_11	Data reads are triggered by the SRC_ NOT_USED_11 request line	0xB
		DMA_SRC_NOT_USED_12	Data reads are triggered by the SRC_ NOT_USED_12 request line	0xC
		DMA_SRC_NOT_USED_13	Data reads are triggered by the SRC_ NOT_USED_13 request line	0xD
		DMA_SRC_NOT_USED_14	Data reads are triggered by the SRC_ NOT_USED_14 request line	0xE
		DMA_SRC_NOT_USED_15	Data reads are triggered by the SRC_ NOT_USED_15 request line	0xF
3:2	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
1	SRC_DEST_TRANS_ LENGTH_SEL	DEST_TRANS_LENGTH_SEL	Transfer length counter depends on the destination word count	0x0*
		SRC_TRANS_LENGTH_SEL	Transfer length counter depends on the	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
			source word count	
0	BYTE_ORDER	DMA_LITTLE_ENDIAN	DMA channel uses little endian mode	0x0*
		DMA_BIG_ENDIAN	DMA channel uses big endian mode	0x1

13.9.4.2 DMA_CFG1

Bit Field	Read/Write	Field Name	Description
31:16	RW	INT_TRANSFER_LENGTH	Trigger a counter interrupt when the DMA transfer word count reaches this value
15:0	RW	TRANSFER_LENGTH	The length, in words, of each data transfer using DMA channel

13.9.4.3 DMA_CTRL

Bit Field	Read/Write	Field Name	Description
13	RW	INT_CNT_TRIGGER_ENABLE	Enable waiting on a trigger when an interrupt counter event occurs
12:8	RW	TRIGGER_SOURCE	Selects which event triggers this DMA channel when triggering is enabled
6	RW	INT_CNT_ADDR_WRAP_ENABLE	Enable source or destination (depending on SRC_DEST_TRANS_LENGTH_SEL) address wrapping when an interrupt counter event occurs
5	RW	CLEAR_BUFFER_WHEN_WRAPPING	Clear buffer during address wrapping
4	W	BUFFER_CLEAR	Clear source buffer
3	W	CNTS_CLEAR	Clear counters
2:0	RW	MODE_ENABLE	Enable mode of operation of the DMA Channel

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
13	INT_CNT_TRIGGER_ENABLE	DMA_INT_CNT_TRIGGER_DISABLE	The DMA channel does not pause when an interrupt counter event occurs	0x0*
		DMA_INT_CNT_TRIGGER_ENABLE	The DMA channel pauses when an interrupt counter event occurs and waits for a trigger source event to continue	0x1
12:8	TRIGGER_SOURCE	DMA_CH0_COMPLETED	This DMA channel is triggered when CH0_COMPLETED occurs	0x0*
		DMA_CH1_COMPLETED	This DMA channel is triggered when CH1_COMPLETED occurs	0x1

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		DMA_CH2_COMPLETED	This DMA channel is triggered when CH2_COMPLETED occurs	0x2
		DMA_CH3_COMPLETED	This DMA channel is triggered when CH3_COMPLETED occurs	0x3
		DMA_SENSOR	This DMA channel is triggered when SENSOR occurs	0x4
		DMA_LIN_RX	This DMA channel is triggered when LIN_RX occurs	0x5
		DMA_SAR	This DMA channel is triggered when SAR occurs	0x6
		DMA_TIMER0	This DMA channel is triggered when TIMER0 occurs	0x7
		DMA_TIMER1	This DMA channel is triggered when TIMER1 occurs	0x8
		DMA_TIMER2	This DMA channel is triggered when TIMER2 occurs	0x9
		DMA_TIMER3	This DMA channel is triggered when TIMER3 occurs	0xA
		DMA_TRIGGER_NOT_USED_11	This DMA channel is triggered when TRIGGER_NOT_USED_11 occurs	0xB
		DMA_TRIGGER_NOT_USED_12	This DMA channel is triggered when TRIGGER_NOT_USED_12 occurs	0xC
		DMA_TRIGGER_NOT_USED_13	This DMA channel is triggered when TRIGGER_NOT_USED_13 occurs	0xD
		DMA_TRIGGER_NOT_USED_14	This DMA channel is triggered when TRIGGER_NOT_USED_14 occurs	0xE
		DMA_TRIGGER_NOT_USED_15	This DMA channel is triggered when TRIGGER_NOT_USED_15 occurs	0xF
		DMA_TRIGGER_NOT_USED_16	This DMA channel is triggered when TRIGGER_NOT_USED_16 occurs	0x10
		DMA_TRIGGER_NOT_USED_17	This DMA channel is triggered when TRIGGER_NOT_USED_17 occurs	0x11
		DMA_TRIGGER_NOT_USED_18	This DMA channel is triggered when TRIGGER_NOT_USED_18 occurs	0x12
		DMA_TRIGGER_NOT_USED_19	This DMA channel is triggered when TRIGGER_NOT_USED_19 occurs	0x13
		DMA_TRIGGER_NOT_USED_20	This DMA channel is triggered when	0x14

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
			TRIGGER_NOT_USED_20 occurs	
		DMA_TRIGGER_NOT_USED_21	This DMA channel is triggered when TRIGGER_NOT_USED_21 occurs	0x15
		DMA_TRIGGER_NOT_USED_22	This DMA channel is triggered when TRIGGER_NOT_USED_22 occurs	0x16
		DMA_TRIGGER_NOT_USED_23	This DMA channel is triggered when TRIGGER_NOT_USED_23 occurs	0x17
		DMA_TRIGGER_NOT_USED_24	This DMA channel is triggered when TRIGGER_NOT_USED_24 occurs	0x18
		DMA_TRIGGER_NOT_USED_25	This DMA channel is triggered when TRIGGER_NOT_USED_25 occurs	0x19
		DMA_TRIGGER_NOT_USED_26	This DMA channel is triggered when TRIGGER_NOT_USED_26 occurs	0x1A
		DMA_TRIGGER_NOT_USED_27	This DMA channel is triggered when TRIGGER_NOT_USED_27 occurs	0x1B
		DMA_TRIGGER_NOT_USED_28	This DMA channel is triggered when TRIGGER_NOT_USED_28 occurs	0x1C
		DMA_TRIGGER_NOT_USED_29	This DMA channel is triggered when TRIGGER_NOT_USED_29 occurs	0x1D
		DMA_TRIGGER_NOT_USED_30	This DMA channel is triggered when TRIGGER_NOT_USED_30 occurs	0x1E
		DMA_TRIGGER_NOT_USED_31	This DMA channel is triggered when TRIGGER_NOT_USED_31 occurs	0x1F
6	INT_CNT_ADDR_ WRAP_ENABLE	DMA_INT_CNT_ADDR_WRAP_ DISABLE	Source address and destination addresses are wrapped normally	0x0*
		DMA_INT_CNT_ADDR_WRAP_ ENABLE	When the interrupt counter is reached, source or destination address is wrapped depending on SRC_DEST_TRANS_ LENGTH_SEL	0x1
5	CLEAR_BUFFER_ WHEN_WRAPPING	DMA_KEEP_BUFFER_WHEN_ WRAPPING	Buffer contents are not cleared during address wrapping	0x0*
		DMA_CLEAR_BUFFER_WHEN_ WRAPPING	Buffer contents are cleared during address wrapping	0x1
4	BUFFER_CLEAR	DMA_CLEAR_BUFFER	Clears the DMA source buffered data	0x1
3	CNTS_CLEAR	DMA_CLEAR_CNTS	Clears the DMA counters	0x1
2:0	MODE_ENABLE	DMA_DISABLE	Disable DMA channel (channel waits on current READ or WRITE access to finish	0x0*

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
			before the active bit is cleared)	
		DMA_ENABLE	Enable DMA channel and when completed disable	0x1
		DMA_ENABLE_WRAP	Enable DMA channel and when completed, wrap addresses and disable	0x2
		DMA_ENABLE_WRAP_RESTART	Enable DMA channel and when completed, wrap addresses and restart	0x3
		DMA_TRIGGER	When source trigger occurs enable DMA channel, when completed disable	0x4
		DMA_TRIGGER_WRAP	When source trigger occurs enable DMA channel, when completed wrap addresses and disable	0x5
		DMA_TRIGGER_WRAP_RESTART	When source trigger occurs enable DMA channel, when completed wrap addresses and restart	0x6
		DMA_TRIGGER_WRAP_TRIGGER_RESTART	When source trigger occurs enable DMA channel, when completed wrap addresses, and upon next trigger event restart	0x7

13.9.4.4 DMA_STATUS

Bit Field	Read/Write	Field Name	Description
10	R	ACTIVE	Active status of the channel
9	R	CNT_INT	Indicate if a counter interrupt has occurred on DMA channel
8	R	COMPLETE_INT	Indicate if a complete interrupt has occurred on DMA channel
6	W	CNT_INT_CLEAR	Clear the counter interrupt flag
5	W	COMPLETE_INT_CLEAR	Clear the complete interrupt flag
4	W	SRC_BUFFER_FILL_LVL_WR	Enable writing of SRC_BUFFER_FILL_LVL
3:0	RW	SRC_BUFFER_FILL_LVL	The number of nibbles currently in the source buffer (0: buffer is empty, 8: the 32-bit buffer is completely full)

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
10	ACTIVE	DMA_NON_ACTIVE	DMA channel is disabled or waiting for a trigger event.	0x0*
		DMA_ACTIVE	DMA channel is running	0x1
9	CNT_INT	DMA_CNT_INT_FALSE	Indicate that no counter interrupt has occurred	0x0*
		DMA_CNT_INT_TRUE	Indicate that a counter interrupt has occurred	0x1
8	COMPLETE_INT	DMA_COMPLETE_INT_FALSE	Indicate that a no complete interrupt has occurred	0x0*
		DMA_COMPLETE_INT_TRUE	Indicate that a complete interrupt has occurred	0x1
6	CNT_INT_CLEAR	DMA_CNT_INT_CLEAR	Clear the counter interrupt flag	0x1
5	COMPLETE_INT_CLEAR	DMA_COMPLETE_INT_CLEAR	Clear the complete interrupt flag	0x1
4	SRC_BUFFER_FILL_LVL_WR	DMA_SRC_BUFFER_FILL_LVL_WR	Enable writing of SRC_BUFFER_FILL_LVL	0x1

13.9.4.5 DMA_SRC_ADDR

Bit Field	Read/Write	Field Name	Description
31:0	RW	SRC_ADDR	Address for the source of data transferred using DMA channel

13.9.4.6 DMA_DEST_ADDR

Bit Field	Read/Write	Field Name	Description
31:0	RW	DEST_ADDR	Address for the destination of data transferred using DMA channel

13.9.4.7 DMA_CNTS

Bit Field	Read/Write	Field Name	Description
31:16	RW	INT_TRANSFER_WORD_CNT	Interrupt word counter (automatically reset after each counter interrupt)
15:0	RW	TRANSFER_WORD_CNT	The number of words that have been transferred using DMA channel during the current transfer

13.9.4.8 DMA_SRC_BUFFER

Bit Field	Read/Write	Field Name	Description
31:0	RW	SRC_BUFFER	Packed word which has been read from the source addresses.

13.9.4.9 DMA_ID_NUM

Bit Field	Read/Write	Field Name	Description
20	R	DMA_24BIT_WORD	24-bit word size supported
19:16	R	DMA_NUMBER	DMA channel number
15:8	R	DMA_MAJOR_REVISION	DMA Major Revision number
7:0	R	DMA_MINOR_REVISION	DMA Minor Revision number

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
20	DMA_24BIT_WORD	DMA_24BIT_WORD_NOT_SUPPORTED	24-bit word size not supported	0x0*
		DMA_24BIT_WORD_SUPPORTED	24-bit word size supported	0x1
15:8	DMA_MAJOR_REVISION	DMA_MAJOR_REVISION	DMA revision 1.0	0x1*
7:0	DMA_MINOR_REVISION	DMA_MINOR_REVISION	DMA revision 1.0	0x0*

13.9.5 Flash Copier Registers

Register Name	Register Description	Address
FLASH0_COPY_CFG	Flash Copier Config Register	0x4000088C
FLASH0_COPY_CTRL	Flash Copier Control and Status	0x40000898
FLASH0_COPY_SRC_ADDR_PTR	Flash Copier Source Address Pointer	0x4000089C
FLASH0_COPY_DST_ADDR_PTR	Flash Copier Destination Address Pointer	0x400008A0
FLASH0_COPY_WORD_CNT	Flash Copier Word Count	0x400008A4

13.9.5.1 FLASH_COPY_CFG

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

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
18	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
17	COMP_ADDR_DIR	COMP_ADDR_DOWN	FLASH_COPIER address count-down	0x0
		COMP_ADDR_UP	FLASH_COPIER address count-up	0x1*
16	COMP_MODE	COMP_MODE_CONSTANT	FLASH_DATA[1:0] compare with Flash DOUT	0x0*
		COMP_MODE_CHBK	Odd address compare with FLASH_ DATA[1:0], even address compare with inverse FLASH_DATA[1:0]	0x1
9	COPY_DEST	COPY_TO_MEM	Copy Flash to memory	0x0*
		COPY_TO_CRC	Copy Flash to CRC	0x1
8	COPY_MODE	COPY_TO_32BIT	Copy Flash to 32-bit memory	0x0*
1	PRIORITY	FLASH_CM33_PRIORITY	CM33 has priority over Flash Copier	0x0*
		FLASH_COPY_PRIORITY	Flash Copier has priority over CM33	0x1
0	MODE	COPY_MODE	Flash copier mode	0x0*
		COMPARATOR_MODE	Flash comparator mode	0x1

13.9.5.2 FLASH_COPY_CTRL

Bit Field	Read/Write	Field Name	Description
3	R	ERROR	Error status
2	W	STOP	Stop the transfer
1	W	START	Start the transfer
0	R	BUSY	Busy status

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

13.9.5.3 FLASH_COPY_SRC_ADDR_PTR

Bit Field	Read/Write	Field Name	Description
20:0	RW	COPY_SRC_ADDR_PTR	Source address pointer

13.9.5.4 FLASH_COPY_DST_ADDR_PTR

Bit Field	Read/Write	Field Name	Description
31:2	RW	COPY_DST_ADDR_PTR	Destination address pointer

13.9.5.5 FLASH_COPY_WORD_CNT

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

13.9.6 Time of Flight Registers

Register Name	Register Description	Address
TOF_CFG	Time Of Flight Timer Configuration	0x40001E00
TOF_CTRL	Time Of Flight Timer Control	0x40001E04
TOF_STATUS	Time Of Flight Timer Status	0x40001E08
TOF_LINK_CFG	Time Of Flight Timer BLE Link Filtering Configuration	0x40001E0C
TOF_DATA	Time Of Flight Timer Data	0x40001E10
TOF_MIN_DATA	Time Of Flight Timer Minimum Data	0x40001E14
TOF_MAX_DATA	Time Of Flight Timer Maximum Data	0x40001E18
TOF_AVG_DATA	Time Of Flight Timer Average Data	0x40001E1C
TOF_ID_NUM	Time Of Flight Timer ID Number	0x40001EFC

13.9.6.1 TOF_CFG

Bit Field	Read/Write	Field Name	Description
21	RW	ERROR_INT_ENABLE	Enable the interrupt generation when an error is detected
20	RW	OVERRUN_INT_ENABLE	Enable the interrupt generation when a data or average data overrun is detected
19	RW	AVG_DATA_INT_ENABLE	Enable the interrupt generation when a new average data is available
18	RW	DATA_INT_ENABLE	Enable the interrupt generation when a new data is available
17	RW	AVG_DATA_DMA_ENABLE	Enable the DMA request when a new average data is available

Bit Field	Read/Write	Field Name	Description
16	RW	DATA_DMA_ENABLE	Enable the DMA request when a new data is available
14:12	RW	AVG_CFG	Select the amount of data for averaging
10:8	RW	STOP_SRC	Select the source of the external stop trigger
6:4	RW	START_SRC	Select the source of the external start trigger
1:0	RW	CLK_PRESCALE	Select the time of flight timer clock prescale

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
21	ERROR_INT_ENABLE	TOF_ERROR_INT_DISABLE	Disable the error interrupt generation	0x0*
		TOF_ERROR_INT_ENABLE	Enable the error interrupt generation	0x1
20	OVERRUN_INT_ ENABLE	TOF_OVERRUN_INT_DISABLE	Disable the overrun interrupt generation	0x0*
		TOF_OVERRUN_INT_ENABLE	Enable the overrun interrupt generation	0x1
19	AVG_DATA_INT_ ENABLE	TOF_AVG_DATA_INT_DISABLE	Disable the average data interrupt generation	0x0*
		TOF_AVG_DATA_INT_ENABLE	Enable the average data interrupt generation	0x1
18	DATA_INT_ENABLE	TOF_DATA_INT_DISABLE	Disable the data interrupt generation	0x0*
		TOF_DATA_INT_ENABLE	Enable the data interrupt generation	0x1
17	AVG_DATA_DMA_ ENABLE	TOF_AVG_DATA_DMA_DISABLE	Disable the average data DMA request	0x0*
		TOF_AVG_DATA_DMA_ENABLE	Enable the average data DMA request	0x1
16	DATA_DMA_ENABLE	TOF_DATA_DMA_DISABLE	Disable the data DMA request	0x0*
		TOF_DATA_DMA_ENABLE	Enable the data DMA request	0x1
14:12	AVG_CFG	TOF_AVG_DATA_1	Store the latest data until the next stop trigger occurs	0x0*
		TOF_AVG_DATA_2	Compute average of 2 consecutive data	0x1
		TOF_AVG_DATA_4	Compute average of 4 consecutive data	0x2
		TOF_AVG_DATA_8	Compute average of 8 consecutive data	0x3
		TOF_AVG_DATA_16	Compute average of 16 consecutive data	0x4
		TOF_AVG_DATA_32	Compute average of 32 consecutive data	0x5
		TOF_AVG_DATA_64	Compute average of 64 consecutive	0x6

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
			data	
		TOF_AVG_DATA_128	Compute average of 128 consecutive data	0x7
10:8	STOP_SRC	TOF_STOP_SRC_NONE	Disable external stop trigger	0x0*
		TOF_STOP_SRC_TX	Select TX radio interrupt to stop the time of flight timer	0x1
		TOF_STOP_SRC_RX_STOP	Select RX_STOP radio interrupt to stop the time of flight timer	0x2
		TOF_STOP_SRC_RX_RECEIVED	Select RX_RECEIVE radio interrupt to stop the time of flight timer	0x3
		TOF_STOP_SRC_SYNC	Select SYNC radio interrupt to stop the time of flight timer	0x4
		TOF_STOP_SRC_TX_FIFO	Select TX_FIFO radio interrupt to stop the time of flight timer	0x5
		TOF_STOP_SRC_RX_FIFO	Select RX_FIFO radio interrupt to stop the time of flight timer	0x6
6:4	START_SRC	TOF_START_SRC_NONE	Disable external start trigger	0x0*
		TOF_START_SRC_TX	Select TX radio interrupt to start the time of flight timer	0x1
		TOF_START_SRC_RX_STOP	Select RX_STOP radio interrupt to start the time of flight timer	0x2
		TOF_START_SRC_RX_RECEIVED	Select RX_RECEIVE radio interrupt to start the time of flight timer	0x3
		TOF_START_SRC_SYNC	Select SYNC radio interrupt to start the time of flight timer	0x4
		TOF_START_SRC_TX_FIFO	Select TX_FIFO radio interrupt to start the time of flight timer	0x5
		TOF_START_SRC_RX_FIFO	Select RX_FIFO radio interrupt to start the time of flight timer	0x6
1:0	CLK_PRESCALE	TOF_CLK_PRESCALE_1	Divide the system clock by 1 (SYSCLK @ 8 MHz)	0x0*
		TOF_CLK_PRESCALE_2	Divide the system clock by 2 (SYSCLK @ 16 MHz)	0x1
		TOF_CLK_PRESCALE_3	Divide the system clock by 3 (SYSCLK @ 24 MHz)	0x2
		TOF_CLK_PRESCALE_6	Divide the system clock by 6 (SYSCLK @ 48 MHz)	0x3

13.9.6.2 TOF_CTRL

Bit Field	Read/Write	Field Name	Description
8	R	ENABLE_STATUS	Status of the time of flight timer
4	W	STOP	Stop the time of flight timer
3	W	START	Start the time of flight timer
2	W	RESET	Synchronously reset the time of flight timer
1	W	DISABLE	Disable the time of flight timer
0	W	ENABLE	Enable the time of flight timer

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
8	ENABLE_STATUS	TOF_STATUS_DISABLED	Time of flight timer is disabled	0x0*
		TOF_STATUS_ENABLED	Time of flight timer is enabled	0x1
4	STOP	TOF_STOP	Stop the time of flight timer	0x1
3	START	TOF_START	Start the time of flight timer	0x1
2	RESET	TOF_RESET	Synchronously reset the time of flight timer	0x1
1	DISABLE	TOF_DISABLE	Disable the time of flight timer	0x1
0	ENABLE	TOF_ENABLE	Enable the time of flight timer	0x1

13.9.6.3 TOF_STATUS

Bit Field	Read/Write	Field Name	Description
23:16	R	AVG_DATA_STATUS	Average data timer status
13	R	AVG_DATA_REQ	Indicate that a new average data can be read
12	R	DATA_REQ	Indicate that a new data can be read
11	R	BUSY	Indicate if the time of flight timer is idle or busy
10	R	ERROR	Detect two consecutive start triggers (sticky bit)
9	R	AVG_DATA_OVERRUN	Indicate that an average data overrun has occurred (sticky bit)
8	R	DATA_OVERRUN	Indicate that a data overrun has occurred (sticky bit)
5	W	AVG_DATA_CLEAR	Clear the average data register and restart the average computation
4	W	MAX_DATA_CLEAR	Clear the max data register
3	W	MIN_DATA_CLEAR	Clear the min data register
2	W	ERROR_CLEAR	Clear the error flag
1	W	AVG_DATA_OVERRUN_CLEAR	Clear the average data overrun flag
0	W	DATA_OVERRUN_CLEAR	Clear the data overrun flag

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
23:16	AVG_DATA_STATUS	TOF_AVG_DATA_STATUS	Remaining data for averaging	0x0*
13	AVG_DATA_REQ	TOF_NO_AVG_DATA_REQ	New average data are not yet available	0x0*
		TOF_AVG_DATA_REQ	New average data can be read	0x1
12	DATA_REQ	TOF_NO_DATA_REQ	New data are not yet available	0x0*
		TOF_DATA_REQ	New data can be read	0x1
11	BUSY	TOF_IDLE	The time of flight timer is idle	0x0*
		TOF_BUSY	The time of flight timer is busy	0x1
10	ERROR	TOF_NO_ERROR	Start and stop triggers occurred	0x0*
		TOF_ERROR	Two consecutive start triggers occurred	0x1
9	AVG_DATA_OVERRUN	TOF_NO_AVG_DATA_OVERRUN	No average data overrun detected	0x0*
		TOF_AVG_DATA_OVERRUN	Data average overrun detected	0x1
8	DATA_OVERRUN	TOF_NO_DATA_OVERRUN	No data overrun detected	0x0*
		TOF_DATA_OVERRUN	Data overrun detected	0x1
5	AVG_DATA_CLEAR	TOF_AVG_DATA_CLEAR	Clear the average data register and restart the average computation	0x1
4	MAX_DATA_CLEAR	TOF_MAX_DATA_CLEAR	Clear the max data register	0x1
3	MIN_DATA_CLEAR	TOF_MIN_DATA_CLEAR	Clear the min data register	0x1
2	ERROR_CLEAR	TOF_ERROR_CLEAR	Clear the error flag	0x1
1	AVG_DATA_OVERRUN_ CLEAR	TOF_AVG_DATA_OVERRUN_CLEAR	Clear the average data overrun flag	0x1
0	DATA_OVERRUN_ CLEAR	TOF_DATA_OVERRUN_CLEAR	Clear the data overrun flag	0x1

13.9.6.4 TOF_LINK_CFG

Bit Field	Read/Write	Field Name	Description
16:12	RW	LINK_FORMAT	Configure the link format for BLE link filtering
8:4	RW	LINK_LABEL	Configure the link label for BLE link filtering
0	RW	LINK_FILTER_EN	Enable the BLE link filtering based on link label and format

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
16:12	LINK_FORMAT	TOF_LINK_FORMAT	Format of the BLE link to filter in the radio interrupts	0x0*
8:4	LINK_LABEL	TOF_LINK_LABEL	Label of the BLE link to filter in the radio interrupts	0x0*
0	LINK_FILTER_EN	TOF_LINK_FILTER_DISABLE	Disable external start trigger	0x0*
		TOF_LINK_FILTER_ENABLE	Select TX radio interrupt to start the Time of flight timer	0x1

13.9.6.5 TOF_DATA

Bit Field	Read/Write	Field Name	Description
19:0	R	DATA	Time of flight timer data (unsigned)

13.9.6.6 TOF_MIN_DATA

Bit Field	Read/Write	Field Name	Description
19:0	R	MIN_DATA	Time of flight minimum data (unsigned)

13.9.6.7 TOF_MAX_DATA

Bit Field	Read/Write	Field Name	Description
19:0	R	MAX_DATA	Time of flight maximum data (unsigned)

13.9.6.8 TOF_AVG_DATA

Bit Field	Read/Write	Field Name	Description
27:8	R	AVG_DATA_INT	Time of flight average data (unsigned integer part)
7:0	R	AVG_DATA_DEC	Time of flight average data (unsigned decimal part)

13.9.6.9 TOF_ID_NUM

Bit Field	Read/Write	Field Name	Description
15:8	R	TOF_MAJOR_REVISION	Time of flight timer major revision number
7:0	R	TOF_MINOR_REVISION	Time of flight timer minor revision number

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
15:8	TOF_MAJOR_REVISION	TOF_MAJOR_REVISION	TOF revision 1.0	0x1*
7:0	TOF_MINOR_REVISION	TOF_MINOR_REVISION	TOF revision 1.0	0x0*

13.9.7 Timers Registers

Register Name	Register Description	Address
TIMER0_CFG0	Timer Configuration 0 Register	0x40000400
TIMER0_CFG1	Timer Configuration 1 Register	0x40000404
TIMER0_CTRL	General-Purpose timer Control / Status Register	0x40000408
TIMER0_VAL	Timer Current Value Register	0x4000040C
TIMER0_VAL_CAPTURE	Timer GPIO Interrupt Captured Value Register	0x40000410
TIMER0_ID_NUM	Timer ID number	0x400004FC
TIMER1_CFG0	Timer Configuration 0 Register	0x40000500
TIMER1_CFG1	Timer Configuration 1 Register	0x40000504
TIMER1_CTRL	General-Purpose timer Control / Status Register	0x40000508
TIMER1_VAL	Timer Current Value Register	0x4000050C
TIMER1_VAL_CAPTURE	Timer GPIO Interrupt Captured Value Register	0x40000510
TIMER1_ID_NUM	Timer ID number	0x400005FC
TIMER2_CFG0	Timer Configuration 0 Register	0x40000600
TIMER2_CFG1	Timer Configuration 1 Register	0x40000604
TIMER2_CTRL	General-Purpose timer Control / Status Register	0x40000608
TIMER2_VAL	Timer Current Value Register	0x4000060C
TIMER2_VAL_CAPTURE	Timer GPIO Interrupt Captured Value Register	0x40000610
TIMER2_ID_NUM	Timer ID number	0x400006FC
TIMER3_CFG0	Timer Configuration 0 Register	0x40000700
TIMER3_CFG1	Timer Configuration 1 Register	0x40000704
TIMER3_CTRL	General-Purpose timer Control / Status Register	0x40000708
TIMER3_VAL	Timer Current Value Register	0x4000070C
TIMER3_VAL_CAPTURE	Timer GPIO Interrupt Captured Value Register	0x40000710
TIMER3_ID_NUM	Timer ID number	0x400007FC

13.9.7.1 TIMER_CFG0

Bit Field	Read/Write	Field Name	Description
26:24	RW	PRESCALE	Prescale value of the timer
23:0	RW	TIMEOUT_VALUE	Number of Timer clock cycles to time-out

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
26:24	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

13.9.7.2 TIMER_CFG1

Bit Field	Read/Write	Field Name	Description
12	RW	CLK_SRC	Clock source
10:8	RW	MULTI_COUNT	Multi-count value
6:4	RW	GPIO_INT_SRC	GPIO interrupt source selection
2	RW	GPIO_INT_MODE	GPIO interrupt capture mode
1	RW	GPIO_INT_ENABLE	GPIO interrupt capture enable
0	RW	MODE	Timer mode

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
12	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
10:8	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

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
		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
6:4	GPIO_INT_SRC	TIMER_SRC_GPIO_INT0	Select GPIO interrupt defined in GPIO_INT_CFG0	0x0*
		TIMER_SRC_GPIO_INT1	Select GPIO interrupt defined in GPIO_INT_CFG1	0x1
		TIMER_SRC_GPIO_INT2	Select GPIO interrupt defined in GPIO_INT_CFG2	0x2
		TIMER_SRC_GPIO_INT3	Select GPIO interrupt defined in GPIO_INT_CFG3	0x3
2	GPIO_INT_MODE	TIMER_GPIO_INT_SINGLE	GPIO interrupt single capture mode	0x0*
		TIMER_GPIO_INT_CONTINUOUS	GPIO interrupt continuous capture mode	0x1
1	GPIO_INT_ENABLE	TIMER_GPIO_INT_DISABLE	Disable GPIO interrupt capture	0x0*
		TIMER_GPIO_INT_ENABLE	Enable GPIO interrupt capture	0x1
0	MODE	TIMER_SHOT_MODE	One shot / multi shot mode	0x0*
		TIMER_FREE_RUN	Free-run mode	0x1

13.9.7.3 TIMER_CTRL

Bit Field	Read/Write	Field Name	Description
8	R	BUSY	Indicate if the timer is active or not
1	W	START	Start or restart the timer
0	W	STOP	Stop the timer

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
8	BUSY	TIMER_INACTIVE	The timer is inactive	0x0*
		TIMER_ACTIVE	The timer is active	0x1
1	START	TIMER_START	Writing a 1 starts or restarts the timer	0x1
0	STOP	TIMER_STOP	Writing a 1 stops the timer	0x1

13.9.7.4 TIMER_VAL

Bit Field	Read/Write	Field Name	Description
26:24	R	MULTI_COUNT_VAL	Current multi counter value
23:0	R	TIMER_VALUE	Current timer value

13.9.7.5 TIMER_VAL_CAPTURE

Bit Field	Read/Write	Field Name	Description
26:24	RW	MULTI_COUNT_VAL_GPIO	Current multi counter value captured by the GPIO interrupt defined in GPIO_INT_SRC
23:0	RW	TIMER_VALUE_GPIO	Timer value captured by the GPIO interrupt defined in GPIO_INT_SRC

13.9.7.6 TIMER_ID_NUM

Bit Field	Read/Write	Field Name	Description
20	R	TIMER_GPIO_CAP	Implementation of the GPIO triggered capture
19:16	R	TIMER_ID_NUM	TIMER instance number
15:8	R	TIMER_MAJOR_REVISION	TIMER Major Revision number
7:0	R	TIMER_MINOR_REVISION	TIMER Minor Revision number

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
20	TIMER_GPIO_CAP	TIMER_GPIO_CAP_DISABLED	GPIO capture is not implemented	0x0*
		TIMER_GPIO_CAP_ENABLED	GPIO capture is implemented	0x1
15:8	TIMER_MAJOR_REVISION	TIMER_MAJOR_REVISION	Timer revision 1.0	0x1*
7:0	TIMER_MINOR_REVISION	TIMER_MINOR_REVISION	Timer revision 1.0	0x0*

13.9.8 Watchdog Timer Registers

Register Name	Register Description	Address
WATCHDOG_CFG	Watchdog timer Configuration Register	0x40000300
WATCHDOG_CTRL	Watchdog Refresh Control Register	0x40000304
WATCHDOG_ID_NUM	WATCHDOG ID number	0x400003FC

13.9.8.1 WATCHDOG_CFG

Bit Field	Read/Write	Field Name	Description
3:0	RW	TIMEOUT_VALUE	Watchdog timeout period. Values 0xC to 0xF result in the same timeout period as the value 0xB.

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
3:0	TIMEOUT_VALUE	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*

13.9.8.2 WATCHDOG_CTRL

Bit Field	Read/Write	Field Name	Description
31:0	W	WATCHDOG_REFRESH	Write a key to reset the watchdog

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
31:0	WATCHDOG_ REFRESH	WATCHDOG_REFRESH	Write 32-bit key to reset the watchdog (others values have no effect)	0x2B1E211

13.9.8.3 WATCHDOG_ID_NUM

Bit Field	Read/Write	Field Name	Description
15:8	R	WATCHDOG_MAJOR_REVISION	WATCHDOG Major Revision number
7:0	R	WATCHDOG_MINOR_REVISION	WATCHDOG Minor Revision number

Bit Field	Field Name	Value Symbol	Value Description	Hex Value
15:8	WATCHDOG_MAJOR_REVISION	WATCHDOG_MAJOR_REVISION	Watchdog revision 1.0	0x1*
7:0	WATCHDOG_MINOR_REVISION	WATCHDOG_MINOR_REVISION	Watchdog revision 1.0	0x0*

APPENDIX A

Registers

This appendix lists all the registers that are available. Refer to the appropriate section for information about the registers for a block. The sections are:

- Section A.1 “System Control” on page 745
- Section A.2 “Clock Generation” on page 763
- Section A.3 “Reset” on page 765
- Section A.4 “Watchdog Timer” on page 766
- Section A.5 “General-Purpose Timers” on page 767
- Section A.6 “Flash Interface” on page 772
- Section A.8 “DMA Controller” on page 786
- Section A.7 “GPIO Interface and Digital Pad control” on page 781
- Section A.9 “LIN Interface” on page 797
- Section A.10 “LSAD” on page 801
- Section A.11 “Sensor Interface” on page 804
- Section A.12 “SPI Interface” on page 807
- Section A.13 “PCM Interface” on page 813
- Section A.14 “I2C Interface” on page 815
- Section A.15 “TEST Control Interface” on page 822
- Section A.16 “UART Interface” on page 823
- Section A.17 “PWM Interface” on page 825
- Section A.18 “CRC Generator Control” on page 826
- Section A.19 “ASCC” on page 827
- Section A.20 “ACS” on page 829
- Section A.21 “AHB registers” on page 842
- Section A.22 “Baseband Controller Interface” on page 843
- Section A.23 “Baseband Controller” on page 845
- Section A.24 “RF Front-End 2.4 GHz” on page 872
- Section A.25 “Implementation Control Block” on page 1029
- Section A.26 “SysTick Timer” on page 1030
- Section A.27 “Time Of Flight Timer” on page 1031
- Section A.28 “System Control and ID register not in the SCB” on page 1034
- Section A.29 “Nested Vector Interrupt Controller” on page 1035
- Section A.30 “System Control Block” on page 1056
- Section A.31 “Memory Protection Unit” on page 1062
- Section A.32 “Security Attribution Unit” on page 1064
- Section A.33 “Debug Control Block” on page 1065
- Section A.34 “Software Interrupt Generation” on page 1066
- Section A.35 “Floating-Point Extension” on page 1066

A.1 SYSTEM CONTROL

Address	Register Name	Register Write	Register Read	Default	Description
0x40000000	SYSCTRL_CM33_LOOP_CACHE_CFG	(0) CM33_LOOP_CACHE_ENABLE	(0) CM33_LOOP_CACHE_ENABLE	0x0	CM33 loop cache enable
0x40000004	SYSCTRL_ACCESS_ERROR	(16) ACCESS_ERROR_CLEAR	–	N/A	Write a 1 to clear the access error flags
		–	(12) CC312_MEM_ERROR	0x0	CC312 memory error flag
		–	(10) DMA_PERIPH_ERROR	0x0	DMA peripheral access error flag
		–	(9) DMA_MEM_ERROR	0x0	DMA memory error flag
		–	(8) BB_MEM_ERROR	0x0	Baseband memory error flag
		–	(0) FLASH_COPIER_MEM_ERROR	0x0	FLASH[0:0] copier memory error flag
0x40000008	SYSCTRL_MEM_POWER_STARTUP	(17:16) BB_DRAM_STARTUP	(17:16) BB_DRAM_STARTUP	0x3	Baseband DRAM[1:0] power startup control
		(15:8) DRAM_STARTUP	(15:8) DRAM_STARTUP	0xFF	DRAM[7:0] power startup control
		(1) FLASH_STARTUP	(1) FLASH_STARTUP	0x0	Flash[0:0] power startup control
		(0) PROM_STARTUP	(0) PROM_STARTUP	0x1	PROM power startup control
0x4000000C	SYSCTRL_MEM_POWER_ENABLE	(17:16) BB_DRAM_ENABLE	(17:16) BB_DRAM_ENABLE	0x0	Baseband DRAM[1:0] power enable control
		(15:8) DRAM_ENABLE	(15:8) DRAM_ENABLE	0x1	DRAM[7:0] power enable control
		(1) FLASH_ENABLE	(1) FLASH_ENABLE	0x0	Flash[0:0] power enable control
		(0) PROM_ENABLE	(0) PROM_ENABLE	0x1	PROM power enable control
0x40000018	SYSCTRL_MEM_ACCESS_CFG	(29:24) WAKEUP_ADDR_PACKED	(29:24) WAKEUP_ADDR_PACKED	0x0	Wakeup restore address in packed 6-bit format. When written, SYSCTRL_WAKEUP_ADDR is updated. This

Address	Register Name	Register Write	Register Read	Default	Description
					field reads back as zero when SYSCTRL_WAKEUP_ADDR does not point to an enabled RAM instance.
		(17:16) BB_DRAM_ACCESS	(17:16) BB_DRAM_ACCESS	0x0	Baseband DRAM[1:0] access configuration
		(15:8) DRAM_ACCESS	(15:8) DRAM_ACCESS	0x1	DRAM[7:0] access configuration
		(1) FLASH_ACCESS	(1) FLASH_ACCESS	0x0	FLASH[0:0] access configuration
		(0) PROM_ACCESS	(0) PROM_ACCESS	0x1	PROM access configuration
0x4000001C	SYSCTRL_WAKEUP_ADDR	(31:0) WAKEUP_ADDR	(31:0) WAKEUP_ADDR	0x0	Wake-up restore address in unpacked 32-bit format.
0x40000020	SYSCTRL_MEM_RETENTION_CFG	(17:16) BB_DRAM_RETENTION	(17:16) BB_DRAM_RETENTION	0x3	Baseband DRAM[1:0] retention configuration
		(15:8) DRAM_RETENTION	(15:8) DRAM_RETENTION	0xFE	DRAM[7:0] retention configuration
0x40000024	SYSCTRL_MEM_TIMING_CFG	(22:20) RAM_PKA_EMA	(22:20) RAM_PKA_EMA	0x2	RAM extra margin configuration for PKA RAM
		(17:16) RAM_PKA_EMAW	(17:16) RAM_PKA_EMAW	0x0	RAM write extra margin configuration for PKA RAM
		(6:4) PROM_EMA	(6:4) PROM_EMA	0x2	PROM extra margin configuration
		(0) PROM_KEN	(0) PROM_KEN	0x1	PROM bit lines keeper configuration
0x40000028	SYSCTRL_RF_POWER_CFG	(9) BB_ENABLE	(9) BB_ENABLE	0x0	Base Band power enable control
		(8) BB_STARTUP	(8) BB_STARTUP	0x0	Base Band power startup control
		(1) RF_ENABLE	(1) RF_ENABLE	0x0	RF power enable control
		(0) RF_STARTUP	(0) RF_STARTUP	0x0	RF power startup control
0x4000002C	SYSCTRL_RF_ACCESS_CFG	(8) BB_ACCESS	(8) BB_ACCESS	0x0	Base Band access
		(1) RF_IRQ_ACCESS	(1) RF_IRQ_ACCESS	0x0	RF IRQ access configuration

Address	Register Name	Register Write	Register Read	Default	Description
		(0) RF_ACCESS	(0) RF_ACCESS	0x0	RF access configuration
0x40000038	SYSCTRL_FPU_PWR_CFG	(31:16) FPU_PWR_KEY	–	N/A	Key to enable write to this register
		(10) PWRUP_FPU_ TRICKLE	(10) PWRUP_FPU_ TRICKLE	0x1	Power control for FPU primary current in-rush limited supply
		(9) PWRUP_FPU_HAMMER	(9) PWRUP_FPU_ HAMMER	0x1	Power control for FPU primary low impedance supply
		(8) ISOLATE_FPU	(8) ISOLATE_FPU	0x0	Isolation control for FPU
		(3) FPU_Q_REQ	(3) FPU_Q_REQ	0x0	FPU domain quiescence request signal
		–	(2) FPU_Q_ACCEPT	0x0	FPU domain quiescence request accepted
		–	(1) FPU_Q_DENY	0x0	FPU domain quiescence request denied
		–	(0) FPU_Q_ACTIVE	0x1	FPU logic active or activation request
0x4000003C	SYSCTRL_DBG_PWR_CFG	(31:16) DBG_PWR_KEY	–	N/A	Key to enable write to this register
		(10) PWRUP_DBG_ TRICKLE	(10) PWRUP_DBG_ TRICKLE	0x1	Power control for Debug primary current in-rush limited supply
		(9) PWRUP_DBG_HAMMER	(9) PWRUP_DBG_ HAMMER	0x1	Power control for Debug primary low impedance supply
		(8) ISOLATE_DBG	(8) ISOLATE_DBG	0x0	Isolation control for Debug
		(3) DBG_Q_REQ	(3) DBG_Q_REQ	0x0	Debug domain quiescence request signal
		–	(2) DBG_Q_ACCEPT	0x0	Debug domain quiescence request accepted
		–	(1) DBG_Q_DENY	0x0	Debug domain quiescence request denied
		–	(0) DBG_Q_ACTIVE	0x0	Debug logic active or activation

Address	Register Name	Register Write	Register Read	Default	Description
					request
0x40000040	SYSCTRL_CRYPTOCELL_PWR_CFG	(31:16) PWR_KEY	-	N/A	Key to enable write to this register
		-	(4) CC_CG_STATE	0x0	CryptoCell central clock gating state
		(3) CC_POWER_STARTUP	(3) CC_POWER_STARTUP	0x0	Power control for CryptoCell
		(2) CC_POWER_ENABLE	(2) CC_POWER_ENABLE	0x0	Power control for CryptoCell
		(1) CC_ISOLATE_N	(1) CC_ISOLATE_N	0x0	CryptoCell isolate control signal
		-	(0) CC_POWERDOWN_RDY	0x0	Indicates that CryptoCell can be powered down
0x40000044	SYSCTRL_NS_ACCESS_PERIPH_CFG0	(28) PWM_ACCESS	(28) PWM_ACCESS	0x0	Allow Non-Secure code to access the PWM
		(27) LIN_ACCESS	(27) LIN_ACCESS	0x0	Allow Non-Secure code to access the LIN[0:0]
		(26) TOF_ACCESS	(26) TOF_ACCESS	0x0	Allow Non-Secure code to access the TOF
		(25) PCM_ACCESS	(25) PCM_ACCESS	0x0	Allow Non-Secure code to access the PCM[0:0]
		(24) UART_ACCESS	(24) UART_ACCESS	0x0	Allow Non-Secure code to access the UART[0:0]
		(23:22) I2C_ACCESS	(23:22) I2C_ACCESS	0x0	Allow Non-Secure code to access the I2C[1:0]
		(21:20) SPI_ACCESS	(21:20) SPI_ACCESS	0x0	Allow Non-Secure code to access the SPI[1:0]
		(19) GPIO_SRC_ACCESS	(19) GPIO_SRC_ACCESS	0x0	Allow Non-Secure code to access the GPIO_SRC
		(18) GPIO_ACCESS	(18) GPIO_ACCESS	0x0	Allow Non-Secure code to access the

Address	Register Name	Register Write	Register Read	Default	Description
					GPIO
		(17) CC312_ACCESS	(17) CC312_ACCESS	0x0	Allow Non-Secure code to access the CC312
		(16) ASCC_ACCESS	(16) ASCC_ACCESS	0x0	Allow Non-Secure code to access the ASCC
		(15:12) TIMER_ACCESS	(15:12) TIMER_ ACCESS	0x0	Allow Non-Secure code to access the Timer[3:0]
		(10) CRC_ACCESS	(10) CRC_ACCESS	0x0	Allow Non-Secure code to access the CRC
		(8) NMI_ACCESS	(8) NMI_ACCESS	0x0	Allow Non-Secure code to access the NMI
		(7) WATCHDOG_ACCESS	(7) WATCHDOG_ACCESS	0x0	Allow Non-Secure code to access the Watchdog
		(6) SYSCTRL_ACCESS	(6) SYSCTRL_ACCESS	0x0	Allow Non-Secure code to access the SYSCTRL
		(5) SENSOR_ACCESS	(5) SENSOR_ACCESS	0x0	Allow Non-Secure code to access the Sensor
		(4) ACS_ACCESS	(4) ACS_ACCESS	0x0	Allow Non-Secure code to access the ACS
		(3) LSAD_ACCESS	(3) LSAD_ACCESS	0x0	Allow Non-Secure code to access the LSAD config
		(2) TEST_CTRL_ACCESS	(2) TEST_CTRL_ ACCESS	0x0	Allow Non-Secure code to access the TEST_CTRL config
		(1) CLK_ACCESS	(1) CLK_ACCESS	0x0	Allow Non-Secure code to access the CLK config
		(0) RESET_ACCESS	(0) RESET_ACCESS	0x0	Allow Non-Secure code to access the Reset
0x40000048	SYSCTRL_NS_ACCESS_	(11) FLASH_IF_ACCESS	(11) FLASH_IF_	0x0	Allow Non-Secure code to access the

Address	Register Name	Register Write	Register Read	Default	Description
	PERIPH_CFG1		ACCESS		FLASH_IF[0:0]
		(10) RF_ACCESS	(10) RF_ACCESS	0x0	Allow Non-Secure code to access the RF
		(8) BB_ACCESS	(8) BB_ACCESS	0x0	Allow Non-Secure code to access the BB
		(3:0) DMA_ACCESS	(3:0) DMA_ACCESS	0x0	Allow Non-Secure code to access the DMA[3:0]
0x4000004C	SYSCTRL_NS_ACCESS_RAM_CFG0	(7:0) DRAM_ACCESS	(7:0) DRAM_ACCESS	0x0	Allow Non-Secure to access DRAM [7:0]
0x40000050	SYSCTRL_NS_ACCESS_RAM_CFG1	(1:0) BB_DRAM_ACCESS	(1:0) BB_DRAM_ACCESS	0x0	Allow Non-Secure code to access BB_DRAM[1:0]
0x4000005C	SYSCTRL_DEU_STATUS	(17) OVERFLOW_CLEAR	-	N/A	Clear OVERFLOW flag
		(16) FIRST_DAP_W_FLAG_CLEAR	-	N/A	Clear the FIRST_DAP_W_FLAG
		-	(12) FIRST_DAP_W_FLAG	0x0	First Debug access port write flag
		-	(8) OVERFLOW	0x0	High when DEU_DATA register is written before been read by any bus. Clear via CLR_OVERFLOW action bit.
		-	(4) SBUS_W	0x0	Set when SBus writes the DEU_DATA, clear when the debug access port read the DEU_DATA register
		-	(0) DAP_W	0x0	Set when the debug access port writes the DEU_DATA, clear when the SBus reads the DEU_DATA register.
0x40000060	SYSCTRL_DEU_DATA	(31:0) DEU_DATA	(31:0) DEU_DATA	0x0	Data exchange unit data

Address	Register Name	Register Write	Register Read	Default	Description
0x40000064	SYSCTRL_PROD_STATUS	(31:0) PROD_STATUS	–	N/A	Production Status
0x4000007C	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
0x40000080	SYSCTRL_SYSCLK_CNT	(31:0) SYSCLK_CNT	(31:0) SYSCLK_CNT	0x0	System clock counter value
0x40000084	SYSCTRL_CM33_CNT	(31:0) CM33_CNT	(31:0) CM33_CNT	0x0	CM33 activity counter value
0x40000088	SYSCTRL_CBUS_CNT	(31:0) CBUS_CNT	(31:0) CBUS_CNT	0x0	CBus-instruction activity counter value
0x4000008C	SYSCTRL_FLASH_READ_CNT	(31:0) FLASH_READ_CNT	(31:0) FLASH_READ_CNT	0x0	Flash read access counter value
0x40000094	SYSCTRL_CC_DCU_EN0	–	(31) CC_DCU_EN_ICV_GP	0x0	Always On ICV governed dcu_en0 general purpose bits
		–	(30:28) CC_DCU_EN_ICV_EH	0x0	Always On ICV governed energy harversting signature. Majority of the bis must be set to confirm the state.
		–	(27:25) CC_DCU_EN_ICV_PRDSTATE	0x0	Always On ICV governed production state identifier
		–	(24:22) CC_DCU_EN_ICV_TCTRL_ACC	0x0	Always On ICV governed test control configuration access control. Majority of the bits must be set to enable the feature
		–	(21:19) CC_DCU_EN_ICV_TRIM_ACC	0x0	Always On ICV governed MNVR and chip trim access control. Majority of the bits must be set to enable the feature
		–	(18:16) CC_DCU_EN_ICV_NVM_ACC	0x0	Always On ICV governed NVM access control. Majority of the bits

Address	Register Name	Register Write	Register Read	Default	Description
					must be set to enable the feature
		–	(15:13) CC_DCU_EN_ICV_SEC_RST	0x0	Always On ICV governed secure reset enable. If any of these bits are set and the part is in SE state, the cc312_top is reset
		–	(12:10) CC_DCU_EN_ICV_SPINDEN	0x0	Always On ICV governed CM33 secure non-intrusive debug enable control. Majority of the bits must be set to enable the feature
		–	(9:7) CC_DCU_EN_ICV_SPIDEN	0x0	Always On ICV governed CM33 secure intrusive debug enable control. Majority of the bits must be set to enable the feature
		–	(6:4) CC_DCU_EN_ICV_NIDEN	0x0	Always On ICV governed CM33 non-intrusive debug enable control. Majority of the bits must be set to enable the feature
		–	(3:1) CC_DCU_EN_ICV_DBGEN	0x0	Always On ICV governed CM33 debug enable control. Majority of the bits must be set to enable the feature
		–	(0) CC_DCU_EN_ICV_SEC_RST_MASK	0x0	Always On ICV governed secure reset mask used in FW
0x40000098	SYSCTRL_CC_DCU_EN1	–	(31:16) CC_DCU_EN_OEM_GP	0x0	Always On OEM governed dcu_en1 general purpose bits
		–	(15:13) CC_DCU_EN_OEM_SEC_RST	0x0	Always On OEM governed secure reset enable. If any of these bits are set and the part is in SE state, the cc312_top is reset
		–	(12:10) CC_DCU_EN_OEM_SPINDEN	0x0	Always On OEM governed CM33 secure non-intrusive debug enable

Address	Register Name	Register Write	Register Read	Default	Description
					control. Majority of the bits must be set to enable the feature
		–	(9:7) CC_DCU_EN_OEM_SPIDEN	0x0	Always On OEM governed CM33 secure intrusive debug enable control. Majority of the bits must be set to enable the feature
		–	(6:4) CC_DCU_EN_OEM_NIDEN	0x0	Always On OEM governed CM33 non-intrusive debug enable control. Majority of the bits must be set to enable the feature
		–	(3:1) CC_DCU_EN_OEM_DBGEN	0x0	Always On OEM governed CM33 debug enable control. Majority of the bits must be set to enable the feature
		–	(0) CC_DCU_EN_OEM_RESERVED	0x0	Always On OEM allocated reserved bit (unused)
0x4000009C	SYSCTRL_CC_DCU_EN2	–	(31:0) CC_DCU_EN2	0x0	Always On block DCU_EN2 state
0x400000A0	SYSCTRL_CC_DCU_EN3	–	(31:0) CC_DCU_EN3	0x0	Always On block DCU_EN3 state
0x400000A4	SYSCTRL_CC_DCU_LOCK0	–	(31) CC_DCU_LOCK_ICV_GP	0x0	Always On dcu_en0 general purpose bits lock. All bits must be locked to assure that the state is locked
		–	(30:28) CC_DCU_LOCK_ICV_EH	0x0	Always On energy harvesting signature lock. All bits must be locked to assure that the state is locked
		–	(27:25) CC_DCU_LOCK_ICV_PRDSTATE	0x0	Always On production state identifier lock. All bits must be locked to assure that the state is locked
		–	(24:22) CC_DCU_LOCK_ICV_TCTRL_ACC	0x0	Always On test control configuration access control lock. All bits must be locked to assure that the state is locked

Address	Register Name	Register Write	Register Read	Default	Description
		–	(21:19) CC_DCU_ LOCK_ICV_TRIM_ACC	0x0	Always On MNVR and chip trim access control lock. All bits must be locked to assure that the state is locked
		–	(18:16) CC_DCU_ LOCK_ICV_NVM_ACC	0x0	Always On NVM access control lock. All bits must be locked to assure that the state is locked
		–	(15:13) CC_DCU_ LOCK_ICV_SEC_RST	0x0	Always On ICV secure reset enable lock. All bits must be locked to assure that the state is locked
		–	(12:10) CC_DCU_ LOCK_ICV_SPINDEN	0x0	Always On ICV CM33 secure non-intrusive debug enable control lock. All bits must be locked to assure that the state is locked
		–	(9:7) CC_DCU_LOCK_ ICV_SPIDEN	0x0	Always On ICV CM33 secure intrusive debug enable control lock. All bits must be locked to assure that the state is locked
		–	(6:4) CC_DCU_LOCK_ ICV_NIDEN	0x0	Always On ICV CM33 non-intrusive debug enable control lock. All bits must be locked to assure that the state is locked
		–	(3:1) CC_DCU_LOCK_ ICV_DBGEN	0x0	Always On ICV CM33 debug enable control lock. All bits must be locked to assure that the state is locked
		–	(0) CC_DCU_LOCK_ ICV_SEC_RST_MASK	0x0	Always On ICV secure reset mask lock
0x400000A8	SYSCTRL_CC_DCU_LOCK1	–	(31:16) CC_DCU_ LOCK_OEM_GP	0x0	Always On OEM governed dcu_en1 general purpose bits lock
		–	(15:13) CC_DCU_	0x0	Always On OEM governed secure

Address	Register Name	Register Write	Register Read	Default	Description
			LOCK_OEM_SEC_RST		reset lock. All bits must be locked to assure that the state is locked
		–	(12:10) CC_DCU_ LOCK_OEM_SPINDEN	0x0	Always On OEM governed CM33 secure non-intrusive debug enable control lock. All bits must be locked to assure that the state is locked
		–	(9:7) CC_DCU_LOCK_ OEM_SPIDEN	0x0	Always On OEM governed CM33 secure intrusive debug enable control lock. All bits must be locked to assure that the state is locked
		–	(6:4) CC_DCU_LOCK_ OEM_NIDEN	0x0	Always On OEM governed CM33 non-intrusive debug enable control lock. All bits must be locked to assure that the state is locked
		–	(3:1) CC_DCU_LOCK_ OEM_DBGEN	0x0	Always On OEM governed CM33 debug enable control lock. All bits must be locked to assure that the state is locked
		–	(0) CC_DCU_LOCK_ OEM_RESERVED	0x0	Always On OEM allocated reserved bit (unused)
0x400000AC	SYSCTRL_CC_DCU_LOCK2	–	(31:0) CC_DCU_LOCK2	0x0	Always On block DCU_LOCK2 state
0x400000B0	SYSCTRL_CC_DCU_LOCK3	–	(31:0) CC_DCU_LOCK3	0x0	Always On block DCU_LOCK3 state
0x400000B4	SYSCTRL_CC_STATUS	–	(25) CC_SEC_DEBUG_ RESET	0x0	Always On block secure debug reset status
		–	(24) CC_HOST_DFA_ ENABLE_LOCK	0x0	Always On block host AES DFA lock status
		–	(23) CC_HOST_FORCE_ DFA_ENABLE	0x0	Always On block host AES DFA status
		–	(22) CC_RESET_UPON_	0x0	Always On block HUK reset

Address	Register Name	Register Write	Register Read	Default	Description
			DEBUG_DISABLE		mechanism configuration status
		–	(21) CC_HOST_ICV_RMA_LOCK	0x0	Always On block host icv rma bit in NVM lock status
		–	(20) CC_HOST_KCE_LOCK	0x0	Always On block host OEM code encryption key lock status
		–	(19) CC_HOST_KCP_LOCK	0x0	Always On block host OEM provisioning key lock status
		–	(18) CC_HOST_KCEICV_LOCK	0x0	Always On block host ICV code encryption key lock status
		–	(17) CC_HOST_KPICV_LOCK	0x0	Always On block host ICV provisioning key lock status
		–	(16) CC_HOST_FATAL_ERR	0x0	Always On block host fatal error flag
		–	(15) CC_APB_ONLY_PRIV_ACCESS_LOCK	0x0	Always On block APB filtering privileged access configuration lock
		–	(14) CC_APB_ONLY_PRIV_ACCESS	0x0	Always On block APB filtering privileged access configuration
		–	(13) CC_APB_ONLY_SEC_ACCESS_LOCK	0x0	Always On block APB filtering secure access configuration lock
		–	(12) CC_APB_ONLY_SEC_ACCESS	0x0	Always On block APB filtering secure access configuration
		–	(11:4) CC_GPPC	0x0	Always On block GPPC register
		–	(3) CC_LCS_VALID	0x0	Always On block life cycle state valid
		–	(2:0) CC_LCS	0x0	Always On block life cycle state
0x400000B8	SYSCTRL_CC_FEATURES_CTRL	–	(30) CC_OEM_SEC_RST_ALL	0x0	Always On block OEM governed DCU_EN bits fault status allocated for enabling SEC_RST operation

Address	Register Name	Register Write	Register Read	Default	Description
		–	(29) CC_OEM_SPINDEN_ALL	0x0	Always On block OEM governed DCU_EN bits fault status allocated for enabling SPINDEN operation
		–	(28) CC_OEM_SPIDEN_ALL	0x0	Always On block OEM governed DCU_EN bits fault status allocated for enabling SPIDEN operation
		–	(27) CC_OEM_NIDEN_ALL	0x0	Always On block OEM governed DCU_EN bits fault status allocated for enabling NIDEN operation
		–	(26) CC_OEM_DBGEN_ALL	0x0	Always On block OEM governed DCU_EN bits fault status allocated for enabling DBGEN operation
		–	(25) CC_ENERGY_HARVESTING_ALL	0x0	Always On block DCU_EN bits fault status allocated for enabling ENERGY_HARVESTING operation
		–	(24) CC_PROD_STATUS_ALL	0x0	Always On block DCU_EN bits fault status allocated for enabling PROD_STATUS operation
		–	(23) CC_TCTRL_ACC_ALL	0x0	Always On block DCU_EN bits fault status allocated for enabling TCTRL_ACC operation
		–	(22) CC_TRIM_ACC_ALL	0x0	Always On block DCU_EN bits fault status allocated for enabling TRIM_ACC operation
		–	(21) CC_NVM_ACC_ALL	0x0	Always On block DCU_EN bits fault status allocated for enabling NVM_ACC operation
		–	(20) CC_ICV_SEC_RST_ALL	0x0	Always On block ICV governed DCU_EN bits fault status allocated for enabling SEC_RST operation

Address	Register Name	Register Write	Register Read	Default	Description
		–	(19) CC_ICV_ SPINDEN_ALL	0x0	Always On block ICV governed DCU_ EN bits fault status allocated for enabling SPINDEN operation
		–	(18) CC_ICV_SPIDEN_ ALL	0x0	Always On block ICV governed DCU_ EN bits fault status allocated for enabling SPIDEN operation
		–	(17) CC_ICV_NIDEN_ ALL	0x0	Always On block ICV governed DCU_ EN bits fault status allocated for enabling NIDEN operation
		–	(16) CC_ICV_DBGEN_ ALL	0x0	Always On block ICV governed DCU_ EN bits fault status allocated for enabling DBGEN operation
		–	(13) CC_OEM_SPINDEN	0x0	Always On block OEM SPINDEN status as the result of the equality check and production state confirmation
		–	(12) CC_OEM_SPIDEN	0x0	Always On block OEM SPIDEN status as the result of the equality check and production state confirmation
		–	(11) CC_OEM_NIDEN	0x0	Always On block OEM NIDEN status as the result of the equality check and production state confirmation
		–	(10) CC_OEM_DBGEN	0x0	Always On block OEM DBGEN status as the result of the equality check and production state confirmation
		–	(9) CC_ENERGY_ HARVESTING	0x0	Always On block Energy Harvesting status as the result of equality check
		–	(8) CC_PROD_STATUS	0x0	Always On block Production Status as the result of equality check
		–	(7) CC_TCTRL_ACC	0x0	Always On block TCTRL_ACC status

Address	Register Name	Register Write	Register Read	Default	Description
					as the result of the equality check and production state confirmation
		–	(6) CC_TRIM_ACC	0x0	Always On block TRIM_ACC status as the result of the equality check and production state confirmation
		–	(5) CC_NVM_ACC	0x0	Always On block NVM_ACC status as the result of the equality check and production state confirmation
		–	(4) CC_SEC_RST	0x0	Always On block ICV or OEM SEC_ RST status as the result of any bit being set.
		–	(3) CC_ICV_SPINDEN	0x0	Always On block ICV SPINDEN status as the result of the equality check and production state confirmation
		–	(2) CC_ICV_SPIDEN	0x0	Always On block ICV SPIDEN status as the result of the equality check and production state confirmation
		–	(1) CC_ICV_NIDEN	0x0	Always On block ICV NIDEN status as the result of the equality check and production state confirmation
		–	(0) CC_ICV_DBGEN	0x0	Always On block ICV DBGEN status as the result of the equality check and production state confirmation
0x400000D0	SYSCTRL_VDDPA_CFG0	(31:24) DISABLE_DELAY	(31:24) DISABLE_ DELAY	0x0	VDDPA disable delay after ramp down (in system clock cycles)
		(23:16) RAMPUP_DELAY	(23:16) RAMPUP_ DELAY	0x0	VDDPA ramp up delay after regulator enabling (in system clock cycles)
		(15:8) SW_CTRL_DELAY	(15:8) SW_CTRL_ DELAY	0x0	VDDPA SW_CTRL delay after regulator enabling (in system clock

Address	Register Name	Register Write	Register Read	Default	Description
					cycles)
		(0) DYNAMIC_CTRL	(0) DYNAMIC_CTRL	0x0	Dynamic VDDPA control (VDDPA is enabled when RF is in TX mode, disabled otherwise)
0x400000D4	SYSCTRL_VDDPA_CFG1	–	(31) ACTUAL_SW_CTRL	0x0	VDDPA actual SW_CTRL status
		–	(30) ACTUAL_ENABLE	0x0	VDDPA actual enable status
		–	(29:24) ACTUAL_ VTRIM	0x0	VDDPA actual voltage trimming status
		(23:16) RAMPDOWN_ STEP_TIME	(23:16) RAMPDOWN_ STEP_TIME	0x0	VDDPA ramp down step duration (in system clock cycles)
		(15:8) RAMPUP_STEP_ TIME	(15:8) RAMPUP_STEP_ TIME	0x0	VDDPA ramp up step duration (in system clock cycles)
		(6:4) RAMPDOWN_STEP	(6:4) RAMPDOWN_STEP	0x0	Amount of VDDPA ramp down step(s)
		(2:0) RAMPUP_STEP	(2:0) RAMPUP_STEP	0x0	Amount of VDDPA ramp up step(s)
0x400000D8	SYSCTRL_RFIF_CTRL	(13) IQ_CLEAR	–	N/A	Clear IQ data register
		(12) PHASE_ADC_CLEAR	–	N/A	Clear phase ADC registers
		(11) PACKET_END_CLEAR	–	N/A	Clear packet end register
		(10) CTE_MODE_CLEAR	–	N/A	Clear CTE mode register
		(9) SYNC_WORD_CLEAR	–	N/A	Clear sync world register
		(8) COUNTER_CLEAR	–	N/A	Clear counter
		(1) IQ_STREAMING	(1) IQ_STREAMING	0x0	Enable I/Q data streaming
		(0) PHASE_ADC_ STREAMING	(0) PHASE_ADC_ STREAMING	0x0	Enable phase ADC data streaming
0x400000DC	SYSCTRL_RFIF_IQ	–	(15:8) Q	0x0	Quadrature data
		–	(7:0) I	0x0	In-phase data

Address	Register Name	Register Write	Register Read	Default	Description
0x400000E0	SYSCTRL_RFIF_PHASE_ADC	–	(28:24) RSSI_1	0x0	RSSI data 1
		–	(20:16) RSSI_0	0x0	RSSI data 0
		–	(15:12) AGC_1	0x0	AGC data 1
		–	(11:8) AGC_0	0x0	AGC data 0
		–	(7:4) ADC_1	0x0	Phase ADC data 1
		–	(3:0) ADC_0	0x0	Phase ADC data 0
0x400000E4	SYSCTRL_RFIF_PHASE_ADC_FLAGS	–	(31) PACKET_END_FLAG_1	0x0	Packet end flag 1
		–	(30) CTE_MODE_FLAG_1	0x0	CTE mode flag 1
		–	(29) SYNC_WORD_FLAG_1	0x0	Sync word flag 1
		–	(28:24) RSSI_1	0x0	RSSI data 1
		–	(23) PACKET_END_FLAG_0	0x0	Packet end flag 0
		–	(22) CTE_MODE_FLAG_0	0x0	CTE mode flag 0
		–	(21) SYNC_WORD_FLAG_0	0x0	Sync word flag 0
		–	(20:16) RSSI_0	0x0	RSSI data 0
		–	(15:12) AGC_1	0x0	AGC data 1
		–	(11:8) AGC_0	0x0	AGC data 0
		–	(7:4) ADC_1	0x0	Phase ADC data 1
		–	(3:0) ADC_0	0x0	Phase ADC data 0
0x400000E8	SYSCTRL_RFIF_PHASE_ADC_0	–	(26) PACKET_END_FLAG_0	0x0	Packet end flag 0

Address	Register Name	Register Write	Register Read	Default	Description
		–	(25) CTE_MODE_FLAG_0	0x0	CTE mode flag 0
		–	(24) SYNC_WORD_FLAG_0	0x0	Sync word flag 0
		–	(20:16) RSSI_0	0x0	RSSI data 0
		–	(11:8) AGC_0	0x0	AGC data 0
		–	(3:0) ADC_0	0x0	Phase ADC data 0
0x400000EC	SYSCTRL_RFIF_PHASE_ADC_1	–	(26) PACKET_END_FLAG_1	0x0	Packet end flag 1
		–	(25) CTE_MODE_FLAG_1	0x0	CTE mode flag 1
		–	(24) SYNC_WORD_FLAG_1	0x0	Sync word flag 1
		–	(20:16) RSSI_1	0x0	RSSI data 1
		–	(11:8) AGC_1	0x0	AGC data 1
		–	(3:0) ADC_1	0x0	Phase ADC data 1
0x400000F0	SYSCTRL_RFIF_COUNTER	–	(15:0) STATE	0xFFFF	Current data counter state
0x400000F4	SYSCTRL_RFIF_SYNC_WORD	–	(15:0) COUNTER_STATE	0xFFFF	Data counter state coinciding with the sync word detection
0x400000F8	SYSCTRL_RFIF_CTE_MODE	–	(15:0) COUNTER_STATE	0xFFFF	Data counter state coinciding with the CTE mode detection
0x400000FC	SYSCTRL_RFIF_PACKET_END	–	(15:0) COUNTER_STATE	0xFFFF	Data counter state coinciding with the paket end detection

A.2 Clock Generation

Address	Register Name	Register Write	Register Read	Default	Description
0x40000100	CLK_SYS_CFG	(19:16) EXTCLK_PRESCALE	(19:16) EXTCLK_PRESCALE	0x0	Prescale value for the input clock EXTCLK (1 to 16 in steps of 1)
		(11:8) SWCLK_PRESCALE	(11:8) SWCLK_PRESCALE	0x0	Prescale value for the input clock from pad SWCLK (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 : SWCLK, RFCLK, RCCLK, or STANDBYCLK
0x40000104	CLK_DIV_CFG0	(20:16) UARTCLK_PRESCALE	(20:16) UARTCLK_PRESCALE	0x0	Prescale value for the UART peripheral clock (1 to 32 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	(27) SENSOR_CLK_DISABLE	(27) SENSOR_CLK_DISABLE	0x1	Sensor clock disable
		(26:16) SENSOR_CLK_PRESCALE	(26:16) SENSOR_CLK_PRESCALE	0x0	Prescale value for the sensor clock
		(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)

Address	Register Name	Register Write	Register Read	Default	Description
0x4000010C	CLK_DIV_CFG2	(20) PWM_CLK_SRC	(20) PWM_CLK_SRC	0x0	PWM clock source
		(16) USRCLK_SRC_SEL	(16) USRCLK_SRC_SEL	0x0	USR clock source selection
		(11:0) USRCLK_ PRESCALE	(11:0) USRCLK_ PRESCALE	0x0	Prescale value for the USR clock (1 to 4096 in steps of 1)

A.3 RESET

Address	Register Name	Register Write	Register Read	Default	Description
0x40000200	RESET_DIG_STATUS	(12) DEU_RESET_FLAG_CLEAR	-	N/A	Reset the sticky DEU reset flag
		(11) LOCKUP_RESET_FLAG_CLEAR	-	N/A	Reset the sticky LOCKUP flag
		(10) WATCHDOG_RESET_FLAG_CLEAR	-	N/A	Reset the sticky Watchdog time-out reset flag
		(9) CM33_SW_RESET_FLAG_CLEAR	-	N/A	Reset the sticky CM33 software reset flag
		(8) ACS_RESET_FLAG_CLEAR	-	N/A	Reset the sticky ACS reset flag
		-	(4) DEU_RESET_FLAG	0x0	Sticky flag that detect that a DEU reset occured
		-	(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) CM33_SW_RESET_FLAG	0x0	Sticky flag that detects that a CM33 software reset occurred
		-	(0) ACS_RESET_FLAG	0x1	Sticky flag that detects that a ACS reset occurred

A.4 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
0x400003FC	WATCHDOG_ID_NUM	–	(15:8) WATCHDOG_ MAJOR_REVISION	0x1	WATCHDOG Major Revision number
		–	(7:0) WATCHDOG_ MINOR_REVISION	0x0	WATCHDOG Minor Revision number

A.5 General-Purpose Timers

Address	Register Name	Register Write	Register Read	Default	Description
0x40000400	TIMER0_CFG0	(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
0x40000404	TIMER0_CFG1	(12) CLK_SRC	(12) CLK_SRC	0x0	Clock source
		(10:8) MULTI_COUNT	(10:8) MULTI_COUNT	0x0	Multi-count value
		(6:4) GPIO_INT_SRC	(6:4) GPIO_INT_SRC	0x0	GPIO interrupt source selection
		(2) GPIO_INT_MODE	(2) GPIO_INT_MODE	0x0	GPIO interrupt capture mode
		(1) GPIO_INT_ENABLE	(1) GPIO_INT_ENABLE	0x0	GPIO interrupt capture enable
		(0) MODE	(0) MODE	0x0	Timer mode
0x40000408	TIMER0_CTRL	–	(8) BUSY	0x0	Indicate if the timer is active or not
		(1) START	–	N/A	Start or restart the timer
		(0) STOP	–	N/A	Stop the timer
0x4000040C	TIMER0_VAL	–	(26:24) MULTI_COUNT_VAL	0x0	Current multi counter value
		–	(23:0) TIMER_VALUE	0x0	Current timer value
0x40000410	TIMER0_VAL_CAPTURE	(26:24) MULTI_COUNT_VAL_GPIO	(26:24) MULTI_COUNT_VAL_GPIO	0x0	Current multi counter value captured by the GPIO interrupt defined in GPIO_INT_SRC
		(23:0) TIMER_VALUE_GPIO	(23:0) TIMER_VALUE_GPIO	0x0	Timer value captured by the GPIO interrupt defined in GPIO_INT_SRC
0x400004FC	TIMER0_ID_NUM	–	(20) TIMER_GPIO_CAP	0x0	Implementation of the GPIO triggered capture
		–	(19:16) TIMER_ID_NUM	0x0	TIMER instance number

Address	Register Name	Register Write	Register Read	Default	Description
		–	(15:8) TIMER_MAJOR_REVISION	0x1	TIMER Major Revision number
		–	(7:0) TIMER_MINOR_REVISION	0x0	TIMER Minor Revision number
0x40000500	TIMER1_CFG0	(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
0x40000504	TIMER1_CFG1	(12) CLK_SRC	(12) CLK_SRC	0x0	Clock source
		(10:8) MULTI_COUNT	(10:8) MULTI_COUNT	0x0	Multi-count value
		(6:4) GPIO_INT_SRC	(6:4) GPIO_INT_SRC	0x0	GPIO interrupt source selection
		(2) GPIO_INT_MODE	(2) GPIO_INT_MODE	0x0	GPIO interrupt capture mode
		(1) GPIO_INT_ENABLE	(1) GPIO_INT_ENABLE	0x0	GPIO interrupt capture enable
		(0) MODE	(0) MODE	0x0	Timer mode
0x40000508	TIMER1_CTRL	–	(8) BUSY	0x0	Indicate if the timer is active or not
		(1) START	–	N/A	Start or restart the timer
		(0) STOP	–	N/A	Stop the timer
0x4000050C	TIMER1_VAL	–	(26:24) MULTI_COUNT_VAL	0x0	Current multi counter value
		–	(23:0) TIMER_VALUE	0x0	Current timer value
0x40000510	TIMER1_VAL_CAPTURE	(26:24) MULTI_COUNT_VAL_GPIO	(26:24) MULTI_COUNT_VAL_GPIO	0x0	Current multi counter value captured by the GPIO interrupt defined in GPIO_INT_SRC
		(23:0) TIMER_VALUE_GPIO	(23:0) TIMER_VALUE_GPIO	0x0	Timer value captured by the GPIO interrupt defined in GPIO_INT_SRC
0x400005FC	TIMER1_ID_NUM	–	(20) TIMER_GPIO_CAP	0x0	Implementation of the GPIO triggered capture

Address	Register Name	Register Write	Register Read	Default	Description
		-	(19:16) TIMER_ID_NUM	0x0	TIMER instance number
		-	(15:8) TIMER_MAJOR_REVISION	0x1	TIMER Major Revision number
		-	(7:0) TIMER_MINOR_REVISION	0x0	TIMER Minor Revision number
0x40000600	TIMER2_CFG0	(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
0x40000604	TIMER2_CFG1	(12) CLK_SRC	(12) CLK_SRC	0x0	Clock source
		(10:8) MULTI_COUNT	(10:8) MULTI_COUNT	0x0	Multi-count value
		(6:4) GPIO_INT_SRC	(6:4) GPIO_INT_SRC	0x0	GPIO interrupt source selection
		(2) GPIO_INT_MODE	(2) GPIO_INT_MODE	0x0	GPIO interrupt capture mode
		(1) GPIO_INT_ENABLE	(1) GPIO_INT_ENABLE	0x0	GPIO interrupt capture enable
		(0) MODE	(0) MODE	0x0	Timer mode
0x40000608	TIMER2_CTRL	-	(8) BUSY	0x0	Indicate if the timer is active or not
		(1) START	-	N/A	Start or restart the timer
		(0) STOP	-	N/A	Stop the timer
0x4000060C	TIMER2_VAL	-	(26:24) MULTI_COUNT_VAL	0x0	Current multi counter value
		-	(23:0) TIMER_VALUE	0x0	Current timer value
0x40000610	TIMER2_VAL_CAPTURE	(26:24) MULTI_COUNT_VAL_GPIO	(26:24) MULTI_COUNT_VAL_GPIO	0x0	Current multi counter value captured by the GPIO interrupt defined in GPIO_INT_SRC
		(23:0) TIMER_VALUE_GPIO	(23:0) TIMER_VALUE_GPIO	0x0	Timer value captured by the GPIO interrupt defined in GPIO_INT_SRC

Address	Register Name	Register Write	Register Read	Default	Description
0x400006FC	TIMER2_ID_NUM	–	(20) TIMER_GPIO_CAP	0x0	Implementation of the GPIO triggered capture
		–	(19:16) TIMER_ID_NUM	0x0	TIMER instance number
		–	(15:8) TIMER_MAJOR_REVISION	0x1	TIMER Major Revision number
		–	(7:0) TIMER_MINOR_REVISION	0x0	TIMER Minor Revision number
0x40000700	TIMER3_CFG0	(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
0x40000704	TIMER3_CFG1	(12) CLK_SRC	(12) CLK_SRC	0x0	Clock source
		(10:8) MULTI_COUNT	(10:8) MULTI_COUNT	0x0	Multi-count value
		(6:4) GPIO_INT_SRC	(6:4) GPIO_INT_SRC	0x0	GPIO interrupt source selection
		(2) GPIO_INT_MODE	(2) GPIO_INT_MODE	0x0	GPIO interrupt capture mode
		(1) GPIO_INT_ENABLE	(1) GPIO_INT_ENABLE	0x0	GPIO interrupt capture enable
		(0) MODE	(0) MODE	0x0	Timer mode
0x40000708	TIMER3_CTRL	–	(8) BUSY	0x0	Indicate if the timer is active or not
		(1) START	–	N/A	Start or restart the timer
		(0) STOP	–	N/A	Stop the timer
0x4000070C	TIMER3_VAL	–	(26:24) MULTI_COUNT_VAL	0x0	Current multi counter value
		–	(23:0) TIMER_VALUE	0x0	Current timer value
0x40000710	TIMER3_VAL_CAPTURE	(26:24) MULTI_COUNT_VAL_GPIO	(26:24) MULTI_COUNT_VAL_GPIO	0x0	Current multi counter value captured by the GPIO interrupt defined in GPIO_INT_SRC

Address	Register Name	Register Write	Register Read	Default	Description
		(23:0) TIMER_VALUE_GPIO	(23:0) TIMER_VALUE_GPIO	0x0	Timer value captured by the GPIO interrupt defined in GPIO_INT_SRC
0x400007FC	TIMER3_ID_NUM	-	(20) TIMER_GPIO_CAP	0x0	Implementation of the GPIO triggered capture
		-	(19:16) TIMER_ID_NUM	0x0	TIMER instance number
		-	(15:8) TIMER_MAJOR_REVISION	0x1	TIMER Major Revision number
		-	(7:0) TIMER_MINOR_REVISION	0x0	TIMER Minor Revision number

A.6 FLASH INTERFACE

Address	Register Name	Register Write	Register Read	Default	Description
0x40000800	FLASH0_IF_CTRL	(16) NOT_LOAD_AUTO	(16) NOT_LOAD_AUTO	0x0	Do not automatically load the configuration registers and the patch information from MNVR 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.
		(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 Flash 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.
0x40000804	FLASH0_MAIN_WRITE_UNLOCK	(31:0) UNLOCK_KEY	-	N/A	32-bit key to allow for write accesses into the Flash MAIN Block
0x40000808	FLASH0_MAIN_CTRL	(11) DATA_A_35K_TO_40K_W_EN	(11) DATA_A_35K_TO_40K_W_EN	0x0	Authorize the write access to the Flash MAIN Data array from 35K to 40K word block through the FLASH_IF registers.
		(10) DATA_A_30K_TO_35K_W_EN	(10) DATA_A_30K_TO_35K_W_EN	0x0	Authorize the write access to the Flash MAIN Data array from 30K to 35K word block through the FLASH_IF registers.

Address	Register Name	Register Write	Register Read	Default	Description
		(9) DATA_A_25K_TO_30K_W_EN	(9) DATA_A_25K_TO_30K_W_EN	0x0	Authorize the write access to the Flash MAIN Data array from 25K to 30K word block through the FLASH_IF registers.
		(8) DATA_A_20K_TO_25K_W_EN	(8) DATA_A_20K_TO_25K_W_EN	0x0	Authorize the write access to the Flash MAIN Data array from 20K to 25K word block through the FLASH_IF registers.
		(7) DATA_A_15K_TO_20K_W_EN	(7) DATA_A_15K_TO_20K_W_EN	0x0	Authorize the write access to the Flash MAIN Data array from 15K to 20K word block through the FLASH_IF registers.
		(6) DATA_A_10K_TO_15K_W_EN	(6) DATA_A_10K_TO_15K_W_EN	0x0	Authorize the write access to the Flash MAIN Data array from 10K to 15K word block through the FLASH_IF registers.
		(5) DATA_A_5K_TO_10K_W_EN	(5) DATA_A_5K_TO_10K_W_EN	0x0	Authorize the write access to the Flash MAIN Data array from 5K to 10K word block through the FLASH_IF registers.
		(4) DATA_A_0K_TO_5K_W_EN	(4) DATA_A_0K_TO_5K_W_EN	0x0	Authorize the write access to the Flash MAIN Data array from 0 to 5K word block through the FLASH_IF registers.
		(3) CODE_A_66K_TO_88K_W_EN	(3) CODE_A_66K_TO_88K_W_EN	0x0	Authorize the write access to the Flash MAIN Code array from 66K to 88K word block through the FLASH_IF registers.
		(2) CODE_A_44K_TO_66K_W_EN	(2) CODE_A_44K_TO_66K_W_EN	0x0	Authorize the write access to the Flash MAIN Code array from 44K to 66K word block through the FLASH_IF registers.
		(1) CODE_A_22K_TO_44K_W_EN	(1) CODE_A_22K_TO_44K_W_EN	0x0	Authorize the write access to the Flash MAIN Code array from 22K to 44K word block through the FLASH_IF registers.

Address	Register Name	Register Write	Register Read	Default	Description
		(0) CODE_A_0K_TO_22K_W_EN	(0) CODE_A_0K_TO_22K_W_EN	0x0	Authorize the write access to the Flash MAIN Code array from 0 to 22K word block through the FLASH_IF registers.
0x4000080C	FLASH0_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
0x40000830	FLASH0_CMD_CTRL	(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
0x40000834	FLASH0_IF_STATUS	–	(31) TRIMMED_STATUS	0x0	Flash trimming status
		–	(30) ISOLATE_STATUS	0x1	Flash isolate status
		–	(29) PROG_SEQ_DATA_REQ	0x0	Request new data while in sequential program mode
		–	(28) BUSY	0x0	Flash interface busy status bit
		–	(27) DATA_RED2_W_UNLOCK	0x0	Flash Data array RED2 write unlock status bit
		–	(26) DATA_RED1_W_UNLOCK	0x0	Flash Data array RED1 write unlock status bit
		–	(25) CODE_RED2_W_UNLOCK	0x0	Flash Code array RED2 write unlock status bit
		–	(24) CODE_RED1_W_UNLOCK	0x0	Flash Code array RED1 write unlock status bit
		–	(22) NVR7_W_UNLOCK	0x0	Flash NVR7 write unlock status bit
		–	(21) NVR6_W_UNLOCK	0x0	Flash NVR6 write unlock status bit
		–	(20) NVR5_W_UNLOCK	0x0	Flash NVR5 write unlock status bit

Address	Register Name	Register Write	Register Read	Default	Description
		–	(19) NVR4_W_UNLOCK	0x0	Flash NVR4 write unlock status bit
		–	(18) NVR3_W_UNLOCK	0x0	Flash NVR3 write unlock status bit
		–	(17) NVR2_W_UNLOCK	0x0	Flash NVR2 write unlock status bit
		–	(16) NVR1_W_UNLOCK	0x0	Flash NVR1 write unlock status bit
		–	(15) NVR0_W_UNLOCK	0x0	Flash NVR0 write unlock status bit
		–	(11) DATA_A_35K_TO_40K_W_UNLOCK	0x0	Write unlock status bit of the part 35K to 40K of the Flash MAIN Data array
		–	(10) DATA_A_30K_TO_35K_W_UNLOCK	0x0	Write unlock status bit of the part 30K to 35K of the Flash MAIN Data array
		–	(9) DATA_A_25K_TO_30K_W_UNLOCK	0x0	Write unlock status bit of the part 25K to 30K of the Flash MAIN Data array
		–	(8) DATA_A_20K_TO_25K_W_UNLOCK	0x0	Write unlock status bit of the part 20K to 25K of the Flash MAIN Data array
		–	(7) DATA_A_15K_TO_20K_W_UNLOCK	0x0	Write unlock status bit of the part 15K to 20K of the Flash MAIN Data array
		–	(6) DATA_A_10K_TO_15K_W_UNLOCK	0x0	Write unlock status bit of the part 10K to 15K of the Flash MAIN Data array
		–	(5) DATA_A_5K_TO_10K_W_UNLOCK	0x0	Write unlock status bit of the part 5K to 10K of the Flash MAIN Data array
		–	(4) DATA_A_0K_TO_5K_W_UNLOCK	0x0	Write unlock status bit of the part 0K to 5K of the Flash MAIN Data array
		–	(3) CODE_A_66K_TO_88K_W_UNLOCK	0x0	Write unlock status bit of the part 66K to 88K of the Flash MAIN Code array
		–	(2) CODE_A_44K_TO_66K_W_UNLOCK	0x0	Write unlock status bit of the part 44K to 66K of the Flash MAIN Code array
		–	(1) CODE_A_22K_TO_44K_W_UNLOCK	0x0	Write unlock status bit of the part 22K to 44K of the Flash MAIN Code array

Address	Register Name	Register Write	Register Read	Default	Description
		–	(0) CODE_A_0K_TO_22K_W_UNLOCK	0x0	Write unlock status bit of the part 0K to 22K of the Flash MAIN Code array
0x40000838	FLASH0_ADDR	(20:2) FLASH_ADDR	(20:2) FLASH_ADDR	0x0	Flash Byte Address
0x4000083C – 0x40000840	FLASH0_DATA_*	(31:0) DATA	(31:0) DATA	0x0	32-bit Flash Data
0x40000844	FLASH0_NVR_WRITE_UNLOCK	(31:0) UNLOCK_KEY	–	N/A	32-bit key to allow for write access to NVR sectors of the Flash
0x40000848	FLASH0_NVR_CTRL	(7) NVR7_W_EN	(7) NVR7_W_EN	0x0	Authorize write access to the Flash NVR7 sector through the FLASH_IF registers.
		(6) NVR6_W_EN	(6) NVR6_W_EN	0x0	Authorize write access to the Flash NVR6 sector through the FLASH_IF registers.
		(5) NVR5_W_EN	(5) NVR5_W_EN	0x0	Authorize write access to the Flash NVR5 sector through the FLASH_IF registers.
		(4) NVR4_W_EN	(4) NVR4_W_EN	0x0	Authorize write access to the Flash NVR4 sector through the FLASH_IF registers.
		(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.
		(0) NVR0_W_EN	(0) NVR0_W_EN	0x0	Authorize write access to the Flash

Address	Register Name	Register Write	Register Read	Default	Description
					NVR0 sector through the FLASH_IF registers.
0x40000864 - 0x40000880	FLASH0_PATCH_ADDR_*	(31) PATCH_NOT_VALID	(31) PATCH_NOT_ VALID	0x1	Indicates if the patch address is valid
		(20:8) PATCH_ADDR	(20:8) PATCH_ADDR	0x1FFF	
0x4000088C	FLASH0_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)
		(1) PRIORITY	(1) PRIORITY	0x0	Copier Priority Configuration
		(0) MODE	(0) MODE	0x0	Copier or Comparator Mode Configuration
0x40000898	FLASH0_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
0x4000089C	FLASH0_COPY_SRC_ADDR_ PTR	(20:0) COPY_SRC_ADDR_ PTR	(20:0) COPY_SRC_ ADDR_PTR	0x0	Source address pointer
0x400008A0	FLASH0_COPY_DST_ADDR_ PTR	(31:2) COPY_DST_ADDR_ PTR	(31:2) COPY_DST_ ADDR_PTR	0x0	Destination address pointer
0x400008A4	FLASH0_COPY_WORD_CNT	(16:0) COPY_WORD_CNT	(16:0) COPY_WORD_ CNT	0x0	Number of words to copy / compare

Address	Register Name	Register Write	Register Read	Default	Description
0x400008A8	FLASH0_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 CM33 interrupt
		(3) COPIER_ECC_CTRL	(3) COPIER_ECC_CTRL	0x1	
		(2) CMD_ECC_CTRL	(2) CMD_ECC_CTRL	0x1	
		(0) CBUS_ECC_CTRL	(0) CBUS_ECC_CTRL	0x1	Select the operating mode of the Flash ECC
0x400008AC	FLASH0_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
0x400008B0	FLASH0_ECC_ERROR_ADDR	–	(20:2) ECC_ERROR_ADDR	0x0	Store the Flash address of the latest Flash ECC error
0x400008B4	FLASH0_ECC_UNCOR_ERROR_CNT	–	(7:0) ECC_UNCOR_ERROR_CNT	0x0	Flash ECC uncorrected error counter
0x400008B8	FLASH0_ECC_COR_ERROR_CNT	–	(7:0) ECC_COR_ERROR_CNT	0x0	Flash ECC corrected error counter
0x400008BC	FLASH0_NVM_STATUS	(16) CLEAR_NVM_STATUS	–	N/A	Clear all the NVM status bits
		–	(8) NVM_BIT_FAILURE	0x0	Indicates if a bit has failed in an address from the CC-NVM layout used by the CryptoCell
		–	(5:0) FAILED_NVM_ADDRESS	0x0	Last failing word address in CC-NVM layout (0x00 to 0x3F)

Address	Register Name	Register Write	Register Read	Default	Description
0x400008C0	FLASH0_MAIN_MASK	–	(11) DATA_A_35K_TO_40K_W_MASK	0x1	Authorize the access to the Flash MAIN Data array from 35K to 40K word block through the FLASH_IF registers.
		–	(10) DATA_A_30K_TO_35K_W_MASK	0x1	Authorize the access to the Flash MAIN Data array from 30K to 35K word block through the FLASH_IF registers.
		–	(9) DATA_A_25K_TO_30K_W_MASK	0x1	Authorize the access to the Flash MAIN Data array from 25K to 30K word block through the FLASH_IF registers.
		–	(8) DATA_A_20K_TO_25K_W_MASK	0x1	Authorize the access to the Flash MAIN Data array from 20K to 25K word block through the FLASH_IF registers.
		–	(7) DATA_A_15K_TO_20K_W_MASK	0x1	Authorize the access to the Flash MAIN Data array from 15K to 20K word block through the FLASH_IF registers.
		–	(6) DATA_A_10K_TO_15K_W_MASK	0x1	Authorize the access to the Flash MAIN Data array from 10K to 15K word block through the FLASH_IF registers.
		–	(5) DATA_A_5K_TO_10K_W_MASK	0x1	Authorize the access to the Flash MAIN Data array from 5K to 10K word block through the FLASH_IF registers.
		–	(4) DATA_A_0K_TO_5K_W_MASK	0x1	Authorize the access to the Flash MAIN Data array from 0 to 5K word block through the FLASH_IF registers.
		–	(3) CODE_A_66K_TO_88K_W_MASK	0x1	Authorize the access to the Flash MAIN Code array from 66K to 88K word block through the FLASH_IF registers.
		–	(2) CODE_A_44K_TO_66K_W_MASK	0x1	Authorize the access to the Flash MAIN Code array from 44K to 66K

Address	Register Name	Register Write	Register Read	Default	Description
					word block through the FLASH_IF registers.
		–	(1) CODE_A_22K_TO_44K_W_MASK	0x1	Authorize the access to the Flash MAIN Code array from 22K to 44K word block through the FLASH_IF registers.
		–	(0) CODE_A_0K_TO_22K_W_MASK	0x1	Authorize the access to the Flash MAIN Code array from 0 to 22K word block through the FLASH_IF registers.
0x400008FC	FLASH0_IF_ID_NUM	–	(20) FLASH_IF_NVR_FOR_CC312	0x0	NVR sectors 0 to 3 are used for the CryptoCell
		–	(19:16) FLASH_IF_NUMBER	0x0	FLASH_IF Instance number
		–	(15:8) FLASH_IF_MAJOR_REVISION	0x1	FLASH_IF Major Revision number
		–	(7:0) FLASH_IF_MINOR_REVISION	0x0	FLASH_IF Minor Revision number

A.7 GPIO Interface and Digital Pad control

Address	Register Name	Register Write	Register Read	Default	Description
0x40000900 - 0x4000093C	GPIO_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
		(7) NS_ACCESS_GPIO	(7) NS_ACCESS_GPIO	0x0	Non-Secure code can use GPIO (can only be written by a secure code)
		(6:0) IO_MODE	(6:0) IO_MODE	0x0	IO mode selection
0x40000940	GPIO_INPUT_DATA	-	(15:0) DATA	0x0	GPIO[15:0] read data
0x40000944	GPIO_OUTPUT_DATA	(15:0) DATA	(15:0) DATA	0x0	GPIO[15:0] output data
0x40000948	GPIO_OUTPUT_DATA_SET	(15:0) GPIO	-	N/A	GPIO[15:0] output data set
0x4000094C	GPIO_OUTPUT_DATA_CLR	(15:0) GPIO	-	N/A	GPIO[15:0] output data clear
0x40000950	GPIO_DIR	-	(15:0) GPIO	0x0	Get GPIO[15:0] direction
		(15:0) GPIO	-	N/A	Set GPIO[15:0] GPIO direction (only in GPIO_MODE_GPIO_x)
0x40000954	GPIO_MODE	-	(15:0) GPIO	0x0	GPIO[15:0] mode
0x40000958 - 0x40000964	GPIO_INT_CFG_*	(12) NS_ACCESS	(12) NS_ACCESS	0x0	Non-Secure code can access this GPIO_INT (can only be written by a secure code)
		(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
0x40000968	GPIO_INT_STATUS_S	-	(7) GPIO_INT3_STATUS	0x0	GPIO interrupt 3 status

Address	Register Name	Register Write	Register Read	Default	Description
		–	(6) GPIO_INT2_STATUS	0x0	GPIO interrupt 2 status
		–	(5) GPIO_INT1_STATUS	0x0	GPIO interrupt 1 status
		–	(4) GPIO_INT0_STATUS	0x0	GPIO interrupt 0 status
		(3) GPIO_INT3_CLEAR	–	N/A	GPIO interrupt 3 clear
		(2) GPIO_INT2_CLEAR	–	N/A	GPIO interrupt 2 clear
		(1) GPIO_INT1_CLEAR	–	N/A	GPIO interrupt 1 clear
		(0) GPIO_INT0_CLEAR	–	N/A	GPIO interrupt 0 clear
0x4000096C – 0x40000978	GPIO_INT_STATUS_*	–	(4) GPIO_INT_STATUS	0x0	GPIO interrupt status
		(0) GPIO_INT_CLEAR	–	N/A	GPIO interrupt clear
0x4000097C	GPIO_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
0x40000980	GPIO_JTAG_SW_PAD_CFG	(9) SWCLK_LPF	(9) SWCLK_LPF	0x0	SWCLK Low-Pass-Filter enable / disable
		(8) SWDIO_LPF	(8) SWDIO_LPF	0x0	SWDIO Low-Pass-Filter enable / disable
		(7) CM33_JTAG_DATA_EN	(7) CM33_JTAG_DATA_EN	0x1	CM33 JTAG data (TDI and TDO) on GPIO[3:2]
		(6) CM33_JTAG_TRST_EN	(6) CM33_JTAG_TRST_EN	0x1	CM33 JTAG TRST on GPIO4
		(5:4) SWCLK_PULL	(5:4) SWCLK_PULL	0x1	SWCLK pull-up enable / disable
		(3:2) SWDIO_DRIVE	(3:2) SWDIO_DRIVE	0x3	SWDIO drive strength
		(1:0) SWDIO_PULL	(1:0) SWDIO_PULL	0x1	SWDIO pull-up enable / disable
0x40000A00 –	GPIO_SRC_SPI_*	(12:8) CS	(12:8) CS	0x11	SPI_CS input selection

Address	Register Name	Register Write	Register Read	Default	Description
0x40000A04					
		(4:0) CLK	(4:0) CLK	0x11	SPI_CLK input selection
0x40000A08 - 0x40000A0C	GPIO_SRC_SPI_IO_*	(28:24) IO3	(28:24) IO3	0x11	SPI_IO3 input selection
		(20:16) IO2	(20:16) IO2	0x11	SPI_IO2 input selection
		(12:8) IO1	(12:8) IO1	0x11	SPI_IO1 input selection (master SERI)
		(4:0) IO0	(4:0) IO0	0x11	SPI_IO0 input selection (slave SERI)
0x40000A10	GPIO_SRC_UART	(4:0) RX	(4:0) RX	0x11	UART_RX input selection
0x40000A14 - 0x40000A18	GPIO_SRC_I2C_*	(12:8) SDA	(12:8) SDA	0x11	SDA input selection
		(4:0) SCL	(4:0) SCL	0x11	SCL input selection
0x40000A1C	GPIO_SRC_PCM	(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
0x40000A20	GPIO_SRC_LIN	(5) LIN_POLARITY	(5) LIN_POLARITY	0x1	LIN polarity
		(4:0) LIN	(4:0) LIN	0x10	LIN input selection
0x40000A24	GPIO_SRC_NMI	(5) NMI_POLARITY	(5) NMI_POLARITY	0x1	NMI polarity
		(4:0) NMI	(4:0) NMI	0x10	NMI input selection
0x40000A28	GPIO_SRC_BB_RX	(20:16) SYNC_P	(20:16) SYNC_P	0x12	Baseband controller interface 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
0x40000A2C	GPIO_SRC_BB_SPI	(4:0) MISO	(4:0) MISO	0x12	Baseband controller SPI_MISO input

Address	Register Name	Register Write	Register Read	Default	Description
					selection
0x40000A30	GPIO_SRC_BB_COEX	(12:8) WLAN_RX	(12:8) WLAN_RX	0x10	Baseband controller WLAN_RX input selection
		(4:0) WLAN_TX	(4:0) WLAN_TX	0x10	Baseband controller WLAN_TX input selection
0x40000A34	GPIO_SRC_BB_IQ_DATA	(28:24) IQ_DATA_3	(28:24) IQ_DATA_3	0x12	Baseband controller IQ_DATA_3 input selection
		(20:16) IQ_DATA_2	(20:16) IQ_DATA_2	0x12	Baseband controller IQ_DATA_2 input selection
		(12:8) IQ_DATA_1	(12:8) IQ_DATA_1	0x12	Baseband controller IQ_DATA_1 input selection
		(4:0) IQ_DATA_0	(4:0) IQ_DATA_0	0x12	Baseband controller IQ_DATA_0 input selection
0x40000A38	GPIO_SRC_BB_IQ_DATA_P	(4:0) IQ_DATA_P	(4:0) IQ_DATA_P	0x12	Baseband controller IQ_DATA_P input selection
0x40000A3C	GPIO_SRC_RF_SPI	(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
0x40000A40	GPIO_SRC_RF_GPIO03	(28:24) GPIO3	(28:24) GPIO3	0x12	RF front-end GPIO3 input selection
		(20:16) GPIO2	(20:16) GPIO2	0x10	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
0x40000A44	GPIO_SRC_RF_GPIO47	(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

Address	Register Name	Register Write	Register Read	Default	Description
		(4:0) GPIO4	(4:0) GPIO4	0x12	RE front-end GPIO4 input selection
0x40000A48	GPIO_SRC_RF_GPIO89	(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
0x40000A4C	GPIO_SRC_RF_CTE	(12:8) CTE_MODE	(12:8) CTE_MODE	0x12	RF front-end CTE_MODE input selection
		(4:0) CTE_SAMPLE_P	(4:0) CTE_SAMPLE_P	0x12	RF front-end CTE_SAMPLE_P input selection
0x40000A50	GPIO_SRC_ASCC	(12:8) ASYNC_CLOCK	(12:8) ASYNC_CLOCK	0x10	ASCC asynchronous clock source input selection
		(4:0) SYNC_PULSE	(4:0) SYNC_PULSE	0x10	ASCC synchronization pulse input selection
0x40000A54	GPIO_SRC_EXTCLK	(4:0) EXTCLK	(4:0) EXTCLK	0x10	EXTCLK input selection

A.8 DMA CONTROLLER

Address	Register Name	Register Write	Register Read	Default	Description
0x40001200	DMA0_CFG0	(31) COMPLETE_INT_ENABLE	(31) COMPLETE_INT_ENABLE	0x0	Raise an interrupt when the DMA transfer completes
		(30) CNT_INT_ENABLE	(30) CNT_INT_ENABLE	0x0	Raise an interrupt when the DMA transfer reaches the counter value
		(29) DEST_ADDR_LSB_TOGGLE	(29) DEST_ADDR_LSB_TOGGLE	0x0	Enable an address LSB toggling for the destination
		(28) SRC_ADDR_LSB_TOGGLE	(28) SRC_ADDR_LSB_TOGGLE	0x0	Enable an address LSB toggling for the source
		(27:24) DEST_ADDR_STEP	(27:24) DEST_ADDR_STEP	0x0	Configure whether the destination address increments/decrements in terms of destination word size
		(23:20) SRC_ADDR_STEP	(23:20) SRC_ADDR_STEP	0x0	Configure whether the source address increments/decrements in terms of source word size
		(19:14) SRC_DEST_WORD_SIZE	(19:14) SRC_DEST_WORD_SIZE	0x0	Select the source and destination word sizes for the transfer
		(13:10) DEST_SELECT	(13:10) DEST_SELECT	0x0	Select the request line for the destination
		(8:5) SRC_SELECT	(8:5) SRC_SELECT	0x0	Select the request line for the source
		(3:2) CHANNEL_PRIORITY	(3:2) CHANNEL_PRIORITY	0x0	Select the priority level for this channel
		(1) SRC_DEST_TRANS_LENGTH_SEL	(1) SRC_DEST_TRANS_LENGTH_SEL	0x0	Selects whether the transfer length counter depends on either the source word counts or the destination word count
		(0) BYTE_ORDER	(0) BYTE_ORDER	0x0	Select the byte ordering of the DMA channel

Address	Register Name	Register Write	Register Read	Default	Description
0x40001204	DMA0_CFG1	(31:16) INT_TRANSFER_LENGTH	(31:16) INT_TRANSFER_LENGTH	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
0x40001208	DMA0_CTRL	(13) INT_CNT_TRIGGER_ENABLE	(13) INT_CNT_TRIGGER_ENABLE	0x0	Enable waiting on a trigger when an interrupt counter event occurs
		(12:8) TRIGGER_SOURCE	(12:8) TRIGGER_SOURCE	0x0	Selects which event triggers this DMA channel when triggering is enabled
		(6) INT_CNT_ADDR_WRAP_ENABLE	(6) INT_CNT_ADDR_WRAP_ENABLE	0x0	Enable source or destination (depending on SRC_DEST_TRANS_LENGTH_SEL) address wrapping when an interrupt counter event occurs
		(5) CLEAR_BUFFER_WHEN_WRAPPING	(5) CLEAR_BUFFER_WHEN_WRAPPING	0x0	Clear buffer during address wrapping
		(4) BUFFER_CLEAR	–	N/A	Clear source buffer
		(3) CNTS_CLEAR	–	N/A	Clear counters
		(2:0) MODE_ENABLE	(2:0) MODE_ENABLE	0x0	Enable mode of operation of the DMA Channel
0x4000120C	DMA0_STATUS	–	(10) ACTIVE	0x0	Active status of the channel
		–	(9) CNT_INT	0x0	Indicate if a counter interrupt has occurred on DMA channel
		–	(8) COMPLETE_INT	0x0	Indicate if a complete interrupt has occurred on DMA channel
		(6) CNT_INT_CLEAR	–	N/A	Clear the counter interrupt flag
		(5) COMPLETE_INT_CLEAR	–	N/A	Clear the complete interrupt flag
		(4) SRC_BUFFER_FILL_	–	N/A	Enable writing of SRC_BUFFER_

Address	Register Name	Register Write	Register Read	Default	Description
		LVL_WR			FILL_LVL
		(3:0) SRC_BUFFER_ FILL_LVL	(3:0) SRC_BUFFER_ FILL_LVL	0x0	The number of nibbles currently in the source buffer (0: buffer is empty, 8: the 32-bit buffer is completely full)
0x40001210	DMA0_SRC_ADDR	(31:0) SRC_ADDR	(31:0) SRC_ADDR	0x0	Address for the source of data transferred using DMA channel
0x40001214	DMA0_DEST_ADDR	(31:0) DEST_ADDR	(31:0) DEST_ADDR	0x0	Address for the destination of data transferred using DMA channel
0x40001218	DMA0_CNTS	(31:16) INT_TRANSFER_ WORD_CNT	(31:16) INT_ TRANSFER_WORD_CNT	0x0	Interrupt word counter (automatically reset after each counter interrupt)
		(15:0) TRANSFER_WORD_ CNT	(15:0) TRANSFER_ WORD_CNT	0x0	The number of words that have been transferred using DMA channel during the current transfer
0x4000121C	DMA0_SRC_BUFFER	(31:0) SRC_BUFFER	(31:0) SRC_BUFFER	0x0	Packed word which has been read from the source addresses.
0x400012FC	DMA0_ID_NUM	–	(20) DMA_24BIT_WORD	0x0	24-bit word size supported
		–	(19:16) DMA_NUMBER	0x0	DMA channel number
		–	(15:8) DMA_MAJOR_ REVISION	0x1	DMA Major Revision number
		–	(7:0) DMA_MINOR_ REVISION	0x0	DMA Minor Revision number
0x40001300	DMA1_CFG0	(31) COMPLETE_INT_ ENABLE	(31) COMPLETE_INT_ ENABLE	0x0	Raise an interrupt when the DMA transfer completes
		(30) CNT_INT_ENABLE	(30) CNT_INT_ENABLE	0x0	Raise an interrupt when the DMA transfer reaches the counter value
		(29) DEST_ADDR_LSB_ TOGGLE	(29) DEST_ADDR_LSB_ TOGGLE	0x0	Enable an address LSB toggling for the destination
		(28) SRC_ADDR_LSB_	(28) SRC_ADDR_LSB_	0x0	Enable an address LSB toggling for the

Address	Register Name	Register Write	Register Read	Default	Description
		TOGGLE	TOGGLE		source
		(27:24) DEST_ADDR_STEP	(27:24) DEST_ADDR_STEP	0x0	Configure whether the destination address increments/decrements in terms of destination word size
		(23:20) SRC_ADDR_STEP	(23:20) SRC_ADDR_STEP	0x0	Configure whether the source address increments/decrements in terms of source word size
		(19:14) SRC_DEST_WORD_SIZE	(19:14) SRC_DEST_WORD_SIZE	0x0	Select the source and destination word sizes for the transfer
		(13:10) DEST_SELECT	(13:10) DEST_SELECT	0x0	Select the request line for the destination
		(8:5) SRC_SELECT	(8:5) SRC_SELECT	0x0	Select the request line for the source
		(3:2) CHANNEL_PRIORITY	(3:2) CHANNEL_PRIORITY	0x0	Select the priority level for this channel
		(1) SRC_DEST_TRANS_LENGTH_SEL	(1) SRC_DEST_TRANS_LENGTH_SEL	0x0	Selects whether the transfer length counter depends on either the source word counts or the destination word count
		(0) BYTE_ORDER	(0) BYTE_ORDER	0x0	Select the byte ordering of the DMA channel
0x40001304	DMA1_CFG1	(31:16) INT_TRANSFER_LENGTH	(31:16) INT_TRANSFER_LENGTH	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
0x40001308	DMA1_CTRL	(13) INT_CNT_TRIGGER_ENABLE	(13) INT_CNT_TRIGGER_ENABLE	0x0	Enable waiting on a trigger when an interrupt counter event occurs
		(12:8) TRIGGER_SOURCE	(12:8) TRIGGER_SOURCE	0x0	Selects which event triggers this DMA channel when triggering is enabled

Address	Register Name	Register Write	Register Read	Default	Description
		(6) INT_CNT_ADDR_ WRAP_ENABLE	(6) INT_CNT_ADDR_ WRAP_ENABLE	0x0	Enable source or destination (depending on SRC_DEST_TRANS_ LENGTH_SEL) address wrapping when an interrupt counter event occurs
		(5) CLEAR_BUFFER_ WHEN_WRAPPING	(5) CLEAR_BUFFER_ WHEN_WRAPPING	0x0	Clear buffer during address wrapping
		(4) BUFFER_CLEAR	-	N/A	Clear source buffer
		(3) CNTS_CLEAR	-	N/A	Clear counters
		(2:0) MODE_ENABLE	(2:0) MODE_ENABLE	0x0	Enable mode of operation of the DMA Channel
0x4000130C	DMA1_STATUS	-	(10) ACTIVE	0x0	Active status of the channel
		-	(9) CNT_INT	0x0	Indicate if a counter interrupt has occurred on DMA channel
		-	(8) COMPLETE_INT	0x0	Indicate if a complete interrupt has occurred on DMA channel
		(6) CNT_INT_CLEAR	-	N/A	Clear the counter interrupt flag
		(5) COMPLETE_INT_ CLEAR	-	N/A	Clear the complete interrupt flag
		(4) SRC_BUFFER_FILL_ LVL_WR	-	N/A	Enable writing of SRC_BUFFER_ FILL_LVL
		(3:0) SRC_BUFFER_ FILL_LVL	(3:0) SRC_BUFFER_ FILL_LVL	0x0	The number of nibbles currently in the source buffer (0: buffer is empty, 8: the 32-bit buffer is completely full)
0x40001310	DMA1_SRC_ADDR	(31:0) SRC_ADDR	(31:0) SRC_ADDR	0x0	Address for the source of data transferred using DMA channel
0x40001314	DMA1_DEST_ADDR	(31:0) DEST_ADDR	(31:0) DEST_ADDR	0x0	Address for the destination of data transferred using DMA channel
0x40001318	DMA1_CNTS	(31:16) INT_TRANSFER_	(31:16) INT_	0x0	Interrupt word counter (automatically

Address	Register Name	Register Write	Register Read	Default	Description
		WORD_CNT	TRANSFER_WORD_CNT		reset after each counter interrupt)
		(15:0) TRANSFER_WORD_CNT	(15:0) TRANSFER_WORD_CNT	0x0	The number of words that have been transferred using DMA channel during the current transfer
0x4000131C	DMA1_SRC_BUFFER	(31:0) SRC_BUFFER	(31:0) SRC_BUFFER	0x0	Packed word which has been read from the source addresses.
0x400013FC	DMA1_ID_NUM	–	(20) DMA_24BIT_WORD	0x0	24-bit word size supported
		–	(19:16) DMA_NUMBER	0x0	DMA channel number
		–	(15:8) DMA_MAJOR_REVISION	0x1	DMA Major Revision number
		–	(7:0) DMA_MINOR_REVISION	0x0	DMA Minor Revision number
0x40001400	DMA2_CFG0	(31) COMPLETE_INT_ENABLE	(31) COMPLETE_INT_ENABLE	0x0	Raise an interrupt when the DMA transfer completes
		(30) CNT_INT_ENABLE	(30) CNT_INT_ENABLE	0x0	Raise an interrupt when the DMA transfer reaches the counter value
		(29) DEST_ADDR_LSB_TOGGLE	(29) DEST_ADDR_LSB_TOGGLE	0x0	Enable an address LSB toggling for the destination
		(28) SRC_ADDR_LSB_TOGGLE	(28) SRC_ADDR_LSB_TOGGLE	0x0	Enable an address LSB toggling for the source
		(27:24) DEST_ADDR_STEP	(27:24) DEST_ADDR_STEP	0x0	Configure whether the destination address increments/decrements in terms of destination word size
		(23:20) SRC_ADDR_STEP	(23:20) SRC_ADDR_STEP	0x0	Configure whether the source address increments/decrements in terms of source word size
		(19:14) SRC_DEST_WORD_SIZE	(19:14) SRC_DEST_WORD_SIZE	0x0	Select the source and destination word sizes for the transfer

Address	Register Name	Register Write	Register Read	Default	Description
		(13:10) DEST_SELECT	(13:10) DEST_SELECT	0x0	Select the request line for the destination
		(8:5) SRC_SELECT	(8:5) SRC_SELECT	0x0	Select the request line for the source
		(3:2) CHANNEL_ PRIORITY	(3:2) CHANNEL_ PRIORITY	0x0	Select the priority level for this channel
		(1) SRC_DEST_TRANS_ LENGTH_SEL	(1) SRC_DEST_TRANS_ LENGTH_SEL	0x0	Selects whether the transfer length counter depends on either the source word counts or the destination word count
		(0) BYTE_ORDER	(0) BYTE_ORDER	0x0	Select the byte ordering of the DMA channel
0x40001404	DMA2_CFG1	(31:16) INT_TRANSFER_ LENGTH	(31:16) INT_ TRANSFER_LENGTH	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
0x40001408	DMA2_CTRL	(13) INT_CNT_TRIGGER_ ENABLE	(13) INT_CNT_ TRIGGER_ENABLE	0x0	Enable waiting on a trigger when an interrupt counter event occurs
		(12:8) TRIGGER_SOURCE	(12:8) TRIGGER_ SOURCE	0x0	Selects which event triggers this DMA channel when triggering is enabled
		(6) INT_CNT_ADDR_ WRAP_ENABLE	(6) INT_CNT_ADDR_ WRAP_ENABLE	0x0	Enable source or destination (depending on SRC_DEST_TRANS_ LENGTH_SEL) address wrapping when an interrupt counter event occurs
		(5) CLEAR_BUFFER_ WHEN_WRAPPING	(5) CLEAR_BUFFER_ WHEN_WRAPPING	0x0	Clear buffer during address wrapping
		(4) BUFFER_CLEAR	–	N/A	Clear source buffer
		(3) CNTS_CLEAR	–	N/A	Clear counters

Address	Register Name	Register Write	Register Read	Default	Description
		(2:0) MODE_ENABLE	(2:0) MODE_ENABLE	0x0	Enable mode of operation of the DMA Channel
0x4000140C	DMA2_STATUS	–	(10) ACTIVE	0x0	Active status of the channel
		–	(9) CNT_INT	0x0	Indicate if a counter interrupt has occurred on DMA channel
		–	(8) COMPLETE_INT	0x0	Indicate if a complete interrupt has occurred on DMA channel
		(6) CNT_INT_CLEAR	–	N/A	Clear the counter interrupt flag
		(5) COMPLETE_INT_ CLEAR	–	N/A	Clear the complete interrupt flag
		(4) SRC_BUFFER_FILL_ LVL_WR	–	N/A	Enable writing of SRC_BUFFER_ FILL_LVL
		(3:0) SRC_BUFFER_ FILL_LVL	(3:0) SRC_BUFFER_ FILL_LVL	0x0	The number of nibbles currently in the source buffer (0: buffer is empty, 8: the 32-bit buffer is completely full)
0x40001410	DMA2_SRC_ADDR	(31:0) SRC_ADDR	(31:0) SRC_ADDR	0x0	Address for the source of data transferred using DMA channel
0x40001414	DMA2_DEST_ADDR	(31:0) DEST_ADDR	(31:0) DEST_ADDR	0x0	Address for the destination of data transferred using DMA channel
0x40001418	DMA2_CNTS	(31:16) INT_TRANSFER_ WORD_CNT	(31:16) INT_ TRANSFER_WORD_CNT	0x0	Interrupt word counter (automatically reset after each counter interrupt)
		(15:0) TRANSFER_WORD_ CNT	(15:0) TRANSFER_ WORD_CNT	0x0	The number of words that have been transferred using DMA channel during the current transfer
0x4000141C	DMA2_SRC_BUFFER	(31:0) SRC_BUFFER	(31:0) SRC_BUFFER	0x0	Packed word which has been read from the source addresses.
0x400014FC	DMA2_ID_NUM	–	(20) DMA_24BIT_WORD	0x0	24-bit word size supported

Address	Register Name	Register Write	Register Read	Default	Description
		-	(19:16) DMA_NUMBER	0x0	DMA channel number
		-	(15:8) DMA_MAJOR_REVISION	0x1	DMA Major Revision number
		-	(7:0) DMA_MINOR_REVISION	0x0	DMA Minor Revision number
0x40001500	DMA3_CFG0	(31) COMPLETE_INT_ENABLE	(31) COMPLETE_INT_ENABLE	0x0	Raise an interrupt when the DMA transfer completes
		(30) CNT_INT_ENABLE	(30) CNT_INT_ENABLE	0x0	Raise an interrupt when the DMA transfer reaches the counter value
		(29) DEST_ADDR_LSB_TOGGLE	(29) DEST_ADDR_LSB_TOGGLE	0x0	Enable an address LSB toggling for the destination
		(28) SRC_ADDR_LSB_TOGGLE	(28) SRC_ADDR_LSB_TOGGLE	0x0	Enable an address LSB toggling for the source
		(27:24) DEST_ADDR_STEP	(27:24) DEST_ADDR_STEP	0x0	Configure whether the destination address increments/decrements in terms of destination word size
		(23:20) SRC_ADDR_STEP	(23:20) SRC_ADDR_STEP	0x0	Configure whether the source address increments/decrements in terms of source word size
		(19:14) SRC_DEST_WORD_SIZE	(19:14) SRC_DEST_WORD_SIZE	0x0	Select the source and destination word sizes for the transfer
		(13:10) DEST_SELECT	(13:10) DEST_SELECT	0x0	Select the request line for the destination
		(8:5) SRC_SELECT	(8:5) SRC_SELECT	0x0	Select the request line for the source
		(3:2) CHANNEL_PRIORITY	(3:2) CHANNEL_PRIORITY	0x0	Select the priority level for this channel
		(1) SRC_DEST_TRANS_LENGTH_SEL	(1) SRC_DEST_TRANS_LENGTH_SEL	0x0	Selects whether the transfer length counter depends on either the source

Address	Register Name	Register Write	Register Read	Default	Description
					word counts or the destination word count
		(0) BYTE_ORDER	(0) BYTE_ORDER	0x0	Select the byte ordering of the DMA channel
0x40001504	DMA3_CFG1	(31:16) INT_TRANSFER_LENGTH	(31:16) INT_TRANSFER_LENGTH	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
0x40001508	DMA3_CTRL	(13) INT_CNT_TRIGGER_ENABLE	(13) INT_CNT_TRIGGER_ENABLE	0x0	Enable waiting on a trigger when an interrupt counter event occurs
		(12:8) TRIGGER_SOURCE	(12:8) TRIGGER_SOURCE	0x0	Selects which event triggers this DMA channel when triggering is enabled
		(6) INT_CNT_ADDR_WRAP_ENABLE	(6) INT_CNT_ADDR_WRAP_ENABLE	0x0	Enable source or destination (depending on SRC_DEST_TRANS_LENGTH_SEL) address wrapping when an interrupt counter event occurs
		(5) CLEAR_BUFFER_WHEN_WRAPPING	(5) CLEAR_BUFFER_WHEN_WRAPPING	0x0	Clear buffer during address wrapping
		(4) BUFFER_CLEAR	–	N/A	Clear source buffer
		(3) CNTS_CLEAR	–	N/A	Clear counters
		(2:0) MODE_ENABLE	(2:0) MODE_ENABLE	0x0	Enable mode of operation of the DMA Channel
0x4000150C	DMA3_STATUS	–	(10) ACTIVE	0x0	Active status of the channel
		–	(9) CNT_INT	0x0	Indicate if a counter interrupt has occurred on DMA channel
		–	(8) COMPLETE_INT	0x0	Indicate if a complete interrupt has occurred on DMA channel

Address	Register Name	Register Write	Register Read	Default	Description
		(6) CNT_INT_CLEAR	-	N/A	Clear the counter interrupt flag
		(5) COMPLETE_INT_CLEAR	-	N/A	Clear the complete interrupt flag
		(4) SRC_BUFFER_FILL_LVL_WR	-	N/A	Enable writing of SRC_BUFFER_FILL_LVL
		(3:0) SRC_BUFFER_FILL_LVL	(3:0) SRC_BUFFER_FILL_LVL	0x0	The number of nibbles currently in the source buffer (0: buffer is empty, 8: the 32-bit buffer is completely full)
0x40001510	DMA3_SRC_ADDR	(31:0) SRC_ADDR	(31:0) SRC_ADDR	0x0	Address for the source of data transferred using DMA channel
0x40001514	DMA3_DEST_ADDR	(31:0) DEST_ADDR	(31:0) DEST_ADDR	0x0	Address for the destination of data transferred using DMA channel
0x40001518	DMA3_CNTS	(31:16) INT_TRANSFER_WORD_CNT	(31:16) INT_TRANSFER_WORD_CNT	0x0	Interrupt word counter (automatically reset after each counter interrupt)
		(15:0) TRANSFER_WORD_CNT	(15:0) TRANSFER_WORD_CNT	0x0	The number of words that have been transferred using DMA channel during the current transfer
0x4000151C	DMA3_SRC_BUFFER	(31:0) SRC_BUFFER	(31:0) SRC_BUFFER	0x0	Packed word which has been read from the source addresses.
0x400015FC	DMA3_ID_NUM	-	(20) DMA_24BIT_WORD	0x0	24-bit word size supported
		-	(19:16) DMA_NUMBER	0x0	DMA channel number
		-	(15:8) DMA_MAJOR_REVISION	0x1	DMA Major Revision number
		-	(7:0) DMA_MINOR_REVISION	0x0	DMA Minor Revision number

A.9 LIN INTERFACE

Address	Register Name	Register Write	Register Read	Default	Description
0x40002000	LIN0_CFG	(8) CHECKSUM_MODE	(8) CHECKSUM_MODE	0x0	Select checksum mode
		(4) INIT	(4) INIT	0x0	Start C617 initialization
		–	(0) STANDBY	0x0	C617 state
0x40002004	LIN0_CTRL	(8) RESET	–	N/A	Reset the LIN module to default configuration
		–	(7) RHC	0x0	This bit is set, when frame header is successfully received (synchronization passed and frame identifier received). Bit is cleared during transition to RX_ID state. Setting of bit to high by controller has higher priority than bit clear by CM33 (frame identifier received). Bit is cleared on MCU read. Bit is cleared during transition to RX_ID state. Setting of bit to high by controller has higher priority than bit clear by CM33
		–	(6) RXF	0x0	This bit is set when LIN controller successfully receives all data bytes and checksum is correct. Bit is cleared during transition to RX_ID state. Setting of bit to high by controller has higher priority than bit clear by CM33.
		–	(5) TXF	0x0	This bit is set when controller starts to transmit. Bit is cleared when all bytes are transmitted or bit error occurs or break/sync is detected or

Address	Register Name	Register Write	Register Read	Default	Description
					during transition to RX_ID state.
		–	(4) ENABLE_STATUS	0x0	LIN enable status
		(1) DISABLE	–	N/A	Disable the LIN
		(0) ENABLE	–	N/A	Enable the LIN
0x40002008	LIN0_ERROR	(11) CLR_CE	–	N/A	Clear Checksum error flag
		(10) CLR_PE	–	N/A	Clear Parity error flag
		(9) CLR_BE	–	N/A	Clear Bit error flag
		(8) CLR_FE	–	N/A	Clear Frame error flag
		–	(7) CE	0x0	Checksum error: this bit is set to high when all data bytes are received but value of calculated checksum does not equal the value of 255.
		–	(6) PE	0x0	Parity error: this bit is set when parity of received identifier is different from calculated parity.
		–	(5) BE	0x0	Bit error: this bit is set when transmitted data is different from read back data. This bit is cleared on MCU read.
		–	(4) FE	0x0	Framing error: this bit is set when received stop bit is 0 in sync field, identifier or data byte.
		–	(3) CE_WRC	0x0	Checksum error: this bit is set to high when all data bytes are received but value of calculated checksum does not equal the value of 255. This bit is cleared on MCU read.

Address	Register Name	Register Write	Register Read	Default	Description
		–	(2) PE_WRC	0x0	Parity error: this bit is set when parity of received identifier is different from calculated parity. This bit is cleared on MCU read.
		–	(1) BE_WRC	0x0	Bit error: this bit is set when transmitted data is different from read back data. This bit is cleared on MCU read.
		–	(0) FE_WRC	0x0	Framing error: this bit is set when received stop bit is 0 in sync field, identifier or data byte. This bit is cleared on MCU read.
0x4000200C	LIN0_PID	–	(7:0) PID	0x0	LIN Protected Identifier register
0x40002010	LIN0_DLB	(4) DELAY	(4) DELAY	0x0	Complete the PID stop bit before sending the first byte
		(2:0) DLBT	(2:0) DLBT	0x0	Number of data bytes to transmit.
0x40002014	LIN0_DLBR	(2:0) DLBR	(2:0) DLBR	0x0	Number of data bytes to receive.
0x40002018 – 0x40002034	LIN0_DATA_*	(7:0) DATA	(7:0) DATA	0x0	LIN Data byte received or to transmit
0x40002038	LIN0_DATA_WORD0	(31:24) DATA3	(31:24) DATA3	0x0	LIN Data[3] byte received or to transmit
		(23:16) DATA2	(23:16) DATA2	0x0	LIN Data[2] byte received or to transmit
		(15:8) DATA1	(15:8) DATA1	0x0	LIN Data[1] byte received or to transmit
		(7:0) DATA0	(7:0) DATA0	0x0	LIN Data[0] byte received or to transmit
0x4000203C	LIN0_DATA_WORD1	(31:24) DATA7	(31:24) DATA7	0x0	LIN Data[7] byte received or to transmit

Address	Register Name	Register Write	Register Read	Default	Description
		(23:16) DATA6	(23:16) DATA6	0x0	LIN Data[6] byte received or to transmit
		(15:8) DATA5	(15:8) DATA5	0x0	LIN Data[5] byte received or to transmit
		(7:0) DATA4	(7:0) DATA4	0x0	LIN Data[4] byte received or to transmit
0x40002040	LIN0_CHECKSUM	–	(7:0) CHECKSUM	0x0	Checksum
0x400020F4	LIN0_SYNCH	–	(13:0) TSYNC	0x0	Duration of TSYNC
0x400020FC	LIN0_ID_NUM	–	(15:8) LIN_MAJOR_ REVISION	0x1	LIN Major Revision number
		–	(7:0) LIN_MINOR_ REVISION	0x0	LIN Minor Revision number

A.10 LSAD

Address	Register Name	Register Write	Register Read	Default	Description
0x40001C00 - 0x40001C1C	LSAD_DATA_TRIM_CH_*	–	(13:0) DATA	0x0	14-bit LSAD conversion result
0x40001C20 - 0x40001C3C	LSAD_DATA_AUDIO_CH_*	–	(31:0) DATA	0x0	14-bit LSAD conversion result (sign extended to 32 bits)
0x40001C40 - 0x40001C5C	LSAD_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
0x40001C60	LSAD_CFG	(4) CONTINUOUS_MODE	(4) CONTINUOUS_MODE	0x0	LSAD continuously sampling the channel selected as interrupt source (LSAD_INT_CH_NUM)
		(3:0) FREQ	(3:0) FREQ	0x0	Defines the sampling frequency of the LSAD channels
0x40001C64	LSAD_OFFSET	(14:0) DATA	(14:0) DATA	0x0	15-bit LSAD signed offset
0x40001C68	LSAD_INT_ENABLE	(3:1) LSAD_INT_CH_NUM	(3:1) LSAD_INT_CH_ NUM	0x0	Channel number triggering the LSAD interrupt
		(0) LSAD_INT_ENABLE	(0) LSAD_INT_ENABLE	0x0	The LSAD new sample ready interrupt mask
0x40001C6C	LSAD_MONITOR_CFG	(23:16) ALARM_COUNT_ VALUE	(23:16) ALARM_ COUNT_VALUE	0x0	An Alarm Status bit is set and an interrupt generated when SUPPLY_ COUNT_VALUE = ALARM_COUNT_ VALUE
		(15:8) MONITOR_ THRESHOLD	(15:8) MONITOR_ THRESHOLD	0xB3	Low voltage detection threshold (7.8 mV steps)
		(2:0) MONITOR_SRC	(2:0) MONITOR_SRC	0x7	Selects the source channel to be monitored
0x40001C70	LSAD_MONITOR_COUNT_VAL	–	(7:0) MONITOR_	0x0	Number of times the voltage has fallen

Address	Register Name	Register Write	Register Read	Default	Description
			COUNT_VALUE		below the monitor voltage threshold. The counter is reset when read.
0x40001C74	LSAD_MONITOR_STATUS	(12) MONITOR_ALARM_CLEAR	–	N/A	Monitoring alarm status bit
		(9) LSAD_OVERRUN_CLEAR	–	N/A	LSAD Overrun condition
		(8) LSAD_READY_CLEAR	–	N/A	LSAD new sample Ready status bit
		–	(4) MONITOR_ALARM_STAT	0x0	Monitoring alarm status bit
		–	(1) LSAD_OVERRUN_STAT	0x0	LSAD Overrun condition
		–	(0) LSAD_READY_STAT	0x0	LSAD new sample Ready status bit
0x40001C78	LSAD_PRE_SEL_INPUT	(14:12) LSAD_PRE_SEL_IN3	(14:12) LSAD_PRE_SEL_IN3	0x0	LSAD input pre-selection
		(10:8) LSAD_PRE_SEL_IN2	(10:8) LSAD_PRE_SEL_IN2	0x0	LSAD input pre-selection
		(6:4) LSAD_PRE_SEL_IN1	(6:4) LSAD_PRE_SEL_IN1	0x0	LSAD input pre-selection
		(2:0) LSAD_PRE_SEL_IN0	(2:0) LSAD_PRE_SEL_IN0	0x0	LSAD input pre-selection
0x40001C7C	LSAD_DUTY	(15:14) CH7_DUTY_CFG	(15:14) CH7_DUTY_CFG	0x0	Input to channel 7 duty config
		(13:12) CH6_DUTY_CFG	(13:12) CH6_DUTY_CFG	0x0	Input to channel 6 duty config
		(11:10) CH5_DUTY_CFG	(11:10) CH5_DUTY_CFG	0x0	Input to channel 5 duty config
		(9:8) CH4_DUTY_CFG	(9:8) CH4_DUTY_CFG	0x0	Input to channel 4 duty config

Address	Register Name	Register Write	Register Read	Default	Description
		(7:6) CH3_DUTY_CFG	(7:6) CH3_DUTY_CFG	0x0	Input to channel 3 duty config
		(5:4) CH2_DUTY_CFG	(5:4) CH2_DUTY_CFG	0x0	Input to channel 2 duty config
		(3:2) CH1_DUTY_CFG	(3:2) CH1_DUTY_CFG	0x0	Input to channel 1 duty config
		(1:0) CH0_DUTY_CFG	(1:0) CH0_DUTY_CFG	0x0	Input to channel 0 duty config
0x40001CFC	LSAD_ID_NUM	–	(15:8) MAJOR_ REVISION	0x1	LSAD Major Revision number
		–	(7:0) MINOR_ REVISION	0x0	LSAD Minor Revision number

A.11 SENSOR INTERFACE

Address	Register Name	Register Write	Register Read	Default	Description
0x40001D00	SENSOR_SAR_CFG	(22:20) SAR_PRE_SEL_ODD	(22:20) SAR_PRE_SEL_ODD	0x0	Pre-selection of odd GPIOs
		(18:16) SAR_PRE_SEL_EVEN	(18:16) SAR_PRE_SEL_EVEN	0x0	Pre-selection of even GPIOs
		(14:12) SAR_IN_P	(14:12) SAR_IN_P	0x4	Defines the positive input signal
		(10:8) SAR_IN_N	(10:8) SAR_IN_N	0x4	Defines the negative input signal
		(3) SAR_DATA_OUT_RX_DMA_EN	(3) SAR_DATA_OUT_RX_DMA_EN	0x0	Enable / disable the DMA trigger when SAR data available
		(2) SAR_DATA_OUT_UPDATE	(2) SAR_DATA_OUT_UPDATE	0x0	Enable / disable the SAR data update
		(1) SAR_SUPPLY_EN	(1) SAR_SUPPLY_EN	0x0	Enable / disable the SAR supply
		(0) SAR_SUPPLY_SELECT	(0) SAR_SUPPLY_SELECT	0x0	SAR supply selection
0x40001D04	SENSOR_SAR_CTRL	(14) START_SINGLE	–	N/A	Start single conversion command
		(13) START_CONTINUOUS	(13) START_CONTINUOUS	0x0	Start continuous conversions command
		–	(12) BUSY	0x0	Conversion in progess flag
		(9:8) MODE	(9:8) MODE	0x0	Conversion mode selection
		(6:4) NUM_SAMPLE	(6:4) NUM_SAMPLE	0x0	Defines the number of clk cycles for data sampling
		(3:0) OUT_SEL	(3:0) OUT_SEL	0x0	Output selection (undefined values return 0)
0x40001D08	SENSOR_SAR_DATA	–	(15:0) SAR_OUT	0x0	15 bit SAR conversion result (signed)
0x40001D0C	SENSOR_PC_CFG	(16) PC_COUNT_ON	(16) PC_COUNT_ON	0x0	Pulse counter count on rising edge or high state

Address	Register Name	Register Write	Register Read	Default	Description
		(14:12) PC_SRC_SEL	(14:12) PC_SRC_SEL	0x5	Pulse count source selection
		(9:0) COUNT_INT	(9:0) COUNT_INT	0x3	Duration of count state
0x40001D10	SENSOR_PC_COUNT	–	(9:0) VALUE	0x0	Sensor pulse counter current value
0x40001D14	SENSOR_TIMER_COUNT	–	(9:0) VALUE	0x0	Sensor timer counter current value
0x40001D18	SENSOR_CFG	(24) CLK_SEL	(24) CLK_SEL	0x1	Clock source selection
		(20) SRC_SEL	(20) SRC_SEL	0x0	Sample source selection
		(13) DLY_EN	(13) DLY_EN	0x1	Delay state enable
		(12) DLY_DIV_EN	(12) DLY_DIV_EN	0x0	Delay divider selection
		(9:0) DLY	(9:0) DLY	0x0	Absolute Value of main counter to trigger the change of delay state
0x40001D1C	SENSOR_CTRL	(31) RESET	–	N/A	Reset the Sensor interface (SAR and Pulse Counter)
		–	(12) STATE	0x0	Sensor state
		–	(8) STATUS	0x0	Sensor status
		(0) ENABLE	(0) ENABLE	0x0	Sensor
0x40001D20	SENSOR_FIFO_CFG	–	(12:8) FIFO_LEVEL	0x0	Number of samples stored in FIFO
		(6) FIFO_RX_INT_EN	(6) FIFO_RX_INT_EN	0x0	Enable / disable the interrupt when FIFO is full
		(5) FIFO_RX_DMA_EN	(5) FIFO_RX_DMA_EN	0x0	Enable / disable the DMA trigger when FIFIO is full
		(4) FIFO_STORE_EN	(4) FIFO_STORE_EN	0x1	Enable the storage of sample on the FIFO
		(3:0) FIFO_SIZE	(3:0) FIFO_SIZE	0x0	Number of samples stored before waking up the core
0x40001D24	SENSOR_PROCESSING	(24) SUM_EN	(24) SUM_EN	0x0	Summation enable
		(19:16) NBR_SAMPLES	(19:16) NBR_SAMPLES	0x0	Number of samples used for wakeup in

Address	Register Name	Register Write	Register Read	Default	Description
					sensor detect mode
0x40001D28	SENSOR_THRESHOLD_MIN	(23) THRESHOLD_MIN_EN	(23) THRESHOLD_MIN_EN	0x0	Threshold min comparator is enabled.
		(18:0) THRESHOLD_MIN	(18:0) THRESHOLD_MIN	0x0	Absolute sensor data threshold min for wake-up
0x40001D2C	SENSOR_THRESHOLD_MAX	(23) THRESHOLD_MAX_EN	(23) THRESHOLD_MAX_EN	0x0	Threshold max comparator is enabled
		(18:0) THRESHOLD_MAX	(18:0) THRESHOLD_MAX	0x0	Absolute sensor data threshold max for wake-up
0x40001D30 - 0x40001D6C	SENSOR_FIFO_DATA_*	–	(31:0) VALUE	0x0	Sensor FIFO output data (signed)

A.12 SPI INTERFACE

Address	Register Name	Register Write	Register Read	Default	Description
0x40000B00	SPI0_CFG	(23) TX_DMA_ENABLE	(23) TX_DMA_ENABLE	0x0	Enable/disable the TX DMA request
		(22) RX_DMA_ENABLE	(22) RX_DMA_ENABLE	0x0	Enable/disable the RX DMA request
		(21) TX_END_INT_ ENABLE	(21) TX_END_INT_ ENABLE	0x0	Enable/disable the TX interrupt
		(20) TX_START_INT_ ENABLE	(20) TX_START_INT_ ENABLE	0x0	Enable/disable the TX interrupt
		(19) RX_INT_ENABLE	(19) RX_INT_ENABLE	0x0	Enable/disable the RX interrupt
		(18) CS_RISE_INT_ ENABLE	(18) CS_RISE_INT_ ENABLE	0x0	Enable/disable the CS rise interrupt (slave mode only)
		(17) OVERRUN_INT_ ENABLE	(17) OVERRUN_INT_ ENABLE	0x0	Enable/disable the overrun interrupt
		(16) UNDERRUN_INT_ ENABLE	(16) UNDERRUN_INT_ ENABLE	0x0	Enable/disable the underrun interrupt
		(14:13) MODE	(14:13) MODE	0x0	Select the SPI master mode (ignored in slave mode)
		(12:8) WORD_SIZE	(12:8) WORD_SIZE	0x0	Select the SPI word size (word size = SPI_WORD_SIZE + 1)
		(7:4) PRESCALE	(7:4) PRESCALE	0x0	Prescale the SPI clock for master mode
		(2) CLK_PHASE	(2) CLK_PHASE	0x0	Select the SPI clock phase
		(1) CLK_POLARITY	(1) CLK_POLARITY	0x0	Select the SPI clock polarity
		(0) SLAVE	(0) SLAVE	0x0	Use the SPI interface as master or slave
0x40000B04	SPI0_CTRL	–	(19) CS_STATUS	0x1	SPI CS status
		–	(18:17) MODE_STATUS	0x0	SPI mode status

Address	Register Name	Register Write	Register Read	Default	Description
		–	(16) ENABLE_STATUS	0x0	SPI enable status
		(9) CS_0	–	N/A	Lower the SPI chip-select line (master mode)
		(8) CS_1	–	N/A	Raise the SPI chip-select line (master mode)
		(7) MODE_NOP	–	N/A	Set mode to no operation
		(6) MODE_WRITE	–	N/A	Set mode to read operation
		(5) MODE_READ	–	N/A	Set mode to write operation
		(4) MODE_READ_WRITE	–	N/A	Set mode to read and write operation
		(3) START	–	N/A	Start a data transfer in master read mode
		(2) RESET	–	N/A	Reset the SPI interface
		(1) DISABLE	–	N/A	Disable the SPI interface
		(0) ENABLE	–	N/A	Enable the SPI interface
0x40000B08	SPI0_STATUS	–	(13) BUSY	0x0	Indicate that the reception or transmission of the data is ongoing
		–	(12) TX_REQ	0x1	Indicate that TX data can be written
		–	(11) RX_REQ	0x0	Indicate that RX data can be read
		–	(10) CS_RISE	0x0	Indicate that CS has risen in slave mode
		–	(9) OVERRUN	0x0	Indicate that an overrun has occurred when receiving data on the SPI interface
		–	(8) UNDERRUN	0x0	Indicate that an underrun has occurred when transmitting data on the SPI interface

Address	Register Name	Register Write	Register Read	Default	Description
		(4) TX_REQ_SET	–	N/A	Set TX_REQ status flag and clear internal TX buffer status
		(2) CS_RISE_CLEAR	–	N/A	Clear the CS rise status flag
		(1) OVERRUN_CLEAR	–	N/A	Clear the overrun status flag
		(0) UNDERRUN_CLEAR	–	N/A	Clear the underrun status flag
0x40000B0C	SPI0_TX_DATA	(31:0) TX_DATA	(31:0) TX_DATA	0x0	Single word buffer for data to be transmitted. When in master write or read_write mode, the transaction is started automatically
0x40000B10	SPI0_RX_DATA	–	(31:0) RX_DATA	0x0	Single word buffer for received data. When in master read mode, a new transaction is started automatically
0x40000B14	SPI0_RX_DATA_NO_START	–	(31:0) RX_DATA	0x0	Single word buffer for received data. Does not start a new transaction in master read mode, but does clear the RX_REQ flag
0x40000B18	SPI0_RX_DATA_MIRROR	–	(31:0) RX_DATA	0x0	Single word buffer for received data. Does not start a new transaction and does clear the RX_REQ flag
0x40000BFC	SPI0_ID_NUM	–	(19:16) SPI_NUMBER	0x0	SPI Instance number
		–	(15:8) SPI_MAJOR_ REVISION	0x2	SPI Major Revision number
		–	(7:0) SPI_MINOR_ REVISION	0x0	SPI Minor Revision number
0x40000C00	SPI1_CFG	(23) TX_DMA_ENABLE	(23) TX_DMA_ENABLE	0x0	Enable/disable the TX DMA request
		(22) RX_DMA_ENABLE	(22) RX_DMA_ENABLE	0x0	Enable/disable the RX DMA request
		(21) TX_END_INT_ ENABLE	(21) TX_END_INT_ ENABLE	0x0	Enable/disable the TX interrupt

Address	Register Name	Register Write	Register Read	Default	Description
		(20) TX_START_INT_ENABLE	(20) TX_START_INT_ENABLE	0x0	Enable/disable the TX interrupt
		(19) RX_INT_ENABLE	(19) RX_INT_ENABLE	0x0	Enable/disable the RX interrupt
		(18) CS_RISE_INT_ENABLE	(18) CS_RISE_INT_ENABLE	0x0	Enable/disable the CS rise interrupt (slave mode only)
		(17) OVERRUN_INT_ENABLE	(17) OVERRUN_INT_ENABLE	0x0	Enable/disable the overrun interrupt
		(16) UNDERRUN_INT_ENABLE	(16) UNDERRUN_INT_ENABLE	0x0	Enable/disable the underrun interrupt
		(14:13) MODE	(14:13) MODE	0x0	Select the SPI master mode (ignored in slave mode)
		(12:8) WORD_SIZE	(12:8) WORD_SIZE	0x0	Select the SPI word size (word size = SPI_WORD_SIZE + 1)
		(7:4) PRESCALE	(7:4) PRESCALE	0x0	Prescale the SPI clock for master mode
		(2) CLK_PHASE	(2) CLK_PHASE	0x0	Select the SPI clock phase
		(1) CLK_POLARITY	(1) CLK_POLARITY	0x0	Select the SPI clock polarity
		(0) SLAVE	(0) SLAVE	0x0	Use the SPI interface as master or slave
0x40000C04	SPI1_CTRL	–	(19) CS_STATUS	0x1	SPI CS status
		–	(18:17) MODE_STATUS	0x0	SPI mode status
		–	(16) ENABLE_STATUS	0x0	SPI enable status
		(9) CS_0	–	N/A	Lower the SPI chip-select line (master mode)
		(8) CS_1	–	N/A	Raise the SPI chip-select line (master mode)
		(7) MODE_NOP	–	N/A	Set mode to no operation

Address	Register Name	Register Write	Register Read	Default	Description
		(6) MODE_WRITE	–	N/A	Set mode to read operation
		(5) MODE_READ	–	N/A	Set mode to write operation
		(4) MODE_READ_WRITE	–	N/A	Set mode to read and write operation
		(3) START	–	N/A	Start a data transfer in master read mode
		(2) RESET	–	N/A	Reset the SPI interface
		(1) DISABLE	–	N/A	Disable the SPI interface
		(0) ENABLE	–	N/A	Enable the SPI interface
0x40000C08	SPI1_STATUS	–	(13) BUSY	0x0	Indicate that the reception or transmission of the data is ongoing
		–	(12) TX_REQ	0x1	Indicate that TX data can be written
		–	(11) RX_REQ	0x0	Indicate that RX data can be read
		–	(10) CS_RISE	0x0	Indicate that CS has risen in slave mode
		–	(9) OVERRUN	0x0	Indicate that an overrun has occurred when receiving data on the SPI interface
		–	(8) UNDERRUN	0x0	Indicate that an underrun has occurred when transmitting data on the SPI interface
		(4) TX_REQ_SET	–	N/A	Set TX_REQ status flag and clear internal TX buffer status
		(2) CS_RISE_CLEAR	–	N/A	Clear the CS rise status flag
		(1) OVERRUN_CLEAR	–	N/A	Clear the overrun status flag
		(0) UNDERRUN_CLEAR	–	N/A	Clear the underrun status flag
0x40000C0C	SPI1_TX_DATA	(31:0) TX_DATA	(31:0) TX_DATA	0x0	Single word buffer for data to be

Address	Register Name	Register Write	Register Read	Default	Description
					transmitted. When in master write or read_write mode, the transaction is started automatically
0x40000C10	SPI1_RX_DATA	–	(31:0) RX_DATA	0x0	Single word buffer for received data. When in master read mode, a new transaction is started automatically
0x40000C14	SPI1_RX_DATA_NO_START	–	(31:0) RX_DATA	0x0	Single word buffer for received data. Does not start a new transaction in master read mode, but does clear the RX_REQ flag
0x40000C18	SPI1_RX_DATA_MIRROR	–	(31:0) RX_DATA	0x0	Single word buffer for received data. Does not start a new transaction and does clear the RX_REQ flag
0x40000CFC	SPI1_ID_NUM	–	(19:16) SPI_NUMBER	0x0	SPI Instance number
		–	(15:8) SPI_MAJOR_ REVISION	0x2	SPI Major Revision number
		–	(7:0) SPI_MINOR_ REVISION	0x0	SPI Minor Revision number

A.13 PCM INTERFACE

Address	Register Name	Register Write	Register Read	Default	Description
0x40001000	PCM0_CFG	(25) TX_ACK_SEL	(25) TX_ACK_SEL	0x0	Select which TX word acknowleges the TX request
		(24) RX_ACK_SEL	(24) RX_ACK_SEL	0x0	Select which RX word acknowleges the RX request
		(23) TX_DMA_ENABLE	(23) TX_DMA_ENABLE	0x0	Enable/disable the TX DMA request
		(22) RX_DMA_ENABLE	(22) RX_DMA_ENABLE	0x0	Enable/disable the RX DMA request
		(21) TX_IOC_ENABLE	(21) TX_IOC_ENABLE	0x0	Enable/disable the TX IOC request
		(20) RX_IOC_ENABLE	(20) RX_IOC_ENABLE	0x0	Enable/disable the RX IOC request
		(19) RX_TX_INT_ENABLE	(19) RX_TX_INT_ENABLE	0x0	Enable/disable the RX_TX interrupt
		(17) OVERRUN_INT_EN	(17) OVERRUN_INT_EN	0x0	Enable/disable the overrun interrupt
		(16) UNDERRUN_INT_EN	(16) UNDERRUN_INT_EN	0x0	Enable/disable the underrun interrupt
		(15) CLK_POLARITY	(15) CLK_POLARITY	0x0	Select the PCM clock polarity
		(14) TX_DATA_ALIGN	(14) TX_DATA_ALIGN	0x0	Select the TX data alignment
		(13) RX_DATA_ALIGN	(13) RX_DATA_ALIGN	0x0	Select the RX data alignment
		(12:8) WORD_SIZE	(12:8) WORD_SIZE	0x8	Select the number of bits per PCM word
		(7) BIT_ORDER	(7) BIT_ORDER	0x0	Select whether the data is transmitted starting with the MSB or LSB
		(6) FRAME_ALIGN	(6) FRAME_ALIGN	0x0	Align the PCM frame signal to the first/last bit
		(5) FRAME_WIDTH	(5) FRAME_WIDTH	0x0	Use a long/short PCM frame signal
		(4:2) FRAME_LENGTH	(4:2) FRAME_LENGTH	0x0	Select the number of words per PCM frame

Address	Register Name	Register Write	Register Read	Default	Description
		(1) SUBFRAME	(1) SUBFRAME	0x0	Enable the frame duration for each word
		(0) SLAVE	(0) SLAVE	0x1	Use the PCM interface as a master/slave
0x40001004	PCM0_CTRL	–	(8) ENABLE_STATUS	0x0	PCM enable status
		(2) RESET	–	N/A	Reset the PCM interface
		(1) DISABLE	–	N/A	Disable the PCM interface
		(0) ENABLE	–	N/A	Enable the PCM interface
0x40001008	PCM0_STATUS	–	(12) BUSY	0x0	Indicate that the reception or transmission of the data is ongoing
		–	(11) TX_REQ	0x1	Indicate that TX data can be written
		–	(10) RX_REQ	0x0	Indicate that RX data can be read
		–	(9) OVERRUN	0x0	Indicate that an overrun occurred when receiving data
		–	(8) UNDERRUN	0x0	Indicate that an underrun occurred when transmitting data
		(1) OVERRUN_CLEAR	–	N/A	Clear the overrun status flag
		(0) UNDERRUN_CLEAR	–	N/A	Clear the underrun status flag
0x40001010	PCM0_TX_DATA0	(31:0) TX_DATA0	(31:0) TX_DATA0	0x0	Data to transmit on channel 0
0x40001014	PCM0_TX_DATA1	(31:0) TX_DATA1	(31:0) TX_DATA1	0x0	Data to transmit on channel 1
0x40001018	PCM0_RX_DATA0	–	(31:0) RX_DATA0	0x0	Data received on channel 0
0x4000101C	PCM0_RX_DATA1	–	(31:0) RX_DATA1	0x0	Data received on channel 1
0x400010FC	PCM0_ID_NUM	–	(19:16) PCM_NUMBER	0x0	PCM Instance number
		–	(15:8) PCM_MAJOR_REVISION	0x1	PCM Major Revision number
		–	(7:0) PCM_MINOR_REVISION	0x0	PCM Minor Revision number

A.14 I2C INTERFACE

Address	Register Name	Register Write	Register Read	Default	Description
0x40000D00	I2C0_CFG	(30) REPEATED_START_INT_ENABLE	(30) REPEATED_START_INT_ENABLE	0x0	Configure whether repeated start interrupts will be generated by the I2C interface for active transactions in slave mode
		(29) CONNECT_IN_STANDBY	(29) CONNECT_IN_STANDBY	0x0	Control if the I2C lines are connected when running on the standby clock
		(28) TX_DMA_ENABLE	(28) TX_DMA_ENABLE	0x0	Enable/disable the TX DMA request
		(27) RX_DMA_ENABLE	(27) RX_DMA_ENABLE	0x0	Enable/disable the RX DMA request
		(26) TX_INT_ENABLE	(26) TX_INT_ENABLE	0x0	Enable/disable the TX interrupt
		(25) RX_INT_ENABLE	(25) RX_INT_ENABLE	0x0	Enable/disable the RX interrupt
		(24) BUS_ERROR_INT_ENABLE	(24) BUS_ERROR_INT_ENABLE	0x0	Enable/disable the bus error interrupt
		(23) OVERRUN_INT_ENABLE	(23) OVERRUN_INT_ENABLE	0x0	Enable/disable the overrun interrupt
		(22) STOP_INT_ENABLE	(22) STOP_INT_ENABLE	0x0	Configure whether stop interrupts will be generated by the I2C interface
		(21) AUTO_ACK_ENABLE	(21) AUTO_ACK_ENABLE	0x0	Select whether acknowledgement is automatically generated or not
		(20:16) SLAVE_PRESCALE	(20:16) SLAVE_PRESCALE	0x0	Controls the number of SYSCLK wait cycles in case of clock streching (in slave mode) between the moment the data is put on the SDA line and the SCL line is released.
		(15:8) MASTER_PRESCALE	(15:8) MASTER_PRESCALE	0x0	Prescaler used to divide SYSCLK to the correct SCL frequency when operating in master mode. SCL is prescaled by (PRESCALE + 1) * 3.

Address	Register Name	Register Write	Register Read	Default	Description
		(7:1) SLAVE_ADDRESS	(7:1) SLAVE_ADDRESS	0x10	Set the I2C slave address for this device
		(0) SLAVE	(0) SLAVE	0x0	Select whether the I2C interface is enabled for slave mode or not
0x40000D04	I2C0_CTRL	–	(9) LAST_DATA_STATUS	0x0	I2C last data status
		–	(8) ENABLE_STATUS	0x0	I2C enable status
		(6) LAST_DATA	–	N/A	Indicate that the current data is the last byte of a data transfer
		(5) STOP	–	N/A	Issue a stop condition on the I2C interface bus
		(4) NACK	–	N/A	Issue a not acknowledge on the I2C interface bus
		(3) ACK	–	N/A	Issue an acknowledge on the I2C interface bus
		(2) RESET	–	N/A	Reset the I2C interface
		(1) DISABLE	–	N/A	Disable the I2C interface
		(0) ENABLE	–	N/A	Enable the I2C interface
0x40000D08	I2C0_ADDR_START	(7:1) ADDRESS	(7:1) ADDRESS	0x0	I2C address to use for the transaction
		(0) READ_WRITE	(0) READ_WRITE	0x0	Select whether a read or a write transaction is started
0x40000D0C	I2C0_STATUS	–	(26) STOP_OR_REPEATED_START_DETECTED	0x0	Indicate if STOP_DETECTED or REPEATED_START_DETECTED bit is set
		–	(25) REPEATED_START_DETECTED	0x0	Indicate if an I2C repeated start has been detected during an active transaction in slave mode

Address	Register Name	Register Write	Register Read	Default	Description
		–	(22) BUS_ERROR	0x0	Bus error status bit
		–	(21) BUSY	0x0	Indicate that the reception or transmission of the data is ongoing
		–	(20) START_PENDING	0x0	Master frame start pending status bit
		–	(19) MASTER_MODE	0x0	Master mode status bit
		–	(18) STOP_DETECTED	0x0	Indicate if an I2C stop bit has been detected
		–	(17) DATA_EVENT	0x0	Indicate that I2C interface either needs data to transmit or has received data
		–	(16) TX_REQ	0x1	Indicate that a TX data can be written
		–	(15) RX_REQ	0x0	Indicate that a RX data can be read
		–	(14) CLK_STRETCH	0x0	Clock stretching flag
		–	(13) LINE_FREE	0x1	Line free flag
		–	(12) ADDR_DATA	0x0	Address / Data byte
		–	(11) READ_WRITE	0x0	Read / Write frame
		–	(10) GEN_CALL	0x0	General call flag
		–	(9) ACK	0x0	Acknowledge status
		–	(8) OVERRUN	0x0	Indicate that an overrun has occurred when receiving data
		(4) TX_REQ_SET	–	N/A	Set TX_REQ status flag
		(3) REPEATED_START_ DETECTED_CLEAR	–	N/A	Clear REPEATED_START_ DETECTED status flag
		(2) STOP_DETECTED_ CLEAR	–	N/A	Clear STOP_DETECTED status flag
		(1) BUS_ERROR_CLEAR	–	N/A	Clear BUS_ERROR status flag

Address	Register Name	Register Write	Register Read	Default	Description
		(0) OVERRUN_CLEAR	-	N/A	Clear OVERRUN status flag
0x40000D10	I2C0_TX_DATA	(7:0) TX_DATA	(7:0) TX_DATA	0x0	Single byte buffer for data transmitted over the I2C interface
0x40000D14	I2C0_RX_DATA	-	(7:0) RX_DATA	0x0	Single byte buffer for data received over the I2C interface
0x40000D18	I2C0_RX_DATA_MIRROR				
0x40000DFC	I2C0_ID_NUM	-	(22) I2C_WATCHDOG	0x0	Implementation of the watchdog counter
		-	(21) I2C_DEBUG	0x0	Implementation of the debug interface
		-	(20) I2C_DMA	0x0	Implementation of the DMA interface
		-	(19:16) I2C_NUMBER	0x0	I2C instance number
		-	(15:8) I2C_MAJOR_REVISION	0x1	I2C Major Revision number
		-	(7:0) I2C_MINOR_REVISION	0x0	I2C Minor Revision number
0x40000E00	I2C1_CFG	(30) REPEATED_START_INT_ENABLE	(30) REPEATED_START_INT_ENABLE	0x0	Configure whether repeated start interrupts will be generated by the I2C interface for active transactions in slave mode
		(29) CONNECT_IN_STANDBY	(29) CONNECT_IN_STANDBY	0x0	Control if the I2C lines are connected when running on the standby clock
		(28) TX_DMA_ENABLE	(28) TX_DMA_ENABLE	0x0	Enable/disable the TX DMA request
		(27) RX_DMA_ENABLE	(27) RX_DMA_ENABLE	0x0	Enable/disable the RX DMA request
		(26) TX_INT_ENABLE	(26) TX_INT_ENABLE	0x0	Enable/disable the TX interrupt
		(25) RX_INT_ENABLE	(25) RX_INT_ENABLE	0x0	Enable/disable the RX interrupt
		(24) BUS_ERROR_INT_ENABLE	(24) BUS_ERROR_INT_ENABLE	0x0	Enable/disable the bus error interrupt

Address	Register Name	Register Write	Register Read	Default	Description
		(23) OVERRUN_INT_ENABLE	(23) OVERRUN_INT_ENABLE	0x0	Enable/disable the overrun interrupt
		(22) STOP_INT_ENABLE	(22) STOP_INT_ENABLE	0x0	Configure whether stop interrupts will be generated by the I2C interface
		(21) AUTO_ACK_ENABLE	(21) AUTO_ACK_ENABLE	0x0	Select whether acknowledgement is automatically generated or not
		(20:16) SLAVE_PRESCALE	(20:16) SLAVE_PRESCALE	0x0	Controls the number of SYSCLK wait cycles in case of clock streching (in slave mode) between the moment the data is put on the SDA line and the SCL line is released.
		(15:8) MASTER_PRESCALE	(15:8) MASTER_PRESCALE	0x0	Prescaler used to divide SYSCLK to the correct SCL frequency when operating in master mode. SCL is prescaled by (PRESCALE + 1) * 3.
		(7:1) SLAVE_ADDRESS	(7:1) SLAVE_ADDRESS	0x10	Set the I2C slave address for this device
		(0) SLAVE	(0) SLAVE	0x0	Select whether the I2C interface is enabled for slave mode or not
0x40000E04	I2C1_CTRL	–	(9) LAST_DATA_STATUS	0x0	I2C last data status
		–	(8) ENABLE_STATUS	0x0	I2C enable status
		(6) LAST_DATA	–	N/A	Indicate that the current data is the last byte of a data transfer
		(5) STOP	–	N/A	Issue a stop condition on the I2C interface bus
		(4) NACK	–	N/A	Issue a not acknowledge on the I2C interface bus
		(3) ACK	–	N/A	Issue an acknowledge on the I2C

Address	Register Name	Register Write	Register Read	Default	Description
					interface bus
		(2) RESET	–	N/A	Reset the I2C interface
		(1) DISABLE	–	N/A	Disable the I2C interface
		(0) ENABLE	–	N/A	Enable the I2C interface
0x40000E08	I2C1_ADDR_START	(7:1) ADDRESS	(7:1) ADDRESS	0x0	I2C address to use for the transaction
		(0) READ_WRITE	(0) READ_WRITE	0x0	Select whether a read or a write transaction is started
0x40000E0C	I2C1_STATUS	–	(26) STOP_OR_ REPEATED_START_ DETECTED	0x0	Indicate if STOP_DETECTED or REPEATED_START_DETECTED bit is set
		–	(25) REPEATED_ START_DETECTED	0x0	Indicate if an I2C repeated start has been detected during an active transaction in slave mode
		–	(22) BUS_ERROR	0x0	Bus error status bit
		–	(21) BUSY	0x0	Indicate that the reception or transmission of the data is ongoing
		–	(20) START_PENDING	0x0	Master frame start pending status bit
		–	(19) MASTER_MODE	0x0	Master mode status bit
		–	(18) STOP_DETECTED	0x0	Indicate if an I2C stop bit has been detected
		–	(17) DATA_EVENT	0x0	Indicate that I2C interface either needs data to transmit or has received data
		–	(16) TX_REQ	0x1	Indicate that a TX data can be written
		–	(15) RX_REQ	0x0	Indicate that a RX data can be read
		–	(14) CLK_STRETCH	0x0	Clock stretching flag
		–	(13) LINE_FREE	0x1	Line free flag

Address	Register Name	Register Write	Register Read	Default	Description
		–	(12) ADDR_DATA	0x0	Address / Data byte
		–	(11) READ_WRITE	0x0	Read / Write frame
		–	(10) GEN_CALL	0x0	General call flag
		–	(9) ACK	0x0	Acknowledge status
		–	(8) OVERRUN	0x0	Indicate that an overrun has occurred when receiving data
		(4) TX_REQ_SET	–	N/A	Set TX_REQ status flag
		(3) REPEATED_START_ DETECTED_CLEAR	–	N/A	Clear REPEATED_START_ DETECTED status flag
		(2) STOP_DETECTED_ CLEAR	–	N/A	Clear STOP_DETECTED status flag
		(1) BUS_ERROR_CLEAR	–	N/A	Clear BUS_ERROR status flag
		(0) OVERRUN_CLEAR	–	N/A	Clear OVERRUN status flag
0x40000E10	I2C1_TX_DATA	(7:0) TX_DATA	(7:0) TX_DATA	0x0	Single byte buffer for data transmitted over the I2C interface
0x40000E14	I2C1_RX_DATA	–	(7:0) RX_DATA	0x0	Single byte buffer for data received over the I2C interface
0x40000E18	I2C1_RX_DATA_MIRROR				
0x40000EFC	I2C1_ID_NUM	–	(22) I2C_WATCHDOG	0x0	Implementation of the watchdog counter
		–	(21) I2C_DEBUG	0x0	Implementation of the debug interface
		–	(20) I2C_DMA	0x0	Implementation of the DMA interface
		–	(19:16) I2C_NUMBER	0x0	I2C instance number
		–	(15:8) I2C_MAJOR_ REVISION	0x1	I2C Major Revision number
		–	(7:0) I2C_MINOR_ REVISION	0x0	I2C Minor Revision number

A.15 TEST CONTROL INTERFACE

Address	Register Name	Register Write	Register Read	Default	Description

A.16 UART INTERFACE

Address	Register Name	Register Write	Register Read	Default	Description
0x40000F00	UART0_CFG	(21) TX_DMA_ENABLE	(21) TX_DMA_ENABLE	0x0	Enable/disable the TX DMA request
		(20) RX_DMA_ENABLE	(20) RX_DMA_ENABLE	0x0	Enable/disable the RX DMA request
		(19) TX_END_INT_ ENABLE	(19) TX_END_INT_ ENABLE	0x0	Enable/disable the TX end interrupt
		(18) TX_START_INT_ ENABLE	(18) TX_START_INT_ ENABLE	0x0	Enable/disable the TX start interrupt
		(17) RX_INT_ENABLE	(17) RX_INT_ENABLE	0x0	Enable/disable the RX interrupt
		(16) OVERRUN_INT_ ENABLE	(16) OVERRUN_INT_ ENABLE	0x0	Enable/disable the overrun interrupt
		(15:0) CNT_STEP	(15:0) CNT_STEP	0x0	Counter step size that configures the baud rate
0x40000F04	UART0_CTRL	–	(8) ENABLE_STATUS	0x0	UART enable status
		(2) RESET	–	N/A	Reset the UART interface
		(1) DISABLE	–	N/A	Disable the UART interface
		(0) ENABLE	–	N/A	Enable the UART interface
0x40000F08	UART0_STATUS	–	(12) TX_BUSY	0x0	Indicate that a TX transaction is ongoing
		–	(11) RX_BUSY	0x0	Indicate that a RX transaction is ongoing
		–	(10) TX_REQ	0x1	Indicate that a TX data can be written
		–	(9) RX_REQ	0x0	Indicate that a RX data can be read
		–	(8) OVERRUN	0x0	Indicate that an overrun occurred when receiving data
		(0) OVERRUN_CLEAR	–	N/A	Clear the overrun status flag
0x40000F0C	UART0_TX_DATA	(7:0) TX_DATA	(7:0) TX_DATA	0x0	Transmitted data

Address	Register Name	Register Write	Register Read	Default	Description
0x40000F10	UART0_RX_DATA	–	(7:0) RX_DATA	0x0	Received data
0x40000FFC	UART0_ID_NUM	–	(19:16) UART_ NUMBER	0x0	UART Instance number
		–	(15:8) UART_MAJOR_ REVISION	0x1	UART Major Revision number
		–	(7:0) UART_MINOR_ REVISION	0x0	UART Minor Revision number

A.17 PWM INTERFACE

Address	Register Name	Register Write	Register Read	Default	Description
0x40001100 – 0x40001110	PWM_PERIOD_*	(11:0) PERIOD	(11:0) PERIOD	0x0	PWM period
0x40001114	PWM_CTRL	(28:24) RESET	–	N/A	PWM[4:0] channel reset
		–	(20:16) ENABLE_ STATUS	0x0	PWM enable status
		(12:8) DISABLE	–	N/A	Disable the PWM[4:0] channel
		(4:0) ENABLE	–	N/A	Enable the PWM[4:0] channel
0x40001118 – 0x40001124	PWM_OFFSET_*	(12) OFFSET_ENABLE	(12) OFFSET_ENABLE	0x0	Enable/disable the PWM offset function
		(11:0) OFFSET	(11:0) OFFSET	0x0	PWM(0) to PWM(i) offset
0x40001128 – 0x40001138	PWM_HIGH_*	(19:8) HIGH	(19:8) HIGH	0x0	PWM high duration
		(7:0) HIGH_FRACTIONAL	(7:0) HIGH_ FRACTIONAL	0x0	PWM high fractional
0x400011FC	PWM_ID_NUM	–	(18:16) PWM_NUMBER	0x0	PWM number of channel
		–	(15:8) PWM_MAJOR_ REVISION	0x1	PWM Major Revision number
		–	(7:0) PWM_MINOR_ REVISION	0x0	PWM Minor Revision number

A.18 CRC GENERATOR CONTROL

Address	Register Name	Register Write	Register Read	Default	Description
0x40001600	CRC_CFG	(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
0x40001604	CRC_VALUE	(31:0) CURRENT_CRC	(31:0) CURRENT_CRC	0xFFFF	CRC generator value: Write 0xFFFFFFF (32) or 0xFFFF (CCITT) to initialize the CRC, read provides the current CRC value.
0x40001608	CRC_ADD_1	(0) CRC_ADD_1	–	N/A	Add 1 bit to the CRC calculation
0x4000160C	CRC_ADD_8	(7:0) CRC_ADD_8	–	N/A	Add 1 byte (8 bits) to the CRC calculation
0x40001610	CRC_ADD_16	(15:0) CRC_ADD_16	–	N/A	Add 1 half-word (16 bits) to the CRC calculation
0x40001614	CRC_ADD_24	(23:0) CRC_ADD_24	–	N/A	Add 3 bytes (24 bits) to the CRC calculation
0x40001618	CRC_ADD_32	(31:0) CRC_ADD_32	–	N/A	Add 1 word (32 bits) to the CRC calculation
0x4000161C	CRC_FINAL	–	(31:0) FINAL_CRC	0x0	CRC generator final value: After XOR for CCITT or byte reversal for CRC-32
0x400016FC	CRC_ID_NUM	–	(15:8) CRC_MAJOR_ REVISION	0x1	CRC Major Revision number
		–	(7:0) CRC_MINOR_ REVISION	0x0	CRC Minor Revision number

A.19 ASCC

Address	Register Name	Register Write	Register Read	Default	Description
0x40001700	ASCC_CTRL	(8) PHASE_CNT_START_NO_WAIT	-	N/A	Start the asynchronous clock phase counter mechanism without waiting on a sync pulse
		-	(7) PERIOD_CNT_STATUS	0x0	Asynchronous clock period counter status
		(6) PERIOD_CNT_STOP	-	N/A	Stop the asynchronous clock period counter mechanism
		(5) PERIOD_CNT_START	-	N/A	Start the asynchronous clock period counter mechanism
		-	(4) PHASE_CNT_MISSED_STATUS	0x0	Asynchronous clock phase counter missed status
		-	(3) PHASE_CNT_STATUS	0x0	Asynchronous clock phase counter status
		(2) PHASE_CNT_STOP	-	N/A	Stop the asynchronous clock phase counter mechanism
		(1) PHASE_CNT_START	-	N/A	Start the asynchronous clock phase counter mechanism and wait for sync pulse
		(0) CNT_RESET	-	N/A	Reset asynchronous clock counter
0x40001704	ASCC_CFG	(3:0) PERIODS_CFG	(3:0) PERIODS_CFG	0x0	Defines over how many asynchronous clock periods the period counter measures
0x40001708	ASCC_CNT	-	(11:0) CNT	0x0	Asynchronous clock counter value
0x4000170C	ASCC_PHASE_CNT	(15:0) PHASE_CNT	(15:0) PHASE_CNT	0x0	Asynchronous clock phase counter value
0x40001710	ASCC_PERIOD_CNT	(15:0) PERIOD_CNT	(15:0) PERIOD_CNT	0x0	Asynchronous clock period counter

Address	Register Name	Register Write	Register Read	Default	Description
					value
0x400017FC	ASCC_ID_NUM	–	(15:8) ASCC_MAJOR_REVISION	0x1	ASCC Major Revision number
		–	(7:0) ASCC_MINOR_REVISION	0x0	ASCC Minor Revision number

A.20 ACS

Address	Register Name	Register Write	Register Read	Default	Description
0x40001B00	ACS_BG_CTRL	–	(31) READY	0x0	Bandgap ready
		(28:24) SLOPE_ITRIM	(28:24) SLOPE_ITRIM	0x17	Current temperature coefficient trimming
		(21:16) ITRIM	(21:16) ITRIM	0x24	Reference current trimming
		(13) SEL_VTRIM_SRC	(13) SEL_VTRIM_SRC	0x1	Select trim source for SLOPE_VTRIM and VTRIM
		(12:8) SLOPE_VTRIM	(12:8) SLOPE_VTRIM	0x18	Voltage temperature coefficient trimming
		(5:0) VTRIM	(5:0) VTRIM	0x15	Reference voltage trimming (5 mV steps)
0x40001B04	ACS_VCC_CTRL	–	(24) READY	0x0	Supply ready (only makes sense in test mode)
		(21:16) VTRIM_LIMIT	(21:16) VTRIM_LIMIT	0x1F	Max VTRIM value that can be used. If VTRIM value is greater it will be limited to this value
		(15:12) ICH_TRIM	(15:12) 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
		(5:0) VTRIM	(5:0) VTRIM	0x14	Output voltage trimming configuration in 10 mV steps
0x40001B08	ACS_VDDCP_CTRL	–	(24) READY	0x0	Supply ready
		(13) COMP_ENABLE	(13) COMP_ENABLE	0x1	Force cp comparator enable

Address	Register Name	Register Write	Register Read	Default	Description
		(11:8) CPCLK_FREQ	(11:8) CPCLK_FREQ	0x2	Charge Pump clock frequency during power down modes
		(1:0) PTRIM	(1:0) PTRIM	0x3	Output power trimming
0x40001B0C	ACS_VDDC_CTRL	–	(24) READY	0x0	Supply ready
		(21:16) STANDBY_VTRIM	(21:16) STANDBY_VTRIM	0x2D	VDDC standby voltage trimming (10 mV steps)
		(12) ENABLE_LOW_BIAS	(12) ENABLE_LOW_BIAS	0x0	Low power mode control
		(8) SLEEP_CLAMP	(8) SLEEP_CLAMP	0x0	Sleep mode clamp control
		(5:0) VTRIM	(5:0) VTRIM	0x23	Output voltage trimming configuration in 10 mV steps
0x40001B10	ACS_VDDM_CTRL	–	(24) READY	0x0	Supply ready
		(21:16) STANDBY_VTRIM	(21:16) STANDBY_VTRIM	0x2D	VDDM standby voltage trimming (10 mV steps)
		(12) ENABLE_LOW_BIAS	(12) ENABLE_LOW_BIAS	0x0	Low power mode control
		(8) SLEEP_CLAMP	(8) SLEEP_CLAMP	0x0	Sleep mode clamp control
		(5:0) VTRIM	(5:0) VTRIM	0x28	Output voltage trimming configuration in 10 mV steps
0x40001B14	ACS_VDDPA_CTRL	(19:16) INITIAL_VTRIM	(19:16) INITIAL_VTRIM	0x0	Initial output voltage trimming configuration in 10 mV steps
		(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

Address	Register Name	Register Write	Register Read	Default	Description
0x40001B18	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
0x40001B1C	ACS_VDDFLASH_CTRL	-	(24) READY	0x0	Supply ready
		(11) MASK_READY	(11) MASK_READY	0x0	Apply a mask to reset logic to ignore VDDFLASH ready signal
		(10) SOFT_START	(10) SOFT_START	0x1	Current limiter threshold setting
		(9) ENABLE_LIMITER	(9) ENABLE_LIMITER	0x1	Enable current limiter circuit
		(8) ENABLE	(8) ENABLE	0x1	Enable control
		(5:0) VTRIM	(5:0) VTRIM	0x28	Output voltage trimming configuration in 25 mV steps
0x40001B20	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) VDDACS_VTRIM	(10:9) VDDACS_ VTRIM	0x3	VDDACS regulator voltage trimming
		(8) VDDTRET_ENABLE	(8) VDDTRET_ENABLE	0x0	Enables the VDDT retention power (activate switch from VDDACS)
		(2:1) VDDCRET_VTRIM	(2:1) VDDCRET_ VTRIM	0x3	VDDCRET Retention regulator voltage trimming
		(0) VDDCRET_ENABLE	(0) VDDCRET_ENABLE	0x0	Enables the VDDCRET Retention regulator
0x40001B24	ACS_RCOSC_CTRL	(26:25) RC_FSEL	(26:25) RC_FSEL	0x0	Select RC oscillator frequency

Address	Register Name	Register Write	Register Read	Default	Description
		(24) RC_OSC_EN	(24) RC_OSC_EN	0x0	Enable/Disable the RC Oscillator
		(23) RC_FTRIM_FLAG	(23) RC_FTRIM_FLAG	0x0	RC Oscillator Trimming flag
		(22) RC_FTRIM_ADJ	(22) RC_FTRIM_ADJ	0x0	Adjust RC oscillator frequency range
		(21:16) RC_FTRIM	(21:16) RC_FTRIM	0x20	RC oscillator frequency trimming
		(15:14) RC32_VDDLOC_ITRIM	(15:14) RC32_VDDLOC_ITRIM	0x0	VDDLOC current trimming
		(13:12) RC32_VDDLOC_VTRIM	(13:12) RC32_VDDLOC_VTRIM	0x0	VDDLOC voltage trimming
		(9) RC32_VDDLOC_OD_EN	(9) RC32_VDDLOC_OD_EN	0x0	Enable the overdrive of vddloc with VBAT
		(8) RC32_OSC_EN	(8) RC32_OSC_EN	0x0	Enable/Disable the 32 kHz RC Oscillator
		(7) RC32_TEMP_COMP_EN	(7) RC32_TEMP_COMP_EN	0x0	Enable/Disable the 32 kHz RC Oscillator
		(6) RC32_FTRIM_ADJ	(6) RC32_FTRIM_ADJ	0x0	Adjust 32 kHz RC oscillator frequency range
		(5:0) RC32_FTRIM	(5:0) RC32_FTRIM	0x20	32 kHz RC oscillator frequency trimming
0x40001B28	ACS_XTAL32K_CTRL	(26) XTAL_N_OK_RESET	–	N/A	Reset Xtal not ok sticky flag
		–	(25) XTAL_N_OK	0x0	Xtal not ready sticky flag
		–	(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	0x1	Xtal enable amplitude control (regulation)
		(16) FORCE_READY	(16) FORCE_READY	0x0	Xtal bypass the ready detector

Address	Register Name	Register Write	Register Read	Default	Description
		(13:8) CLOAD_TRIM	(13:8) CLOAD_TRIM	0x16	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
0x40001B2C	ACS_ACOMP_CFG	(22:20) ACOMP_PRE_SEL_ODD	(22:20) ACOMP_PRE_SEL_ODD	0x0	Pre selection of odd GPIOs
		(18:16) ACOMP_PRE_SEL_EVEN	(18:16) ACOMP_PRE_SEL_EVEN	0x0	Pre selection of even GPIOs
		(14:12) ACOMP_IN_P	(14:12) ACOMP_IN_P	0x2	Defines the positive input signal
		(10:8) ACOMP_IN_N	(10:8) ACOMP_IN_N	0x2	Defines the negative input signal
		(6:4) ACOMP_HYST	(6:4) ACOMP_HYST	0x0	Hysteresis level
		(2:1) MODE	(2:1) MODE	0x1	Power mode selection
		(0) ENABLE	(0) ENABLE	0x0	Enable signal
0x40001B30	ACS_ACOMP_OUT	–	(0) ACOMP_OUT	0x0	Comparison result
0x40001B34	ACS_SDAC_CFG	(2) SDAC_TO_GPIO7	(2) SDAC_TO_GPIO7	0x0	GPIO7 drive by SDAC
		(1) SDAC_GAIN	(1) SDAC_GAIN	0x0	Sets the SDAC buffer gain
		(0) SDAC_EN	(0) SDAC_EN	0x0	Enable signal for SDAC buffer
0x40001B38	ACS_WEDAC_CTRL	(5:0) WEDAC	(5:0) WEDAC	0x26	WEDAC setting
0x40001B3C	ACS_RTC_CFG	(6:4) RTC_CLOCK_SRC	(6:4) RTC_CLOCK_SRC	0x0	Select the RTC Clock event source
		(2:0) CLK_SRC_SEL	(2:0) CLK_SRC_SEL	0x0	Select the RTC, standby, bb timer, and sensor block clock source
0x40001B40	ACS_RTC_CTRL	–	(18) ENABLE_CLOCK_EVENT_STATUS	0x0	Status of the RTC clock event enable
		–	(17) ENABLE_ALARM_	0x0	Status of the RTC alarm event enable

Address	Register Name	Register Write	Register Read	Default	Description
			EVENT_STATUS		
		–	(16) ENABLE_STATUS	0x0	Status of the RTC enable
		(7) FORCE_CLOCK	–	N/A	Force a clock on RTC timer (Test Purpose)
		(6) DISABLE_CLOCK_EVENT	–	N/A	Disable clock event and its interrupt
		(5) ENABLE_CLOCK_EVENT	–	N/A	Enable clock event and its interrupt
		(4) DISABLE_ALARM_EVENT	–	N/A	Disable alarm event and its interrupt
		(3) ENABLE_ALARM_EVENT	–	N/A	Enable alarm event and its interrupt
		(2) RESET	–	N/A	Reset the RTC timer
		(1) DISABLE	–	N/A	Disable counter and RTC interrupt every 1s
		(0) ENABLE	–	N/A	Enable counter and RTC interrupt every 1s
0x40001B44	ACS_RTC_COUNT_THRES	(31:0) THRESHOLD	(31:0) THRESHOLD	0xFFFFFFFF	Compare value for the RTC counter
0x40001B48	ACS_RTC_COUNT	–	(31:0) VALUE	0x0	RTC timer current value
0x40001B4C	ACS_RTC_SECONDS	–	(16:0) VALUE	0x0	RTC timer current value in seconds
0x40001B50	ACS_RTC_COUNT_LOAD	(31:0) VALUE	–	N/A	Load the RTC timer current value
0x40001B54	ACS_BB_TIMER_CTRL	(0) BB_TIMER_NRESET	(0) BB_TIMER_NRESET	0x0	nReset signal for the baseband timer
0x40001B58	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

Address	Register Name	Register Write	Register Read	Default	Description
0x40001B5C	ACS_PWR_MODES_CTRL	(31:0) POWER_MODE	–	N/A	32-bit key to enter STANDBY, SLEEP, or DEEP_SLEEP mode. This register must be written using a 32-bit access.
0x40001B60	ACS_SLEEP_MODE_CFG	(2) BG_ENABLE	(2) BG_ENABLE	0x0	Keep Band-gap enabled during sleep
		(1) VCC_ENABLE	(1) VCC_ENABLE	0x0	Keep VCC low power enabled during sleep
		(0) VDDCP_ENABLE	(0) VDDCP_ENABLE	0x0	Keep VDDCP charge pump enabled during sleep
0x40001B64	ACS_WAKEUP_CTRL	–	(27) RTC_OVERFLOW_ WAKEUP	0x0	Indicate if RTC overflow has triggered a wake-up event
		–	(26) RTC_CLOCK_ WAKEUP	0x0	Indicate if RTC clock has triggered a wake-up event
		–	(25) RTC_ALARM_ WAKEUP	0x0	Indicate if RTC alarm has triggered a wake-up event
		–	(24) THRESHOLD_ WAKEUP	0x0	Indicate if the sensor interface threshold has triggered a wake-up event
		–	(23) FIFO_FULL_ WAKEUP	0x0	Indicate if the sensor interface FIFO has triggered a wake-up event
		–	(22) ACOMP_WAKEUP	0x0	Indicate if ACOMP has triggered a wake-up event
		–	(21) DCDC_ OVERLOAD_WAKEUP	0x0	Indicate if DCDC overload has triggered a wake-up event
		–	(20) BB_TIMER_ WAKEUP	0x0	Indicate if baseband timer has triggered a wake-up event
		–	(19) GPIO3_WAKEUP	0x0	Indicate if GPIO3 has triggered a wake-up event

Address	Register Name	Register Write	Register Read	Default	Description
		–	(18) GPIO2_WAKEUP	0x0	Indicate if GPIO2 has triggered a wake-up event
		–	(17) GPIO1_WAKEUP	0x0	Indicate if GPIO1 has triggered a wake-up event
		–	(16) GPIO0_WAKEUP	0x0	Indicate if GPIO0 has triggered a wake-up event
		(11) THRESHOLD_ WAKEUP_CLEAR	–	N/A	Reset the sticky WAKEUP_ THRESHOLD_EVENT flag
		(10) FIFO_FULL_ WAKEUP_CLEAR	–	N/A	Reset the sticky WAKEUP_FIFO_ FULL_EVENT flag
		(9) ACOMP_WAKEUP_ CLEAR	–	N/A	Reset the sticky WAKEUP_ACOMP_ EVENT flag
		(8) DCDC_OVERLOAD_ WAKEUP_CLEAR	–	N/A	Reset the sticky WAKEUP_DCDC_ OVERLOAD flag
		(7) RTC_OVERFLOW_ WAKEUP_CLEAR	–	N/A	Reset the sticky WAKEUP_RTC_ OVERFLOW_EVENT flag
		(6) RTC_CLOCK_ WAKEUP_CLEAR	–	N/A	Reset the sticky WAKEUP_RTC_ CLOCK_EVENT flag
		(5) RTC_ALARM_ WAKEUP_CLEAR	–	N/A	Reset the sticky WAKEUP_RTC_ ALARM_EVENT flag
		(4) BB_TIMER_WAKEUP_ CLEAR	–	N/A	Reset the sticky WAKEUP_BB_ TIMER_EVENT flag
		(3) GPIO3_WAKEUP_ CLEAR	–	N/A	Reset the sticky WAKEUP_GPIO3_ EVENT flag
		(2) GPIO2_WAKEUP_ CLEAR	–	N/A	Reset the sticky WAKEUP_GPIO2_ EVENT flag
		(1) GPIO1_WAKEUP_ CLEAR	–	N/A	Reset the sticky WAKEUP_GPIO1_ EVENT flag

Address	Register Name	Register Write	Register Read	Default	Description
		(0) GPIO0_WAKEUP_CLEAR	-	N/A	Reset the sticky WAKEUP_GPIO0_EVENT flag
0x40001B68	ACS_WAKEUP_CFG	(18:16) DELAY	(18:16) DELAY	0x4	Delay from VDDD ready to digital clock enable (power of 2)
		(11) DCDC_OVERLOAD_EN	(11) DCDC_OVERLOAD_EN	0x0	Enable / Disable the Wake-up functionality on the DCDC overload flag
		(9) FIFO_FULL_EN	(9) FIFO_FULL_EN	0x0	Enable the wake-up on full FIFO
		(8) RTC_OVERFLOW_EN	(8) RTC_OVERFLOW_EN	0x0	Enable / Disable the Wake-up functionality on RTC overflow flag
		(7) GPIO3_POL	(7) GPIO3_POL	0x0	Wake-up polarity on the GPIO3 pad
		(6) GPIO2_POL	(6) GPIO2_POL	0x0	Wake-up polarity on the GPIO2 pad
		(5) GPIO1_POL	(5) GPIO1_POL	0x0	Wake-up polarity on the GPIO1 pad
		(4) GPIO0_POL	(4) GPIO0_POL	0x0	Wake-up polarity on the GPIO0 pad
		(3) GPIO3_EN	(3) GPIO3_EN	0x0	Enable the wake-up functionality on the GPIO3 pad
		(2) GPIO2_EN	(2) GPIO2_EN	0x0	Enable the wake-up functionality on the GPIO2 pad
		(1) GPIO1_EN	(1) GPIO1_EN	0x0	Enable the wake-up functionality on the GPIO1 pad
		(0) GPIO0_EN	(0) GPIO0_EN	0x0	Enable the wake-up functionality on the GPIO0 pad
0x40001B6C	ACS_WAKEUP_STATE	-	(19:16) 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)

Address	Register Name	Register Write	Register Read	Default	Description
0x40001B70	ACS_BOOT_CFG	(8) PADS_RETENTION_EN	(8) PADS_RETENTION_EN	0x0	Enable / Disable the retention mode of the pads
		(6) BOOT_ROT_BYPASS	(6) BOOT_ROT_BYPASS	0x0	Boot bypass execution of root of trust
		(5) BOOT_PWR_CAL_BYPASS	(5) BOOT_PWR_CAL_BYPASS	0x0	Boot bypass execution of system calibration
		–	(4:3) RC_CLOCK_FSEL	0x0	RC oscillator clock multiplier read only flag (mirror of CLOCK_MULT of ACS_RCOSC_CTRL register)
		–	(2) RC_FTRIM_FLAG	0x0	RC oscillator trimming read only flag (mirror of FTRIM_FLAG of ACS_RCOSC_CTRL register
		(1:0) BOOT_SELECT	(1:0) BOOT_SELECT	0x0	Boot selection to indicate boot source
0x40001B74	ACS_BOOT_GP_DATA	(31:0) GP_DATA	(31:0) GP_DATA	0x0	32-bit General-Purpose RW Data
0x40001B78	ACS_RESET_STATUS	–	(25) CCAO_REBOOT_RESET_FLAG	0x1	Sticky flag that detects that a CryptoCell Alway ON reboot reset occurred
		–	(24) WRONG_STATE_RESET_FLAG	0x1	Sticky flag that detects that a wrong state reset occurred
		–	(23) TIMEOUT_RESET_FLAG	0x1	Sticky flag that detects that a timeout in the power up sequence occurred
		–	(22) CLK_DET_RESET_FLAG	0x1	Sticky flag that detects that a clock detector reset occurred
		–	(21) VDDFLASH_RESET_FLAG	0x1	Sticky flag that detects that a VDDFLASH reset occurred (triggered by VDDFLASH_ready = 0)
		–	(20) VDDM_RESET_FLAG	0x1	Sticky flag that detects that a VDDM reset occurred (triggered by VDDM_ready = 0)

Address	Register Name	Register Write	Register Read	Default	Description
		–	(19) VDDC_RESET_FLAG	0x1	Sticky flag that detects that a VDDC reset occurred (triggered by VDDC_ready = 0)
		–	(18) BG_VREF_RESET_FLAG	0x1	Sticky flag that detects that a Bandagap reference voltage reset occurred
		–	(17) PAD_RESET_FLAG	0x1	Sticky flag that detects that a reset occurred due to pad NRESET
		–	(16) POR_RESET_FLAG	0x1	Sticky flag that detects that a POR reset occurred
		(9) CCAO_REBOOT_RESET_FLAG_CLEAR	–	N/A	Reset the sticky CCAO_REBOOT_RESET flag.
		(8) WRONG_STATE_RESET_FLAG_CLEAR	–	N/A	Reset the sticky WRONG_STATE_RESET flag.
		(7) TIMEOUT_RESET_FLAG_CLEAR	–	N/A	Reset the sticky TIMEOUT_RESET flag.
		(6) CLK_DET_RESET_FLAG_CLEAR	–	N/A	Reset the sticky CLK_DET_RESET flag.
		(5) VDDFLASH_RESET_FLAG_CLEAR	–	N/A	Reset the sticky VDDFLASH_RESET flag.
		(4) VDDM_RESET_FLAG_CLEAR	–	N/A	Reset the sticky VDDM_RESET flag.
		(3) VDDC_RESET_FLAG_CLEAR	–	N/A	Reset the sticky VDDC_RESET flag.
		(2) BG_VREF_RESET_FLAG_CLEAR	–	N/A	Reset the sticky BG_VREF_RESET flag.
		(1) PAD_RESET_FLAG_CLEAR	–	N/A	Reset the sticky PAD_RESET flag.

Address	Register Name	Register Write	Register Read	Default	Description
		(0) POR_RESET_FLAG_CLEAR	-	N/A	Reset the sticky POR_RESET flag.
0x40001B7C	ACS_AOUT_CTRL	(21) RTC_CLOCK_GPIO0_STOP_EDGE	(21) RTC_CLOCK_GPIO0_STOP_EDGE	0x0	Stop edge for RTC clock output on AOUT
		(20:19) RTC_CLOCK_GPIO0_STOP_SRC	(20:19) RTC_CLOCK_GPIO0_STOP_SRC	0x0	Stop source for RTC clock output on AOUT
		(18:16) RTC_CLOCK_GPIO0_START	(18:16) RTC_CLOCK_GPIO0_START	0x0	Start event for RTC clock output on AOUT (RTC prescaler and counter need to be enabled)
		(13) AOUT_IOUT_SEL_TO_GPIO	(13) AOUT_IOUT_SEL_TO_GPIO	0x1	Selection between AOUT voltage or PTAT current source to GPIO
		(12:8) AOUT_TO_GPIO	(12:8) AOUT_TO_GPIO	0x10	Select to which GPIO the AOUT voltage or PTAT current provided
		(5:0) TEST_AOUT	(5:0) TEST_AOUT	0x0	AOUT test signal selection
0x40001B80	ACS_TEMP_SENSOR_CFG	(1) DUTY_TEMP_SENS	(1) DUTY_TEMP_SENS	0x0	Duty cycling temperature sensor
		(0) ENABLE	(0) ENABLE	0x0	Internal temperature sensor enable
0x40001B84	ACS_TEMP_CURR_CFG	(19:16) CURRENT_VALUE	(19:16) CURRENT_VALUE	0x9	Temperature current value
		(13:8) CURRENT_TRIM	(13:8) CURRENT_TRIM	0x0	Temperature current trimming
		(7:4) GPIO_IN_USE	(7:4) GPIO_IN_USE	0x0	GPIO from which the duty cycle is used
		(1) DUTY_CURR	(1) DUTY_CURR	0x0	Duty cycling current enable
		(0) ENABLE	(0) ENABLE	0x0	Temperature current enable
0x40001B88	ACS_GP_DATA	(31:0) GP_DATA	(31:0) GP_DATA	0x0	32-bit General-Purpose RW Data
0x40001B8C	ACS_PWR_CTRL	(27) SENSOR_PWR_EN	(27) SENSOR_PWR_EN	0x1	Sensor power control
		(26) SENSOR_ISOLATE	(26) SENSOR_	0x0	Sensor isolation control

Address	Register Name	Register Write	Register Read	Default	Description
			ISOLATE		
		(25) CCAO_PWR_EN	(25) CCAO_PWR_EN	0x1	CryptoCell always on power control
		(24) CCAO_ISOLATE	(24) CCAO_ISOLATE	0x0	CryptoCell always on isolation control
		(23:0) POWER_KEY	–	N/A	Write a key to enable the write to power controls
0x40001B90	ACS_PWM_AO_CFG	(15:8) HIGH	(15:8) HIGH	0x0	PWM high duty cycle
		(7:0) PERIOD	(7:0) PERIOD	0x0	PWM period
0x40001B94	ACS_PWM_AO_CTRL	–	(8) ENABLE_STATUS	0x0	Status of the PWM enable
		(2) RESET	–	N/A	Reset the PWM
		(1) DISABLE	–	N/A	Disable the PWM
		(0) ENABLE	–	N/A	Enable the PWM
0x40001B98	ACS_PWM_AO_COUNT	–	(7:0) COUNTER	0x0	PWM counter
0x40001B9C	ACS_TEST	–	(23:16) JIC_BYTE0_RO	0xFF	JIC read only register bits (returning signals from analog part: tied to 1)
		–	(9) DEBUG_ENABLE_FLAG	0x0	Status flag of debug enable
		–	(8) RTC_CLK_VALUE	0x0	Value of RTC clock
		–	(2) EN_TEST_PAD_STATUS	0x0	EN_TEST pad state
		–	(1) EN_TEST_CTRL_STATUS	0x0	EN_TEST_CTRL state
		(0) EN_TEST_CTRL	–	N/A	EN_TEST_CTRL can be written only one time

A.21 AHB REGISTERS

Address	Register Name	Register Write	Register Read	Default	Description
0x1FFFFFFC	AHBREGS_CHIP_ID_NUM	–	(31:24) CHIP_FAMILY	0xB	Chip Family number
		–	(23:16) CHIP_VERSION	0x2	Chip Version number
		–	(15:8) CHIP_MAJOR_REVISION	0x2	Chip Major Revision number
		–	(7:0) CHIP_MINOR_REVISION	0x0	Chip Minor Revision number

A.22 BASEBAND CONTROLLER INTERFACE

Address	Register Name	Register Write	Register Read	Default	Description
0x40001800	BBIF_CTRL	(9:4) CLK_SEL	(9:4) CLK_SEL	0x8	Configure the internal baseband controller clock divider in order to provide a 1MHz reference clock
		(1) WAKEUP_REQ	(1) WAKEUP_REQ	0x0	External wake up request used to sort-out sleep modes
		(0) CLK_ENABLE	(0) CLK_ENABLE	0x0	Enable the baseband controller clocks generation
0x40001804	BBIF_STATUS	-	(16:12) LINK_FORMAT	0x0	BLE link format
		-	(8:4) LINK_LABEL	0x0	BLE link label
		-	(2) CLK_STATUS	0x0	Clock status defining the current active clock in use
		-	(1) OSC_EN	0x1	Oscillator front-end enabling
		-	(0) RADIO_EN	0x1	RF front-end enabling
0x40001808	BBIF_COEX_CTRL	(1) TX	(1) TX	0x0	WLAN collocated device transmit active
		(0) RX	(0) RX	0x0	WLAN collocated device reception active
0x4000180C	BBIF_COEX_STATUS	-	(11:8) BLE_PTI	0x0	BLE packet traffic information
		-	(4) BLE_SYNC	0x0	BLE 625us timing pulse reference
		-	(3) EVENT_IN_ PROCESS	0x0	BLE event status
		-	(2) BLE_IN_PROCESS	0x0	BLE in process indicator
		-	(1) BLE_TX	0x0	BLE transmit indicator
		-	(0) BLE_RX	0x0	BLE reception indicator
0x40001810	BBIF_COEX_INT_CFG	(7:6) EVENT_IN_	(7:6) EVENT_IN_	0x0	EVENT_IN_PROCESS event interrupt

Address	Register Name	Register Write	Register Read	Default	Description
		PROCESS	PROCESS		configuration
		(5:4) BLE_IN_PROCESS	(5:4) BLE_IN_ PROCESS	0x0	BLE_IN_PROCESS event interrupt configuration
		(3:2) BLE_TX	(3:2) BLE_TX	0x0	BLE_TX event interrupt configuration
		(1:0) BLE_RX	(1:0) BLE_RX	0x0	BLE_RX event interrupt configuration
0x40001814	BBIF_COEX_INT_STATUS	–	(11) EVENT_IN_ PROCESS	0x0	EVENT_IN_PROCESS interrupt status flag
		–	(10) BLE_IN_PROCESS	0x0	BLE_IN_PROCESS interrupt status flag
		–	(9) BLE_TX	0x0	BLE_TX interrupt status flag
		–	(8) BLE_RX	0x0	BLE_RX interrupt status flag
		(3) EVENT_IN_PROCESS_ CLEAR	–	N/A	Clear EVENT_IN_PROCESS status flag
		(2) BLE_IN_PROCESS_ CLEAR	–	N/A	Clear BLE_IN_PROCESS status flag
		(1) BLE_TX_CLEAR	–	N/A	Clear BLE_TX status flag
		(0) BLE_RX_CLEAR	–	N/A	Clear BLE_RX status flag
0x400018FC	BBIF_ID_NUM	–	(15:8) BBIF_MAJOR_ REVISION	0x1	Baseband controller interface Major Revision number
		–	(7:0) BBIF_MINOR_ REVISION	0x2	Baseband controller interface Minor Revision number

A.23 BASEBAND CONTROLLER

Address	Register Name	Register Write	Register Read	Default	Description
0x40001900	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) RADIOCNTL_SOFT_RST	(28) RADIOCNTL_SOFT_RST	0x0	Reset the radio controller
		(27) SWINT_REQ	(27) SWINT_REQ	0x0	Force 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
		(20) MD_DSB	(20) MD_DSB	0x0	Allow a single Tx/Rx exchange whatever the MD bits are
		(19) SN_DSB	(19) SN_DSB	0x0	Disable sequence number management
		(18) NESN_DSB	(18) NESN_DSB	0x0	Disable acknowledge scheme
		(17) CRYPT_DSB	(17) CRYPT_DSB	0x0	Disable encryption/decryption
		(16) LRPMAP_DSB	(16) LRPMAP_DSB	0x0	LR pattern mapper/demapper enabled (has effect only if RW_BLE_LONG_RANGE_INST is defined)
		(15) LRFEC_DSB	(15) LRFEC_DSB	0x0	LR FEC encoder/decoder enabled (has effect only if RW_BLE_LONG_RANGE_INST is defined)

Address	Register Name	Register Write	Register Read	Default	Description
		(14) WHIT_DSB	(14) WHIT_DSB	0x0	Disable whitening
		(13) CRC_DSB	(13) CRC_DSB	0x0	Disable CRC stripping
		(12) HOP_REMAP_DSB	(12) HOP_REMAP_DSB	0x0	Disable frequency hopping remapping algorithm
		(11) RXCTEERR_RETX_EN	(11) RXCTEERR_RETX_EN	0x0	Rx CTE error detection
		(10) ANONYMOUS_ ADVERT_FILT_EN	(10) ANONYMOUS_ ADVERT_FILT_EN	0x0	Anonymous extended advertising filtering enable control (operate in extended active scanner and extended passive scanner modes only, and when white list is used by device filtering policy)
		(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
		(3:0) RXWINSZDEF	(3:0) RXWINSZDEF	0x0	Default Rx window size in us (used when device is master connected or performs its second receipt)
0x40001904	BB_VERSION	–	(31:24) TYP	0xA	RW-BLE core type
		–	(23:16) REL	0x0	RW-BLE core version - major release number
		–	(15:8) UPG	0x11	RW-BLE core upgrade - upgrade number
		–	(7:0) BUILD	0x0	RW-BLE core build - build number
0x40001908	BB_RWBLEBCONF	–	(31) DMMODE	0x0	RW-BLE core dual mode
		–	(29) WLANCOEX	0x1	WLAN coexistence mechanism
		–	(28) CORRELATOR	0x0	Correlator present

Address	Register Name	Register Write	Register Read	Default	Description
		–	(27) USERXLR	0x0	Long range Rx present
		–	(26) USETXLR	0x1	Long range Tx present
		–	(24) USEISO	0x0	Support of isochronous channels
		–	(22:16) RFIF	0x8	Support of the RF front-end
		–	(15) USEDBG	0x1	Diagnostic port
		–	(14) DECIPHER	0x0	AES deciphering present
		–	(13:8) CLK_SEL	0x8	Operating frequency (in MHz)
		–	(7) INTMODE	0x0	Interruption mode
		–	(6) BUSTYPE	0x1	Processor bus type
		–	(4:0) ADD_WIDTH	0xE	Value of the RW_BLE_ADDRESS_WIDTH parameter concerted into binary
0x4000190C	BB_INTCNTL0	(16) ERRORINTMSK	(16) ERRORINTMSK	0x0	Error interrupt mask
		(6) ISORXINTMSK	(6) ISORXINTMSK	0x0	Isochronous channel Rx interrupt mask
		(5) ISOTXINTMSK	(5) ISOTXINTMSK	0x0	Isochronous channel Tx interrupt mask
		(4) RXINTMSK	(4) RXINTMSK	0x0	Rx interrupt mask
		(3) TXINTMSK	(3) TXINTMSK	0x0	Tx interrupt mask
		(2) SKIPEVTINTMSK	(2) SKIPEVTINTMSK	0x0	Skipped event interrupt mask
		(1) ENDEVTINTMSK	(1) ENDEVTINTMSK	0x1	End of event interrupt mask
		(0) STARTEVTINTMSK	(0) STARTEVTINTMSK	0x1	Start of event interrupt mask
0x40001910	BB_INTSTAT0	–	(16) ERRORINTSTAT	0x0	Error interrupt status
0x40001914	BB_INTACK0	(16) ERRORINTACK	(16) ERRORINTACK	0x0	Error interrupt acknowledgement
0x40001918	BB_INTCNTL1	(30:28) CLKNINTSRMSK	(30:28) CLKNINTSRMSK	0x0	CLKN/half-slot interrupt sub rating

Address	Register Name	Register Write	Register Read	Default	Description
					mask (valid range is [0:4])
		(27:24) CLKNINTSRVAL	(27:24) CLKNINTSRVAL	0x0	CLKN/half-slot interrupt sub rating value
		(15) FIFOINTMSK	(15) FIFOINTMSK	0x1	FIFO interrupt mask
		(6) TIMESTAMPTGT2INTMSK	(6) TIMESTAMPTGT2INTMSK	0x0	Time stamp target timer 2 interrupt mask
		(5) TIMESTAMPTGT1INTMSK	(5) TIMESTAMPTGT1INTMSK	0x0	Time stamp target timer 1 interrupt mask
		(4) FINETGTIMINTMSK	(4) FINETGTIMINTMSK	0x0	Fine target timer interrupt mask
		(3) SWINTMSK	(3) SWINTMSK	0x0	SW triggered interrupt mask
		(2) CRYPTINTMSK	(2) CRYPTINTMSK	0x0	Encryption engine interrupt mask
		(1) SLPINTMSK	(1) SLPINTMSK	0x1	Sleep mode interrupt mask
		(0) CLKNINTMSK	(0) CLKNINTMSK	0x1	CLKN/half slot interrupt mask
0x4000191C	BB_INTSTAT1	–	(15) FIFOINTSTAT	0x0	FIFO interrupt status
		–	(6) TIMESTAMPTGT2INTSTAT	0x0	Time stamp target timer 2 interrupt status
		–	(5) TIMESTAMPTGT1INTSTAT	0x0	Time stamp target timer 1 interrupt status
		–	(4) FINETGTIMINTSTAT	0x0	Fine target timer interrupt status
		–	(3) SWINTSTAT	0x0	SW triggered interrupt status
		–	(2) CRYPTINTSTAT	0x0	Encryption engine interrupt status
		–	(1) SLPINTSTAT	0x0	Sleep mode interrupt status
		–	(0) CLKNINTSTAT	0x0	CLKN/half slot interrupt status
0x40001920	BB_INTACK1	(15) FIFOINTACK	(15) FIFOINTACK	0x0	FIFO interrupt acknowledgement
		(6)	(6)	0x0	Time stamp target timer 2 interrupt

Address	Register Name	Register Write	Register Read	Default	Description
		TIMESTAMPTGT2INTACK	TIMESTAMPTGT2INTACK		acknowledgement
		(5) TIMESTAMPTGT1INTACK	(5) TIMESTAMPTGT1INTACK	0x0	Time stamp target timer 1 interrupt acknowledgement
		(4) FINETGTIMINTACK	(4) FINETGTIMINTACK	0x0	Fine target timer interrupt acknowledgement
		(3) SWINTACK	(3) SWINTACK	0x0	SW triggered interrupt acknowledgement
		(2) CRYPTINTACK	(2) CRYPTINTACK	0x0	Encryption engine interrupt acknowledgement
		(1) SLPINTACK	(1) SLPINTACK	0x0	Sleep mode interrupt acknowledgement
		(0) CLKNINTACK	(0) CLKNINTACK	0x0	CLKN/half slot interrupt acknowledgement
0x40001924	BB_ACTFIFOSTAT	–	(31:28) SKIP_ET_IDX	0x0	Exchange table entry index of the reported skipped event (valid when SKIPACTINTSTAT is set)
		–	(27:24) CURRENT_ET_IDX	0x0	Exchange table entry index of the reported current event (valid for any set reported interrupt except SKIPACTINTSTAT)
		–	(15) ACTFLAG	0x0	Forced to 1 in BLE
		–	(6) ISORXINTSTAT	0x0	Isochronous channel Rx interrupt status
		–	(5) ISOTXINTSTAT	0x0	Isochronous channel Tx interrupt status
		–	(4) RXINTSTAT	0x0	Rx interrupt status
		–	(3) TXINTSTAT	0x0	Tx interrupt status

Address	Register Name	Register Write	Register Read	Default	Description
		-	(2) SKIPACTINTSTAT	0x0	Skipped event interrupt status
		-	(1) ENDACTINTSTAT	0x0	End of event interrupt status
		-	(0) STARTACTINTSTAT	0x0	Start of event interrupt status
0x40001928	BB_CURRENTRXDESCPTR	(13:0) CURRENTRXDESCPTR	(13:0) CURRENTRXDESCPTR	0x0	Rx descriptor pointer that determines the starting point of the receive buffer chained list
0x4000192C	BB_ETPR	(13:0) ETPTR	(13:0) ETPTR	0x0	Exchange table pointer that determines the starting point of the exchange table
0x40001930	BB_DEEPSLCNTL	(31) EXTWKUPDSB	(31) EXTWKUPDSB	0x0	External wake-up disable
		-	(15) DEEP_SLEEP_STAT	0x0	Indicator of current deep sleep clock mux status
		(3) DEEP_SLEEP_CORR_EN	(3) DEEP_SLEEP_CORR_EN	0x0	Half slot counter integer and fractional part correction (apply 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	Control the radio module
		(0) OSC_SLEEP_EN	(0) OSC_SLEEP_EN	0x0	Control the RF high frequency crystal oscillator
0x40001934	BB_DEEPSLWKUP	(31:0) DEEPSLTIME	(31:0) DEEPSLTIME	0x0	Determine the time in low_power_clk clock cycles to spend in deep sleep mode before waking-up the device
0x40001938	BB_DEEPSLSTAT	-	(31:0) DEEPSLDUR	0x0	Actual duration of the last deep sleep phase measured in low_power_clk clock cycle
0x4000193C	BB_ENBPRESET	(31:21) TWEXT	(31:21) TWEXT	0x0	Time in low power oscillator cycles

Address	Register Name	Register Write	Register Read	Default	Description
					allowed for stabilization of the high frequency oscillator following an external wake-up request (signal wakeup_req)
		(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])
		(9:0) TWRM	(9:0) TWRM	0x0	Time in low power oscillator cycles allowed for the radio module to leave low-power mode
0x40001940	BB_FINECNTCORR	(9:0) FINECNTCORR	(9:0) FINECNTCORR	0x0	Phase correction value for the 312.5us reference counter (i.e. fine counter) in half us
0x40001944	BB_CLKNCNTCORR	(31) ABS_DELTA	(31) ABS_DELTA	0x0	Determine whether CLKNCNTCORR is an absolute correction or a signed "delta" increment correction
		(27:0) CLKNCNTCORR	(27:0) CLKNCNTCORR	0x0	CLKN counter correction value
0x40001950	BB_DIAGCNTL	(31) DIAG3_EN	(31) DIAG3_EN	0x0	Enable diagnostic port 3 output
		(30:24) DIAG3	(30:24) DIAG3	0x0	Selection of the outputs that must be driven to the diagnostic port 3
		(23) DIAG2_EN	(23) DIAG2_EN	0x0	Enable diagnostic port 2 output
		(22:16) DIAG2	(22:16) DIAG2	0x0	Selection of the outputs that must be driven to the diagnostic port 2
		(15) DIAG1_EN	(15) DIAG1_EN	0x0	Enable diagnostic port 1 output
		(14:8) DIAG1	(14:8) DIAG1	0x0	Selection of the outputs that must

Address	Register Name	Register Write	Register Read	Default	Description
					be driven to the diagnostic port 1
		(7) DIAG0_EN	(7) DIAG0_EN	0x0	Enable diagnostic port 0 output
		(6:0) DIAG0	(6:0) DIAG0	0x0	Selection of the outputs that must be driven to the diagnostic port 1
0x40001954	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)
0x40001958	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
0x4000195C	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
0x40001960	BB_ERRORTYPESTAT	-	(22) DFCNTL_EMACC_ERROR	0x0	Indicate that direction finding controller has an EM Access error (happen when exchange memory accesses are not served in time and data are corrupted)
		-	(21) FIFOINTOVF	0x0	Indicate that the FIFO IRQ is overflowed

Address	Register Name	Register Write	Register Read	Default	Description
		–	(20) PHY_ERROR	0x0	Indicate that the programmed CS-AUX/TX/RX-RATE fields are not matching the RADIOCNTL2-PHYMSK fields indicating which PHY the radio is currently supporting
		–	(19) TXAEHEADER_PTR_ERROR	0x0	Indicate Tx pointer to the extended advertising packet that has to be sent is null, while the extended header length is not null and the packet to be sent is an extended advertising packet
		–	(18) TMAFS_ERROR	0x0	Indicate T_MAFS is smaller than 300us in between a transmitted advertising packet containing an AuxPtr field and its chained packet (this comes from bad settings of Aux_Offset and/or Offset_Unit values)
		–	(17) RAL_UNDERRUN	0x0	Indicate resolving address list engine under run issue (happen when RAL List parsing not finished on time)
		–	(16) RAL_ERROR	0x0	Indicate resolving address list engine faced a bad setting
		–	(15) RXDATA_PTR_ERROR	0x0	Indicate whether Rx data buffer pointer value programmed is null (major failure)
		–	(14) TXDATA_PTR_ERROR	0x0	Indicate whether Tx data buffer pointer value programmed is null during advertising/scanning/initiating

Address	Register Name	Register Write	Register Read	Default	Description
					events, or during master/slave connections with non-null packet length (major failure)
		–	(13) RXDESC_EMPTY_ ERROR	0x0	Indicate whether Rx descriptor pointer value programmed in register is null (major failure)
		–	(12) TXDESC_EMPTY_ ERROR	0x0	Indicate whether Tx descriptor pointer value programmed in control structure is null during advertising/scanning/initiating events (major failure)
		–	(11) CSFORMAT_ERROR	0x0	Indicate whether CS-FORMAT has been programmed with an invalid value (major failure)
		–	(10) LLCHMAP_ERROR	0x0	Indicate link layer channel map error (happen when actual number of CS-LLCHMAP bit set to one is different from CS-NBCHGOOD at the beginning of frequency hopping process)
		–	(9) ADV_UNDERRUN	0x0	Indicate advertising interval under run
		–	(8) IFS_UNDERRUN	0x0	Indicate inter frame space under run (occur 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)
		–	(7) LIST_ERROR	0x0	Indicate a software programming issue (white list or periodic

Address	Register Name	Register Write	Register Read	Default	Description
					advertiser list or ADI list search request with empty list, or null base pointer)
		–	(6) EVT_CNTL_APFM_ ERROR	0x0	Indicate anticipated pre-fetch mechanism error (happen when 2 consecutive events are programmed, and when the first event is not completely finished while second pre-fetch instant is reached)
		–	(5) ACT_SCHDL_APFM_ ERROR	0x0	Indicate anticipated pre-fetch mechanism error (happen when 2 consecutive events are programmed, and when the first event is not completely finished while second pre-fetch instant is reached)
		–	(4) ACT_SCHDL_ENTRY_ ERROR	0x0	Indicate activity scheduler faced Invalid timing programing on two consecutive ET entries
		–	(3) RADIO_EMACC_ERROR	0x0	Indicate radio controller exchange memory access error (happen when exchange memory accesses are not served in time and data are corrupted)
		–	(2) PKTCNTL_EMACC_ ERROR	0x0	Indicate packet controller exchange memory access error (happen when exchange memory accesses are not served in time and Tx/Rx data are corrupted)
		–	(1) RXCRYPT_ERROR	0x0	Indicate real time decryption error (happen when AES-CCM

Address	Register Name	Register Write	Register Read	Default	Description
					decryption is too slow compared to packet controller requests)
		–	(0) TXCRYPT_ERROR	0x0	Indicate real time encryption error (happen when AES-CCM encryption is too slow compared to packet controller requests)
0x40001964	BB_SWPROFILING	(31:0) SWPROF	(31:0) SWPROF	0x0	Software profiling register (used by RW-BLE software for profiling purpose)
0x40001970	BB_RADIOCNTL0	(29:16) SPIPTR	(29:16) SPIPTR	0x0	Pointer to the buffer containing data to be transferred to or received from the SPI port
		(7) SPICFG	(7) SPICFG	0x0	SPI configuration/used for SW-driven access and SPI structure interpretation (interpretation is radio dependent)
		(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
0x40001974	BB_RADIOCNTL1	(31) FORCEAGC_EN	(31) FORCEAGC_EN	0x0	Control AGC force mode based onto FORCEAGC_LENGTH value
		(30) FORCEIQ	(30) FORCEIQ	0x0	Control modulation mode in between FM and I and Q
		(29) RXDNSL	(29) RXDNSL	0x0	Do not send length (over SPI) during Rx operation
		(28) TXDNSL	(28) TXDNSL	0x0	Do not send length (over SPI) during Tx operation
		(27:16) FORCEAGC_ LENGTH	(27:16) FORCEAGC_ LENGTH	0x0	Control ATLAS/Ripple AGC force mode based on radioCNTL2-

Address	Register Name	Register Write	Register Read	Default	Description
					FORCEAGC_LENGTH value
		(15) SYNC_PULSE_MODE	(15) SYNC_PULSE_MODE	0x0	Define whether the SYNC_P pulse is generated as pulse or level
		(14) SYNC_PULSE_SRC	(14) SYNC_PULSE_SRC	0x0	Define whether access address synchronization detection is generated internally or comes from the radio
		(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	Select jitter elimination FIFO
		(9:4) XRFSEL	(9:4) XRFSEL	0x0	Extended radio selection field
		(3:0) SUBVERSION	(3:0) SUBVERSION	0x0	CSEM RF sub-version selection
0x40001978	BB_RADIOCNTL2	(31:30) LRSYNCCOMPMODE	(31:30) LRSYNCCOMPMODE	0x3	Long-range synchronization compensation operating mode
		(29) RXCITERMBYPASS	(29) RXCITERMBYPASS	0x0	Long-range CI bit[1] and TERM1 bypass mode (allow to receive only CI bit 0 and then wait for Rx payload directly)
		(28:24) LRVTBFLUSH	(28:24) LRVTBFLUSH	0x8	Indicate long range Viterbi flush instant (this value corresponds to the Viterbi trace back depth used in the selected design, hence corresponding to the number of remaining samples flush out just before the end of the packet)
		(23:22) PHYMSK	(23:22) PHYMSK	0x0	Indicate selected radio PHY support capabilities (in addition to 1Mbps that is mandatory)

Address	Register Name	Register Write	Register Read	Default	Description
		(21:20) LRSYNCERR	(21:20) LRSYNCERR	0x0	Number of errors allowed during long range Rx stream detection (when performed internally)
		(18:16) SYNCERR	(18:16) SYNCERR	0x0	Indicate the maximum number of errors allowed to recognize the synchronization word
		(13:0) FREQTABLE_PTR	(13:0) FREQTABLE_PTR	0x40	Frequency table pointer
0x4000197C	BB_RADIOCNTL3	(31:30) RXRATE3CFG	(31:30) RXRATE3CFG	0x3	Rate out programmable value in Rx when CS-RXRATE is set to 0x3 (i.e 500kbps Long range)
		(29:28) RXRATE2CFG	(29:28) RXRATE2CFG	0x2	Rate out programmable value in Rx when CS-RXRATE is set to 0x2 (i.e 125kbps Long range)
		(27:26) RXRATE1CFG	(27:26) RXRATE1CFG	0x1	Rate out programmable value in Rx when CS-RXRATE is set to 0x1 (i.e 2Mbps)
		(25:24) RXRATE0CFG	(25:24) RXRATE0CFG	0x0	Rate out programmable value in Rx when CS-RXRATE is set to 0x0 (i.e 1Mbps)
		(22:20) GETRSSIDELAY	(22:20) GETRSSIDELAY	0x4	Delay to read RSSI after an RSSI read request
		(18) RXSYNC_ROUTING	(18) RXSYNC_ROUTING	0x0	Access address detection information routing
		(17:16) RXVALID_BEH	(17:16) RXVALID_BEH	0x0	Define radio_in[3] expected behavior
		(15:14) TXRATE3CFG	(15:14) TXRATE3CFG	0x3	Rate out programmable value in Tx when CS-TX/AUX-RATE is set to 0x3 (i.e 500kbps Long range)
		(13:12) TXRATE2CFG	(13:12) TXRATE2CFG	0x2	Rate out programmable value in Tx

Address	Register Name	Register Write	Register Read	Default	Description
					when CS-TX/AUX-RATE is set to 0x2 (i.e 125kbps Long range)
		(11:10) TXRATE1CFG	(11:10) TXRATE1CFG	0x1	Rate out programmable value in Tx when CS-TX/AUX-RATE is set to 0x1 (i.e 2Mbps)
		(9:8) TXRATE0CFG	(9:8) TXRATE0CFG	0x0	Rate out programmable value in Tx when CS-TX/AUX-RATE is set to 0x0 (i.e 1Mbps)
		(1:0) TXVALID_BEH	(1:0) TXVALID_BEH	0x0	Define radio_out [3] expected behavior
0x40001980	BB_RADIOPWRUPDN0	(31:24) SYNC_ POSITION0	(31:24) SYNC_ POSITION0	0x0	Access address detection pulse/level position for uncoded PHY at 1Mbps
		(23:16) RXPWRUP0	(23:16) RXPWRUP0	0x0	Radio Rx power up (in us) for uncoded PHY at 1Mbps
		(14:8) TXPWRDN0	(14:8) TXPWRDN0	0x0	Radio Tx power down (in us) for uncoded PHY at 1Mbps
		(7:0) TXPWRUP0	(7:0) TXPWRUP0	0x0	Radio Tx power up (in us) for uncoded PHY at 1Mbps
0x40001984	BB_RADIOPWRUPDN1	(31:24) SYNC_ POSITION1	(31:24) SYNC_ POSITION1	0x0	Access address detection pulse/level position for uncoded PHY at 2Mbps
		(23:16) RXPWRUP1	(23:16) RXPWRUP1	0x0	Radio Rx power up (in us) for uncoded PHY at 2Mbps
		(14:8) TXPWRDN1	(14:8) TXPWRDN1	0x0	Radio Tx power down (in us) for uncoded PHY at 2Mbps
		(7:0) TXPWRUP1	(7:0) TXPWRUP1	0x0	Radio Tx power up (in us) for uncoded PHY at 2Mbps
0x40001988	BB_RADIOPWRUPDN2	(31:24) SYNC_	(31:24) SYNC_	0x0	Access address detection

Address	Register Name	Register Write	Register Read	Default	Description
		POSITION2	POSITION2		pulse/level position for coded PHY at 125kbps
		(23:16) RXPWRUP2	(23:16) RXPWRUP2	0x0	Radio Rx power up (in us) for coded PHY at 125kbps
		(14:8) TXPWRDN2	(14:8) TXPWRDN2	0x0	Radio Tx power down (in us) for coded PHY at 125kbps
		(7:0) TXPWRUP2	(7:0) TXPWRUP2	0x0	Radio Tx power up (in us) for coded PHY at 125kbps
0x4000198C	BB_RADIOPWRUPDN3	(14:8) TXPWRDN3	(14:8) TXPWRDN3	0x0	Radio Tx power down (in us) for coded PHY at 500kbps
		(7:0) TXPWRUP3	(7:0) TXPWRUP3	0x0	Radio Tx power up for (in us) PHY at 500kbps
0x40001990	BB_RADIOTXRXTIM0	(22:16) RFRXTMDA0	(22:16) RFRXTMDA0	0x0	RF Rx test mode delay adjustment for uncoded PHY at 1Mbps
		(14:8) RXPATHDLY0	(14:8) RXPATHDLY0	0x0	Rx path delay (in us) for uncoded PHY at 1Mbps
		(6:0) TXPATHDLY0	(6:0) TXPATHDLY0	0x0	Rx path delay (in us) for uncoded PHY at 1Mbps
0x40001994	BB_RADIOTXRXTIM1	(22:16) RFRXTMDA1	(22:16) RFRXTMDA1	0x0	RF Rx test mode delay adjustment for uncoded PHY at 2Mbps
		(14:8) RXPATHDLY1	(14:8) RXPATHDLY1	0x0	Rx path delay (in us) for uncoded PHY at 2Mbps
		(6:0) TXPATHDLY1	(6:0) TXPATHDLY1	0x0	Rx path delay (in us) for uncoded PHY at 2Mbps
0x40001998	BB_RADIOTXRXTIM2	(31:24) RXFLUSHPATHDLY2	(31:24) RXFLUSHPATHDLY2	0x0	Rx path delay (in us) for coded PHY at 125kbps
		(23:16) RFRXTMDA2	(23:16) RFRXTMDA2	0x0	RF Rx test mode delay adjustment for uncoded PHY at 125kbps

Address	Register Name	Register Write	Register Read	Default	Description
		(15:8) RXPATHDLY2	(15:8) RXPATHDLY2	0x0	Rx path delay (in us) for uncoded PHY at 125kbps
		(6:0) TXPATHDLY2	(6:0) TXPATHDLY2	0x0	Rx path delay (in us) for uncoded PHY at 125kbps
0x4000199C	BB_RADIOTXRXTIM3	(31:24) RXFLUSHPATHDLY3	(31:24) RXFLUSHPATHDLY3	0x0	Rx path delay (in us) for coded PHY at 500kbps
		(22:16) RFRXTMDA3	(22:16) RFRXTMDA3	0x0	RF Rx test mode delay adjustment for coded PHY at 500kbps
		(6:0) TXPATHDLY3	(6:0) TXPATHDLY3	0x0	Rx path delay (in us) for coded PHY at 500kbps
0x400019A0	BB_SPIPTRCNTL0	(29:16) TXOFFPTR	(29:16) TXOFFPTR	0x0	Pointer to the TxOFF sequence address section
		(13:0) TXONPTR	(13:0) TXONPTR	0x0	Pointer to the TxON sequence address section
0x400019A4	BB_SPIPTRCNTL1	(29:16) RXOFFPTR	(29:16) RXOFFPTR	0x0	Pointer to the RxOFF sequence address section
		(13:0) RXONPTR	(13:0) RXONPTR	0x0	Pointer to the RxON sequence address section
0x400019A8	BB_SPIPTRCNTL2	(29:16) RXLENGTHPTR	(29:16) RXLENGTHPTR	0x0	Pointer to the received length write sequence address section
		(13:0) RSSIPTR	(13:0) RSSIPTR	0x0	Pointer to the RSSI read sequence address section
0x400019AC	BB_SPIPTRCNTL3	(29:16) CTESAMPPTR	(29:16) CTESAMPPTR	0x0	Pointer to the CTE sampling indication sequence address section
		(13:0) RXPKTTYPPTR	(13:0) RXPKTTYPPTR	0x0	Pointer to the received packet type indication sequence address section

Address	Register Name	Register Write	Register Read	Default	Description
0x400019B0	BB_AESCNTL	(1) AES_MODE	(1) AES_MODE	0x0	Cipher mode control
		(0) AES_START	(0) AES_START	0x0	Start AES-128 ciphering/deciphering process
0x400019B4	BB_AESKEY31_0	(31:0) AESKEY31_0	(31:0) AESKEY31_0	0x0	AES encryption 128-bit key (bits 31 down to 0)
0x400019B8	BB_AESKEY63_32	(31:0) AESKEY63_32	(31:0) AESKEY63_32	0x0	AES encryption 128-bit key (bits 63 down to 32)
0x400019BC	BB_AESKEY95_64	(31:0) AESKEY95_64	(31:0) AESKEY95_64	0x0	AES encryption 128-bit key (bits 95 down to 64)
0x400019C0	BB_AESKEY127_96	(31:0) AESKEY127_96	(31:0) AESKEY127_96	0x0	AES encryption 128-bit key (bits 127 down to 96)
0x400019C4	BB_AESPTR	(13:0) AESPTR	(13:0) AESPTR	0x0	Pointer to the memory zone where the block to cipher/decipher using AES-128 is stored
0x400019C8	BB_TXMICVAL	-	(31:0) TXMICVAL	0x0	AES-CCM plain MIC value (valid on when MIC has been calculated in Tx)
0x400019CC	BB_RXMICVAL	-	(31:0) RXMICVAL	0x0	AES-CCM plain MIC value (valid on once MIC has been extracted from Rx packet)
0x400019D0	BB_RFTESTCNTL	(31) INFINITERX	(31) INFINITERX	0x0	Applicable in RF test mode only
		(27) RXPKTCNTEN	(27) RXPKTCNTEN	0x0	Applicable in RF test mode only
		(25:24) PERCOUNT_MODE	(25:24) PERCOUNT_MODE	0x0	Applicable in RF direct Rx test mode only, and when RXPKTCNTEN equals to 1
		(15) INFINITETX	(15) INFINITETX	0x0	Applicable in RF test mode only
		(14) TXLENGTHSRC	(14) TXLENGTHSRC	0x0	Applicable only in Tx/Rx RF test mode

Address	Register Name	Register Write	Register Read	Default	Description
		(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	0x0	Applicable in RF test mode only
		(7:0) TXLENGTH	(7:0) TXLENGTH	0x0	Tx packet length in number of byte
0x400019D4	BB_RFTESTTXSTAT	-	(31:0) TXPKTCNT	0x0	Report number of transmitted packet during test modes
0x400019D8	BB_RFTESTRXSTAT	-	(31:0) RXPKTCNT	0x0	Report number of correctly received packet during test modes
0x400019E0	BB_TIMGENCNTL	(25:16) PREFETCHABORT_TIME	(25:16) PREFETCHABORT_TIME	0x1FE	Define the instant in us at which immediate abort is required after anticipated pre-fetch abort
		(8:0) PREFETCH_TIME	(8:0) PREFETCH_TIME	0x96	Define exchange table pre-fetch instant in us
0x400019E4	BB_FINETIMTGT	(27:0) FINETARGET	(27:0) FINETARGET	0x0	Fine timer target value on which a ble_finetgtim_irq must be generated (this timer has a precision of 312.5us and interrupt is generated only when FINETARGET = CLKN)
0x400019E8	BB_CLKNTGT1	(27:0) CLKNTGT1	(27:0) CLKNTGT1	0x0	This timer has a precision of 312.5us (interrupt is generated only when CLKNTGT = CLKN and HMICROSECTGT = FINECNT)
0x400019EC	BB_HMICROSECTGT1	(9:0) HMICROSECTGT1	(9:0) HMICROSECTGT1	0x0	This timer has a precision of 0.5us (interrupt is generated only when CLKNTGT = CLKN and HMICROSECTGT = FINECNT)

Address	Register Name	Register Write	Register Read	Default	Description
0x400019F0	BB_CLKNTGT2	(27:0) CLKNTGT2	(27:0) CLKNTGT2	0x0	This timer has a precision of 312.5us (interrupt is generated only when CLKNTGT = CLKN and HMICROSECTGT = FINECNT)
0x400019F4	BB_HMICROSECTGT2	(9:0) HMICROSECTGT2	(9:0) HMICROSECTGT2	0x0	This timer has a precision of 0.5us (interrupt is generated only when CLKNTGT = CLKN and HMICROSECTGT = FINECNT)
0x400019F8	BB_SLOTCLK	(31) SAMP	(31) SAMP	0x0	Sample the CLKN counter value in SCLK register field
		(30) CLKN_UPD	(30) CLKN_UPD	0x0	Update CLKN counter
		(27:0) SCLK	(27:0) SCLK	0x0	Value of the 312.5us CLKN counter
0x400019FC	BB_FINETIMECNT	–	(9:0) FINECNT	0x0	This timer has a precision of 0.5us (interrupt is generated only when CLKNTGT = CLKN and HMICROSECTGT = FINECNT)
0x40001A00	BB_ACTSCHCNTL	(31) START_ACT	(31) START_ACT	0x0	Request the RW-BLE core to start an event
		(3:0) ENTRY_IDX	(3:0) ENTRY_IDX	0x0	Indicate the activity scheduler entry index which has to be used when START_ACT is set
0x40001A04	BB_STARTEVTCLKNTS	–	(27:0) STARTEVTCLKNTS	0x0	Value of the CLKN counter when ble_start_int is generated
0x40001A08	BB_STARTEVTFINECNTTS	–	(9:0) STARTEVTFINECNTTS	0x0	Value of the fine counter when ble_start_int is generated
0x40001A0C	BB_ENDEVTCLKNTS	–	(27:0) ENDEVTCLKNTS	0x0	Value of the CLKN counter when ble_end_int is generated
0x40001A10	BB_ENDEVTFINECNTTS	–	(9:0) ENDEVTFINECNTTS	0x0	Value of the fine counter when ble_end_int is generated

Address	Register Name	Register Write	Register Read	Default	Description
0x40001A14	BB_SKIPEVTCLKNTS	-	(27:0) SKIPEVTCLKNTS	0x0	Value of the CLKN counter when ble_skip_int is generated
0x40001A18	BB_SKIPEVTFINECNTTS	-	(9:0) SKIPEVTFINECNTTS	0x0	Value of the fine counter when ble_skip_int is generated
0x40001A20	BB_ADVTIM	(31:24) TX_AUXPTR_THR	(31:24) TX_AUXPTR_THR	0x0	Extended advertising AuxPtr threshold value in Tx (granularity 16us)
		(23:16) RX_AUXPTR_THR	(23:16) RX_AUXPTR_THR	0x0	Extended advertising AuxPtr threshold value in Rx (granularity 16us)
		(13:0) ADVINT	(13:0) ADVINT	0x0	Advertising packet interval defining the time interval in between two ADV_xxx packet sent (value in us)
0x40001A24	BB_ACTSCANCNTL	(24:16) BACKOFF	(24:16) BACKOFF	0x1	Active scan mode back-off counter initialization value
		(8:0) UPPERLIMIT	(8:0) UPPERLIMIT	0x1	Active scan mode upper limit counter value
0x40001A30	BB_WPALCNTL	(23:16) WPALNBDEV	(23:16) WPALNBDEV	0x0	Number of devices in the white list
		(13:0) WPALBASEPTR	(13:0) WPALBASEPTR	0x0	Base address pointer of the white list
0x40001A34	BB_WPALCURRENPTR	(13:0) WPALCURRENPTR	(13:0) WPALCURRENPTR	0x0	Current pointer in use for the white list
0x40001A38	BB_SEARCH_TIMEOUT	(5:0) SEARCH_TIMEOUT	(5:0) SEARCH_TIMEOUT	0x10	RAL and list search engines timeout delay in us
0x40001A40	BB_COEXIFCNTL0	(21:20) MWSSCANFREQMSK	(21:20) MWSSCANFREQMSK	0x0	Determine how mws_scan_frequency impacts BLE Tx and Rx
		(19:18) WLCRXPRIOMODE	(19:18) WLCRXPRIOMODE	0x0	Define BLE packet ble_rx mode behavior

Address	Register Name	Register Write	Register Read	Default	Description
		(17:16) WLCTXPRIOMODE	(17:16) WLCTXPRIOMODE	0x0	Define BLE packet ble_tx mode behavior
		(15:14) MWSTXFREQMSK	(15:14) MWSTXFREQMSK	0x0	Determine how MWS Tx Frequency impacts BLE Tx and Rx
		(13:12) MWSRXFREQMSK	(13:12) MWSRXFREQMSK	0x0	Determine how MWS Rx frequency impacts BLE Tx and Rx
		(11:10) MWSTXMSK	(11:10) MWSTXMSK	0x0	Determine how mws_tx impacts BLE Tx and Rx
		(9:8) MWSRXMSK	(9:8) MWSRXMSK	0x0	Determine how mws_rx impacts BLE Tx and Rx
		(7:6) WLANTXMSK	(7:6) WLANTXMSK	0x0	Determine how wlan_tx impacts BLE Tx and Rx
		(5:4) WLANRXMSK	(5:4) WLANRXMSK	0x1	Determine how wlan_rx impacts 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	Determine whether ble_sync is generated or not
		(0) WLANCOEX_EN	(0) WLANCOEX_EN	0x0	Enable/disable control of the MWS/WLAN coexistence control
0x40001A44	BB_COEXIFCNTL1	(28:24) WLCPRXTHR	(28:24) WLCPRXTHR	0x0	Determine the threshold for Rx priority setting (apply on ble_rx if WLCRXPRIOMODE equals "10")
		(20:16) WLCPTXTHR	(20:16) WLCPTXTHR	0x0	Determine the threshold for priority setting (apply on ble_tx if

Address	Register Name	Register Write	Register Read	Default	Description
					WLCTXPRIOMODE equals "10")
		(14:8) WLCPDURATION	(14:8) WLCPDURATION	0x0	Determine how many us the priority information must be maintained (apply on ble_tx and ble_rx if WLCTXPRIOMODE and WLCRXPRIOMODE equal "10")
		(6:0) WLCPDELAY	(6:0) WLCPDELAY	0x0	Determine the delay (in us) in Tx/Rx enables rises the time BLE Tx/Rx priority has to be provided (apply on ble_tx and ble_rx if WLCTXPRIOMODE and WLCRXPRIOMODE equal "10")
0x40001A48	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
0x40001A4C	BB_BLEMPRIO0	(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

Address	Register Name	Register Write	Register Read	Default	Description
		(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
0x40001A50	BB_BLEMPRIO1	(31:28) BLEM15	(31:28) BLEM15	0x3	Set priority value for passive extended passive scanning on secondary advertising channels
		(27:24) BLEM14	(27:24) BLEM14	0x4	Set priority value for non-connectable extended advertising on secondary advertising channels
		(23:20) BLEM13	(23:20) BLEM13	0x8	Set priority value for connectable extended advertising on secondary advertising channels
		(19:16) BLEM12	(19:16) BLEM12	0x9	Set priority value for extended active scanning on secondary advertising channels
		(15:12) BLEM11	(15:12) BLEM11	0xA	Set priority value for extended initiating on secondary advertising channels
		(11:8) BLEM10	(11:8) BLEM10	0xF	Set priority value for connection establishment BLE message on secondary advertising channels
		(7:4) BLEM9	(7:4) BLEM9	0xD	Set default priority value for ISO channel subsequent Tx/Rx attempt
		(3:0) BLEM8	(3:0) BLEM8	0xC	Set default priority value for ISO channel first Tx/Rx attempt
0x40001A54	BB_BLEMPRIO2	(31:28) BLEMDEFAULT	(31:28) BLEMDEFAULT	0x3	Set priority value for extended initiating on secondary advertising channels

Address	Register Name	Register Write	Register Read	Default	Description
		(11:8) BLEM18	(11:8) BLEM18	0x2	Set priority value for connection establishment BLE message on secondary advertising channels
		(7:4) BLEM17	(7:4) BLEM17	0x7	Set default priority value for ISO channel subsequent Tx/Rx attempt
		(3:0) BLEM16	(3:0) BLEM16	0x7	Set default priority value for ISO channel first Tx/Rx attempt
0x40001A60	BB_RALCNTL	(23:16) RALNBDEV	(23:16) RALNBDEV	0x0	Number of devices in RAL structure
		(13:0) RALBASEPTR	(13:0) RALBASEPTR	0x0	Start address pointer of the RAL structure
0x40001A64	BB_RALCURRENTPTR	(13:0) RALCURRENTPTR	(13:0) RALCURRENTPTR	0x0	Current pointer of the RAL structure
0x40001A68	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 (report the current local RPA random number LFSR value otherwise)
0x40001A6C	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 (report the current local RPA random number LFSR value otherwise)
0x40001A70	BB_DFCNTL0_1US	(31:24) RXSAMPSTINST0_1US	(31:24) RXSAMPSTINST0_1US	0x0	Adjustement delay in half us of Rx I and Q sampling start instant for LE 1M PHY (with 1us sampling interval)

Address	Register Name	Register Write	Register Read	Default	Description
		(23:16) RXSWSTINST0_1US	(23:16) RXSWSTINST0_1US	0x0	Adjustement delay in half us of Rx switch start instant for LE 1M PHY (with 1us switching interval)
		(7:0) TXSWSTINST0_1US	(7:0) TXSWSTINST0_1US	0x0	Adjustement delay in half us of Tx switch start instant for LE 1M PHY (with 1us switching interval)
0x40001A74	BB_DFCNTL0_2US	(31:24) RXSAMPSTINST0_2US	(31:24) RXSAMPSTINST0_2US	0x0	Adjustement delay in half us of Rx I and Q sampling start instant for LE 1M PHY (with 2us sampling interval)
		(23:16) RXSWSTINST0_2US	(23:16) RXSWSTINST0_2US	0x0	Adjustement delay in half us of Rx switch start instant for LE 1M PHY (with 2us switching interval)
		(7:0) TXSWSTINST0_2US	(7:0) TXSWSTINST0_2US	0x0	Adjustement delay in half us of Tx switch start instant for LE 1M PHY (with 2us switching interval)
0x40001A78	BB_DFCNTL1_1US	(31:24) RXSAMPSTINST1_1US	(31:24) RXSAMPSTINST1_1US	0x0	Adjustement delay in half us of Rx I and Q sampling start instant for LE 2M PHY (with 1us sampling interval)
		(23:16) RXSWSTINST1_1US	(23:16) RXSWSTINST1_1US	0x0	Adjustement delay in half us of Rx switch start instant for LE 2M PHY (with 1us switching interval)
		(7:0) TXSWSTINST1_1US	(7:0) TXSWSTINST1_1US	0x0	Adjustement delay in half us of Tx switch start instant for LE 2M PHY (with 1us switching interval)
0x40001A7C	BB_DFCNTL1_2US	(31:24) RXSAMPSTINST1_2US	(31:24) RXSAMPSTINST1_2US	0x0	Adjustement delay in half us of Rx I and Q sampling start instant for LE 2M PHY (with 2us sampling interval)

Address	Register Name	Register Write	Register Read	Default	Description
		(23:16) RXSWSTINST1_2US	(23:16) RXSWSTINST1_2US	0x0	Adjustement delay in half us of Rx switch start instant for LE 2M PHY (with 2us switching interval)
		(7:0) TXSWSTINST1_2US	(7:0) TXSWSTINST1_2US	0x0	Adjustement delay in half us of Tx switch start instant for LE 2M PHY (with 2us switching interval)
0x40001A80	BB_DFCURRENTPTR	(13:0) DFCURRENTPTR	(13:0) DFCURRENTPTR	0x0	Rx CTE descriptor current pointer
0x40001A84	BB_DFANTCNTL	(15) RXPRIMIDCNTLEN	(15) RXPRIMIDCNTLEN	0x0	Reception primary antenna ID enable control
		(14:8) RXPRIMANTID	(14:8) RXPRIMANTID	0x0	Primary antenna ID to be used on each Rx start instant
		(7) TXPRIMIDCNTLEN	(7) TXPRIMIDCNTLEN	0x0	Transmit primary antenna ID enable control
		(6:0) TXPRIMANTID	(6:0) TXPRIMANTID	0x0	Primary antenna ID to be used on each Tx start instant
0x40001A88	BB_DFIFCNTL	(7) ANTSWITCH_BEH	(7) ANTSWITCH_BEH	0x0	Define the antenna switch qualifier behavior
		(6) SAMPREQ_BEH	(6) SAMPREQ_BEH	0x0	Define the I and Q sampling request behavior
		(5:4) SAMPVALID_BEH	(5:4) SAMPVALID_BEH	0x3	Define the I and Q sample qualifier behavior
		(3:2) IF_WIDTH	(3:2) IF_WIDTH	0x3	Define the I and Q sample interface width
		(1) MSB_LSB_ORDER	(1) MSB_LSB_ORDER	0x0	Define whether symbol is sent MSB or LSB first
		(0) SYMBOL_ORDER	(0) SYMBOL_ORDER	0x0	Define whether I sample or Q sample is sent first

A.24 RF FRONT-END 2.4 GHZ

Address	Register Name	Register Write	Register Read	Default	Description
0x40040800	RF0_REG00	(31) DATAWHITE_BTLE_DW_BTLE	(31) DATAWHITE_BTLE_DW_BTLE	0x1	Data whitening control
		(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	Enable 4FSK coding
		(22:20) FOURFSK_CODING_TX_FOURFSK_CODING	(22:20) FOURFSK_CODING_TX_FOURFSK_CODING	0x0	Set the 4FSK coding (Tx mode)
		(18:16) FOURFSK_CODING_RX_FOURFSK_CODING	(18:16) FOURFSK_CODING_RX_FOURFSK_CODING	0x0	Set the 4FSK decoding (Rx mode)
		(14) MODE2_DIFF_CODING	(14) MODE2_DIFF_CODING	0x0	Differential coding/decoding
		(13) MODE2_PSK_NFSK	(13) MODE2_PSK_NFSK	0x0	FSK/PSK mode selection
		(12:8) MODE2_TESTMODE	(12:8) MODE2_TESTMODE	0x0	Output test mode
		(7) MODE_NOT_TO_IDLE	(7) MODE_NOT_TO_IDLE	0x0	FSM goes in suspend mode after a Tx or Rx packet
		(5) MODE_EN_FSM	(5) MODE_EN_FSM	0x1	Radio FSM control
		(4) MODE_EN_DESERIALIZER	(4) MODE_EN_DESERIALIZER	0x0	Deserializer control
		(3) MODE_EN_SERIALIZER	(3) MODE_EN_SERIALIZER	0x0	Serializer control

Address	Register Name	Register Write	Register Read	Default	Description
		(2) MODE_TX_NRX	(2) MODE_TX_NRX	0x0	Select Tx or Rx mode
		(1:0) MODE_MODE	(1:0) MODE_MODE	0x2	Select the working mode of the digital baseband
0x40040804	RF0_REG01	(31:24) TAU_PHASE_ RECOV_TAU_PHASE_ RECOV	(31:24) TAU_PHASE_ RECOV_TAU_PHASE_ RECOV	0x14	Time constant of the fine carrier recovery block (banked)
		(23:16) TAU_ROUGH_ RECOV_TAU_ROUGH_ RECOV	(23:16) TAU_ROUGH_ RECOV_TAU_ROUGH_ RECOV	0xB	Time constant of the rough carrier recovery block (banked)
		(15) CARRIER_ RECOVERY_EN_ CORRECT_CFREQ_AFC	(15) CARRIER_ RECOVERY_EN_ CORRECT_CFREQ_AFC	0x0	Automatic AFC correction (banked)
		(14) CARRIER_ RECOVERY_CORRECT_ CFREQ_IF_NEG	(14) CARRIER_ RECOVERY_CORRECT_ CFREQ_IF_NEG	0x0	IF correction (banked)
		(13) CARRIER_ RECOVERY_EN_ CORRECT_CFREQ_IF	(13) CARRIER_ RECOVERY_EN_ CORRECT_CFREQ_IF	0x1	Automatic IF correction (banked)
		(12) CARRIER_ RECOVERY_AFC_NEG	(12) CARRIER_ RECOVERY_AFC_NEG	0x0	AFC correction (banked)
		(11) CARRIER_ RECOVERY_STARTER_ MODE	(11) CARRIER_ RECOVERY_STARTER_ MODE	0x0	Starter mode (banked)
		(10) CARRIER_ RECOVERY_EN_AFC	(10) CARRIER_ RECOVERY_EN_AFC	0x0	Automatic frequency control (banked)
		(9) CARRIER_ RECOVERY_EN_FINE_ RECOV	(9) CARRIER_ RECOVERY_EN_FINE_ RECOV	0x1	Fine carrier recovery (banked)
		(8) CARRIER_	(8) CARRIER_	0x0	Rough carrier recovery (banked)

Address	Register Name	Register Write	Register Read	Default	Description
		RECOVERY_EN_ROUGH_RECOV	RECOVERY_EN_ROUGH_RECOV		
		(6) MOD_TX_PULSE_NSYM	(6) MOD_TX_PULSE_NSYM	0x0	Tx pulse shape function
		(5) MOD_TX_EN_INTERP	(5) MOD_TX_EN_INTERP	0x0	Tx CIC interpolator
		(4:0) MOD_TX_CK_TX_M	(4:0) MOD_TX_CK_TX_M	0x0	Unsigned value determining the Tx CIC interpolator frequency
0x40040808	RF0_REG02	(25:24) DATARATE_OFFSET_DR_LIMIT	(25:24) DATARATE_OFFSET_DR_LIMIT	0x0	Set the data-rate recovery limits
		(23:16) DATARATE_OFFSET_DATARATE_OFFSET	(23:16) DATARATE_OFFSET_DATARATE_OFFSET	0x0	Data-rate offset
		(15:8) TAU_DATARATE_RECOV_TAU_DATARATE_RECOV	(15:8) TAU_DATARATE_RECOV_TAU_DATARATE_RECOV	0x20	Time constant of the data-rate recovery
		(7:0) TAU_CLK_RECOV_TAU_CLK_RECOV	(7:0) TAU_CLK_RECOV_TAU_CLK_RECOV	0x9	Time constant of the clock recovery (banked)
0x4004080C	RF0_REG03	(31:30) MAC_CONF_MAC_TIMER_GR	(31:30) MAC_CONF_MAC_TIMER_GR	0x2	MAC timer granularity
		(29) MAC_CONF_RX_MAC_ACT	(29) MAC_CONF_RX_MAC_ACT	0x0	Switch FSM to Rx or Tx mode after an Rx mode
		(28) MAC_CONF_RX_MAC_TX_NRX	(28) MAC_CONF_RX_MAC_TX_NRX	0x0	Switch FSM to Tx mode after an Rx mode (Rx otherwise)
		(27) MAC_CONF_RX_MAC_START_NSTOP	(27) MAC_CONF_RX_MAC_START_NSTOP	0x0	MAC timer activation after sync word detection
		(26) MAC_CONF_TX_MAC_ACT	(26) MAC_CONF_TX_MAC_ACT	0x0	Switch FSM to Rx or Tx mode after a Tx mode

Address	Register Name	Register Write	Register Read	Default	Description
		(25) MAC_CONF_TX_MAC_TX_NRX	(25) MAC_CONF_TX_MAC_TX_NRX	0x0	Switch FSM to Tx mode after a Tx mode (Rx otherwise)
		(24) MAC_CONF_TX_MAC_START_NSTOP	(24) MAC_CONF_TX_MAC_START_NSTOP	0x0	MAC timer activation after packet transmission
		(23) IRQ_CONF_IRQ_HIGH_Z	(23) IRQ_CONF_IRQ_HIGH_Z	0x0	Pads are set to high-Z when the IRQ is not active
		(22) IRQ_CONF_IRQ_ACTIVE_LOW	(22) IRQ_CONF_IRQ_ACTIVE_LOW	0x1	IRQ are active low
		(21:16) IRQ_CONF_IRQS_MASK	(21:16) IRQ_CONF_IRQS_MASK	0x0	Mask to determine which IRQs are enabled (active high)
		(15:13) FIFO_2_FIFO_THR_TX	(15:13) FIFO_2_FIFO_THR_TX	0x0	Threshold indicating the "almost empty" Tx FIFO state
		(12) FIFO_2_WAIT_TXFIFO_WR	(12) FIFO_2_WAIT_TXFIFO_WR	0x0	FSM will wait a Tx FIFO write before starting the Tx mode in case of an empty Tx FIFO
		(11) FIFO_2_STOP_ON_RXFF_OVFLW	(11) FIFO_2_STOP_ON_RXFF_OVFLW	0x0	Stop the reception in case of a FIFO overflow
		(10) FIFO_2_STOP_ON_TXFF_UNFLW	(10) FIFO_2_STOP_ON_TXFF_UNFLW	0x0	Stop the transmission in case of a FIFO underflow
		(9) FIFO_2_RXFF_FLUSH_ON_START	(9) FIFO_2_RXFF_FLUSH_ON_START	0x1	Flush the Rx FIFO when the Rx mode is enabled in order to receive a packet with an empty FIFO
		(8) FIFO_2_TXFF_FLUSH_ON_STOP	(8) FIFO_2_TXFF_FLUSH_ON_STOP	0x1	Flush the Tx FIFO after the end of a packet transmission in order to have an empty FIFO
		(7) FIFO_FIFO_FLUSH_ON_OVFLW	(7) FIFO_FIFO_FLUSH_ON_OVFLW	0x0	Overflow FIFO flush control
		(6) FIFO_FIFO_FLUSH_ON_ADDR_ERR	(6) FIFO_FIFO_FLUSH_ON_ADDR_ERR	0x0	Address error FIFO flush control

Address	Register Name	Register Write	Register Read	Default	Description
		(5) FIFO_FIFO_FLUSH_ON_PL_ERR	(5) FIFO_FIFO_FLUSH_ON_PL_ERR	0x0	Packet length error FIFO flush control
		(4) FIFO_FIFO_FLUSH_ON_CRC_ERR	(4) FIFO_FIFO_FLUSH_ON_CRC_ERR	0x1	CRC error FIFO flush control
		(3) FIFO_RX_FIFO_ACK	(3) FIFO_RX_FIFO_ACK	0x0	Rx FIFO acknowledgement
		(2:0) FIFO_FIFO_THR	(2:0) FIFO_FIFO_THR	0x0	Threshold indicating the "almost full" Rx FIFO state
0x40040810	RF0_PADS_03	(28:24) PAD_CONF_1_PAD_3_CONF	(28:24) PAD_CONF_1_PAD_3_CONF	0x0	Configuration of GPIO pad 3
		(20:16) PAD_CONF_1_PAD_2_CONF	(20:16) PAD_CONF_1_PAD_2_CONF	0x0	Configuration of GPIO pad 2
		(12:8) PAD_CONF_1_PAD_1_CONF	(12:8) PAD_CONF_1_PAD_1_CONF	0x0	Configuration of GPIO pad 1
		(4:0) PAD_CONF_1_PAD_0_CONF	(4:0) PAD_CONF_1_PAD_0_CONF	0x0	Configuration of GPIO pad 0
0x40040814	RF0_PADS_47	(28:24) PAD_CONF_2_PAD_7_CONF	(28:24) PAD_CONF_2_PAD_7_CONF	0x0	Configuration of GPIO pad 7
		(20:16) PAD_CONF_2_PAD_6_CONF	(20:16) PAD_CONF_2_PAD_6_CONF	0x0	Configuration of GPIO pad 6
		(12:8) PAD_CONF_2_PAD_5_CONF	(12:8) PAD_CONF_2_PAD_5_CONF	0x0	Configuration of GPIO pad 5
		(4:0) PAD_CONF_2_PAD_4_CONF	(4:0) PAD_CONF_2_PAD_4_CONF	0x0	Configuration of GPIO pad 4
0x40040818	RF0_CENTER_FREQ	(31) CENTER_FREQ_ADAPT_CFREQ	(31) CENTER_FREQ_ADAPT_CFREQ	0x1	Frequency adaptation between Tx and Rx modes
		(30) CENTER_FREQ_RX_DIV_5_N6	(30) CENTER_FREQ_RX_DIV_5_N6	0x0	Ratio of the PLL reference between Tx and Rx modes

Address	Register Name	Register Write	Register Read	Default	Description
		(29:0) CENTER_FREQ_CENTER_FREQUENCY	(29:0) CENTER_FREQ_CENTER_FREQUENCY	0x215C71B	Set the center frequency
0x4004081C	RF0_PADS_89	(31:24) TX_MAC_TIMER_TX_MAC_TIMER	(31:24) TX_MAC_TIMER_TX_MAC_TIMER	0x82	Time to wait after the Tx mode
		(23:16) RX_MAC_TIMER_RX_MAC_TIMER	(23:16) RX_MAC_TIMER_RX_MAC_TIMER	0x23	Time to wait after the Rx mode
		(12:8) PAD_CONF_3_PAD_9_CONF	(12:8) PAD_CONF_3_PAD_9_CONF	0x0	Configuration of GPIO pad 9
		(4:0) PAD_CONF_3_PAD_8_CONF	(4:0) PAD_CONF_3_PAD_8_CONF	0x0	Configuration of GPIO pad 8
0x40040820	RF0_REG08	(31:30) MOD_INFO_RX_DIV_CK_RX	(31:30) MOD_INFO_RX_DIV_CK_RX	0x0	Set the clock divider for the Rx mode (banked)
		(29) MOD_INFO_RX_SYMBOL_2BIT_RX	(29) MOD_INFO_RX_SYMBOL_2BIT_RX	0x0	Rx symbol bits composition (banked)
		(28:24) MOD_INFO_RX_DR_M_RX	(28:24) MOD_INFO_RX_DR_M_RX	0x0	Unsigned value determining the oversampling frequency and consequently the data-rate (banked)
		(23:22) MOD_INFO_TX_DIV_CK_TX	(23:22) MOD_INFO_TX_DIV_CK_TX	0x0	Set the clock divider for the Tx mode (banked)
		(21) MOD_INFO_TX_SYMBOL_2BIT_TX	(21) MOD_INFO_TX_SYMBOL_2BIT_TX	0x0	Tx symbol bits composition (banked)
		(20:16) MOD_INFO_TX_DR_M_TX	(20:16) MOD_INFO_TX_DR_M_TX	0x0	Unsigned value determining the oversampling frequency and consequently the data-rate (banked)
		(14) CHANNEL_SWITCH_IQ	(14) CHANNEL_SWITCH_IQ	0x0	Switch I and Q channels
		(13:8) CHANNEL_CHANNEL	(13:8) CHANNEL_CHANNEL	0x0	Channel number

Address	Register Name	Register Write	Register Read	Default	Description
		(3) BANK_DATARATE_TX_NRX	(3) BANK_DATARATE_TX_NRX	0x0	Select the data-rate register
		(2) BANK_STD_BLE_RATES	(2) BANK_STD_BLE_RATES	0x0	Select the actual bank behavior
		(1:0) BANK_BANK	(1:0) BANK_BANK	0x0	Select the used bank
0x40040824	RF0_CODING	(31) CODING_EN_DATAWHITE	(31) CODING_EN_DATAWHITE	0x1	Data-whitening enabling (banked)
		(30) CODING_I_NQ_DELAYED	(30) CODING_I_NQ_DELAYED	0x0	Channel I delay (banked)
		(29) CODING_OFFSET	(29) CODING_OFFSET	0x0	Offset (delay) introduction (banked)
		(28) CODING_BIT_INVERT	(28) CODING_BIT_INVERT	0x0	Bit value inversion in Tx and Rx modes (banked)
		(27) CODING_EVEN_BEFORE_ODD	(27) CODING_EVEN_BEFORE_ODD	0x0	Determine the bit order in case of a 2 bits per symbol modulation (banked)
		(26) CODING_EN_802154_L2F	(26) CODING_EN_802154_L2F	0x0	Linear to frequency encoding needed in order to modulate an OQPSK as an MSK (banked)
		(25) CODING_EN_802154_B2C	(25) CODING_EN_802154_B2C	0x0	Bit to chips encoding used in the IEEE 802.15.4 standard (banked)
		(24) CODING_EN_MANCHESTER	(24) CODING_EN_MANCHESTER	0x0	Manchester encoding (banked)
		(23) CHANNELS_2_EN_CHANNEL_SEL	(23) CHANNELS_2_EN_CHANNEL_SEL	0x1	Definition of channels (banked)
		(22) CHANNELS_2_EN_CHN_BLE	(22) CHANNELS_2_EN_CHN_BLE	0x1	BLE channels index LUT (banked)
		(19:16) CHANNELS_2_CHANNEL_SPACING_HI	(19:16) CHANNELS_2_CHANNEL_SPACING_HI	0x7	Channel spacing MSB (banked)

Address	Register Name	Register Write	Register Read	Default	Description
		(15:0) CHANNELS_1_CHANNEL_SPACING_LO	(15:0) CHANNELS_1_CHANNEL_SPACING_LO	0x1C72	Channel spacing LSB (banked)
0x40040828	RF0_PACKET_HANDLING	(31:24) PREAMBLE_PREAMBLE	(31:24) PREAMBLE_PREAMBLE	0x55	Preamble to be inserted (banked)
		(22) PACKET_LENGTH_OPTS_EN_PACKET_LEN_FIX	(22) PACKET_LENGTH_OPTS_EN_PACKET_LEN_FIX	0x0	Packet length configuration (banked)
		(21:18) PACKET_LENGTH_OPTS_PACKET_LEN_CORR	(21:18) PACKET_LENGTH_OPTS_PACKET_LEN_CORR	0x0	Signed value specifying the correction to apply to the specified packet length (banked)
		(17:16) PACKET_LENGTH_OPTS_PACKET_LEN_POS	(17:16) PACKET_LENGTH_OPTS_PACKET_LEN_POS	0x1	Unsigned value that specifies the position of the packet length after the pattern (banked)
		(15:8) PACKET_LENGTH_PACKET_LEN	(15:8) PACKET_LENGTH_PACKET_LEN	0xFF	The packet length in the fixed packet length mode (banked)
		(7) PACKET_HANDLING_LSB_FIRST	(7) PACKET_HANDLING_LSB_FIRST	0x1	Select LSB or MSB to send first (banked)
		(6) PACKET_HANDLING_EN_CRC	(6) PACKET_HANDLING_EN_CRC	0x1	Automatic CRC evaluation and insertion (banked)
		(5) PACKET_HANDLING_EN_CRC_ON_PKTLEN	(5) PACKET_HANDLING_EN_CRC_ON_PKTLEN	0x1	CRC calculation on the packet length part of the packet (banked)
		(4) PACKET_HANDLING_EN_PREAMBLE	(4) PACKET_HANDLING_EN_PREAMBLE	0x1	Automatic preamble insertion (banked)
		(3) PACKET_HANDLING_EN_MULTI_FRAME	(3) PACKET_HANDLING_EN_MULTI_FRAME	0x0	Multi-frame packet (banked)
		(2) PACKET_	(2) PACKET_	0x0	Data-whitening on the CRC disabling

Address	Register Name	Register Write	Register Read	Default	Description
		HANDLING_ENB_DW_ON_CRC	HANDLING_ENB_DW_ON_CRC		(banked)
		(1) PACKET_HANDLING_EN_PATTERN	(1) PACKET_HANDLING_EN_PATTERN	0x1	Automatic pattern insertion and recognition (banked)
		(0) PACKET_HANDLING_EN_PACKET	(0) PACKET_HANDLING_EN_PACKET	0x1	Packet handler enabling (banked)
0x4004082C	RF0_SYNC_PATTERN	(31:0) PATTERN	(31:0) PATTERN	0x8E89BED6	Pattern (sync word) to be inserted or recognized (banked)
0x40040830	RF0_REG0C	(31:16) ADDRESS_ADDRESS	(31:16) ADDRESS_ADDRESS	0x0	Address of the node (banked)
		(11) ADDRESS_CONF_ADDRESS_LEN	(11) ADDRESS_CONF_ADDRESS_LEN	0x0	Address length selection (banked)
		(10) ADDRESS_CONF_EN_ADDRESS_RX_BR	(10) ADDRESS_CONF_EN_ADDRESS_RX_BR	0x0	Broadcast address detection in Rx mode (banked)
		(9) ADDRESS_CONF_EN_ADDRESS_RX	(9) ADDRESS_CONF_EN_ADDRESS_RX	0x0	Address detection in Rx mode (banked)
		(8) ADDRESS_CONF_EN_ADDRESS_TX	(8) ADDRESS_CONF_EN_ADDRESS_TX	0x0	Address insertion in Tx mode (banked)
		(7:0) PREAMBLE_LENGTH_PREAMBLE_LEN	(7:0) PREAMBLE_LENGTH_PREAMBLE_LEN	0x0	Length of the preamble -1 (banked)
0x40040834	RF0_PACKET_EXTRA	(29:28) CONV_CODES_CONF_STOP_WORD_LEN	(29:28) CONV_CODES_CONF_STOP_WORD_LEN	0x0	Length of the stop word (banked)
		(27:26) CONV_CODES_CONF_CC_VITERBI_LEN	(27:26) CONV_CODES_CONF_CC_VITERBI_LEN	0x2	Set the memory length of the Viterbi decoder (banked)
		(25) CONV_CODES_	(25) CONV_CODES_	0x0	Stop word at the end of the

Address	Register Name	Register Write	Register Read	Default	Description
		CONF_CC_EN_TX_STOP	CONF_CC_EN_TX_STOP		transmission (banked)
		(24) CONV_CODES_ CONF_EN_CONV_CODE	(24) CONV_CODES_ CONF_EN_CONV_CODE	0x0	Convolutional codes (banked)
		(22) PACKET_EXTRA_ FIFO_REWIND	(22) PACKET_EXTRA_ FIFO_REWIND	0x0	Rewind the FIFO to the initial stage at the end of a Tx transmission (banked)
		(21) PACKET_EXTRA_ BLE_PREAMBLE	(21) PACKET_EXTRA_ BLE_PREAMBLE	0x1	Handle the preamble directly in Tx mode (PREAMBLE register is not used) according to the BLE standard (banked)
		(20) PACKET_EXTRA_ PKT_INFO_PRE_NPOST	(20) PACKET_EXTRA_ PKT_INFO_PRE_NPOST	0x0	Packet information sampling (banked)
		(19:18) PACKET_ EXTRA_PATTERN_MAX_ ERR	(19:18) PACKET_ EXTRA_PATTERN_MAX_ ERR	0x0	Unsigned value that specifies the maximum number of errors in the pattern recognition (banked)
		(17:16) PACKET_ EXTRA_PATTERN_WORD_ LEN	(17:16) PACKET_ EXTRA_PATTERN_ WORD_LEN	0x3	Pattern word length (banked)
		(15:0) ADDRESS_ BROADCAST_ADDRESS_ BR	(15:0) ADDRESS_ BROADCAST_ADDRESS_ BR	0x0	Broadcast address (banked)
0x40040838	RF0_CRC_POLYNOMIAL	(31:0) CRC_POLY	(31:0) CRC_POLY	0x80032D	CRC polynomial (banked)
0x4004083C	RF0_CRC_RST	(31:0) CRC_RST	(31:0) CRC_RST	0x555555	CRC reset value (banked)
0x40040840	RF0_REG10	(25:21) CONV_CODES_ PUNCT_CC_PUNCT_1	(25:21) CONV_ CODES_PUNCT_CC_ PUNCT_1	0x1	Puncture of the second convolutional code (banked)
		(20:16) CONV_CODES_ PUNCT_CC_PUNCT_0	(20:16) CONV_ CODES_PUNCT_CC_ PUNCT_0	0x1	Puncture of the first convolutional code (banked)
		(11) FRAC_CONF_TX_	(11) FRAC_CONF_TX_	0x0	Additional gain for fractional data-rates

Address	Register Name	Register Write	Register Read	Default	Description
		FRAC_GAIN	FRAC_GAIN		in Tx mode (banked)
		(10) FRAC_CONF_RX_ FRAC_GAIN	(10) FRAC_CONF_RX_ FRAC_GAIN	0x0	Additional gain for fractional data-rates in Rx mode (banked)
		(9) FRAC_CONF_TX_ EN_FRAC	(9) FRAC_CONF_TX_ EN_FRAC	0x0	Fractional data-rates in Tx mode (banked)
		(8) FRAC_CONF_RX_ EN_FRAC	(8) FRAC_CONF_RX_ EN_FRAC	0x0	Fractional data-rates in Rx mode (banked)
		(7:4) CONV_CODES_ POLY_CC_POLY_1	(7:4) CONV_CODES_ POLY_CC_POLY_1	0xD	Second convolutional code (banked)
		(3:0) CONV_CODES_ POLY_CC_POLY_0	(3:0) CONV_CODES_ POLY_CC_POLY_0	0xF	First convolutional code (banked)
0x40040844	RF0_REG11	(31) FILTER_GAIN_ LIN_FILTER	(31) FILTER_GAIN_ LIN_FILTER	0x0	Enable the linear filtering (banked)
		(30) FILTER_GAIN_ LOW_LIN_GAIN	(30) FILTER_GAIN_ LOW_LIN_GAIN	0x0	Reduce the total gain by two if the linear gain is set (banked)
		(29:27) FILTER_ GAIN_GAIN_M	(29:27) FILTER_ GAIN_GAIN_M	0x0	Mantissa of the final stage gain of the matched filter (banked)
		(26:24) FILTER_ GAIN_GAIN_E	(26:24) FILTER_ GAIN_GAIN_E	0x0	Exponent of the final stage gain of the matched filter (banked)
		(23:20) TX_MULT_TX_ MULT_EXP	(23:20) TX_MULT_ TX_MULT_EXP	0x2	Exponent of the Tx multiplier (banked)
		(19:16) TX_MULT_TX_ MULT_MAN	(19:16) TX_MULT_ TX_MULT_MAN	0x9	Mantissa of the Tx multiplier (banked)
		(15:12) TX_FRAC_ CONF_TX_FRAC_DEN	(15:12) TX_FRAC_ CONF_TX_FRAC_DEN	0x0	Denominator of the fractional data-rate in Tx mode (banked)
		(11:8) TX_FRAC_ CONF_TX_FRAC_NUM	(11:8) TX_FRAC_ CONF_TX_FRAC_NUM	0x0	Numerator of the fractional data-rate in Tx mode (banked)
		(7:4) RX_FRAC_CONF_	(7:4) RX_FRAC_	0x0	Denominator of the fractional data-rate

Address	Register Name	Register Write	Register Read	Default	Description
		RX_FRAC_DEN	CONF_RX_FRAC_DEN		in Rx mode (banked)
		(3:0) RX_FRAC_CONF_ RX_FRAC_NUM	(3:0) RX_FRAC_ CONF_RX_FRAC_NUM	0x0	Numerator of the fractional data-rate in Rx mode (banked)
0x40040848	RF0_TX_PULSE_SHAPE_1	(31:24) TX_PULSE_ SHAPE_1_TX_COEF4	(31:24) TX_PULSE_ SHAPE_1_TX_COEF4	0x0	Tx pulse shape coefficient 4 (banked)
		(23:16) TX_PULSE_ SHAPE_1_TX_COEF3	(23:16) TX_PULSE_ SHAPE_1_TX_COEF3	0x0	Tx pulse shape coefficient 3 (banked)
		(15:8) TX_PULSE_ SHAPE_1_TX_COEF2	(15:8) TX_PULSE_ SHAPE_1_TX_COEF2	0x0	Tx pulse shape coefficient 2 (banked)
		(7:0) TX_PULSE_ SHAPE_1_TX_COEF1	(7:0) TX_PULSE_ SHAPE_1_TX_COEF1	0x0	Tx pulse shape coefficient 1 (banked)
0x4004084C	RF0_TX_PULSE_SHAPE_2	(31:24) TX_PULSE_ SHAPE_2_TX_COEF8	(31:24) TX_PULSE_ SHAPE_2_TX_COEF8	0x2	Tx pulse shape coefficient 8 (banked)
		(23:16) TX_PULSE_ SHAPE_2_TX_COEF7	(23:16) TX_PULSE_ SHAPE_2_TX_COEF7	0x1	Tx pulse shape coefficient 7 (banked)
		(15:8) TX_PULSE_ SHAPE_2_TX_COEF6	(15:8) TX_PULSE_ SHAPE_2_TX_COEF6	0x0	Tx pulse shape coefficient 6 (banked)
		(7:0) TX_PULSE_ SHAPE_2_TX_COEF5	(7:0) TX_PULSE_ SHAPE_2_TX_COEF5	0x0	Tx pulse shape coefficient 5 (banked)
0x40040850	RF0_TX_PULSE_SHAPE_3	(31:24) TX_PULSE_ SHAPE_3_TX_COEF12	(31:24) TX_PULSE_ SHAPE_3_TX_COEF12	0x36	Tx pulse shape coefficient 12 (banked)
		(23:16) TX_PULSE_ SHAPE_3_TX_COEF11	(23:16) TX_PULSE_ SHAPE_3_TX_COEF11	0x20	Tx pulse shape coefficient 11 (banked)
		(15:8) TX_PULSE_ SHAPE_3_TX_COEF10	(15:8) TX_PULSE_ SHAPE_3_TX_COEF10	0x10	Tx pulse shape coefficient 10 (banked)
		(7:0) TX_PULSE_ SHAPE_3_TX_COEF9	(7:0) TX_PULSE_ SHAPE_3_TX_COEF9	0x7	Tx pulse shape coefficient 9 (banked)
0x40040854	RF0_TX_PULSE_SHAPE_4	(31:24) TX_PULSE_	(31:24) TX_PULSE_	0x7D	Tx pulse shape coefficient 16 (banked)

Address	Register Name	Register Write	Register Read	Default	Description
		SHAPE_4_TX_COEF16	SHAPE_4_TX_COEF16		
		(23:16) TX_PULSE_SHAPE_4_TX_COEF15	(23:16) TX_PULSE_SHAPE_4_TX_COEF15	0x75	Tx pulse shape coefficient 15 (banked)
		(15:8) TX_PULSE_SHAPE_4_TX_COEF14	(15:8) TX_PULSE_SHAPE_4_TX_COEF14	0x66	Tx pulse shape coefficient 14 (banked)
		(7:0) TX_PULSE_SHAPE_4_TX_COEF13	(7:0) TX_PULSE_SHAPE_4_TX_COEF13	0x4F	Tx pulse shape coefficient 13 (banked)
0x40040858	RF0_FRONTEND	(25:16) RX_IF_DIG_IF_DIG	(25:16) RX_IF_DIG_IF_DIG	0x40	IF frequency (banked)
		(14:11) FRONTEND_RESAMPLE_PH_GAIN	(14:11) FRONTEND_RESAMPLE_PH_GAIN	0x6	Gain of the phase resampling block (banked)
		(10:8) FRONTEND_RESAMPLE_RSSI_G2	(10:8) FRONTEND_RESAMPLE_RSSI_G2	0x0	Gain of the decimator in the RSSI resampling block (banked)
		(7:6) FRONTEND_RESAMPLE_RSSI_G1	(7:6) FRONTEND_RESAMPLE_RSSI_G1	0x0	Gain of the interpolator in the RSSI resampling block (banked)
		(5) FRONTEND_EN_RESAMPLE_RSSI	(5) FRONTEND_EN_RESAMPLE_RSSI	0x0	RSSI resampling (banked)
		(4) FRONTEND_EN_RESAMPLE_PHADC	(4) FRONTEND_EN_RESAMPLE_PHADC	0x1	Phase resampling (banked)
		(3:0) FRONTEND_DIV_PHADC	(3:0) FRONTEND_DIV_PHADC	0x0	Unsigned value that specifies the divider to obtain the phase ADC clock and RSSI (banked)
0x4004085C	RF0_RX_PULSE_SHAPE	(31:28) RX_PULSE_SHAPE_RX_COEF8	(31:28) RX_PULSE_SHAPE_RX_COEF8	0xF	Rx pulse shape coefficient 8 (banked)
		(27:24) RX_PULSE_SHAPE_RX_COEF7	(27:24) RX_PULSE_SHAPE_RX_COEF7	0xE	Rx pulse shape coefficient 7 (banked)
		(23:20) RX_PULSE_SHAPE_RX_COEF6	(23:20) RX_PULSE_SHAPE_RX_COEF6	0xC	Rx pulse shape coefficient 6 (banked)

Address	Register Name	Register Write	Register Read	Default	Description
		(19:16) RX_PULSE_ SHAPE_RX_COEF5	(19:16) RX_PULSE_ SHAPE_RX_COEF5	0xA	Rx pulse shape coefficient 5 (banked)
		(15:12) RX_PULSE_ SHAPE_RX_COEF4	(15:12) RX_PULSE_ SHAPE_RX_COEF4	0x7	Rx pulse shape coefficient 4 (banked)
		(11:8) RX_PULSE_ SHAPE_RX_COEF3	(11:8) RX_PULSE_ SHAPE_RX_COEF3	0x4	Rx pulse shape coefficient 3 (banked)
		(7:4) RX_PULSE_ SHAPE_RX_COEF2	(7:4) RX_PULSE_ SHAPE_RX_COEF2	0x2	Rx pulse shape coefficient 2 (banked)
		(3:0) RX_PULSE_ SHAPE_RX_COEF1	(3:0) RX_PULSE_ SHAPE_RX_COEF1	0x1	Rx pulse shape coefficient 1 (banked)
0x40040860	RF0_REG18	(28) DELAY_LINE_ CONF_MULTI_SYNC	(28) DELAY_LINE_ CONF_MULTI_SYNC	0x0	Detect multiple syncs (banked)
		(27:25) DELAY_LINE_ CONF_DL_ISI_THR	(27:25) DELAY_ LINE_CONF_DL_ISI_ THR	0x1	Threshold bias for ISI compensation in the delay line sync word comparator (banked)
		(22) DELAY_LINE_ CONF_EN_SYNC_OK_ DELAY_LINE	(22) DELAY_LINE_ CONF_EN_SYNC_OK_ DELAY_LINE	0x1	Use pattern_ok signal in delay line to synchronize the deserializer (banked)
		(21:20) DELAY_LINE_ CONF_MAX_ERR_IN_DL_ SYNC	(21:20) DELAY_ LINE_CONF_MAX_ERR_ IN_DL_SYNC	0x0	Set the maximum errors in the delay line sync detection (banked)
		(19) DELAY_LINE_ CONF_EN_NOT_CAUSAL	(19) DELAY_LINE_ CONF_EN_NOT_CAUSAL	0x0	Non causal processing (banked)
		(18:16) DELAY_LINE_ CONF_NC_SEL_OUT	(18:16) DELAY_ LINE_CONF_NC_SEL_ OUT	0x0	Select the output position for the non causal processing (banked)
		(15:8) FSK_FCR_AMP_ 1_FSK_FCR_AMP1	(15:8) FSK_FCR_ AMP_1_FSK_FCR_AMP1	0x1B	FSK amplitude low (banked)
		(6:4) CARRIER_	(6:4) CARRIER_	0x5	Mantissa of the carrier recovery

Address	Register Name	Register Write	Register Read	Default	Description
		RECOVERY_EXTRA_ FREQ_LIMIT_MAN	RECOVERY_EXTRA_ FREQ_LIMIT_MAN		frequency limit (banked)
		(2:0) CARRIER_ RECOVERY_EXTRA_ FREQ_LIMIT_EXP	(2:0) CARRIER_ RECOVERY_EXTRA_ FREQ_LIMIT_EXP	0x0	Exponent of the carrier recovery frequency limit (banked)
0x40040864	RF0_REG19	(30) RSSI_BANK_EN_ RSSI_DITHER	(30) RSSI_BANK_EN_ RSSI_DITHER	0x0	Speed on the RSSI triangular dithering signal (banked)
		(29) RSSI_BANK_ FAST_RSSI	(29) RSSI_BANK_ FAST_RSSI	0x0	RSSI filtering speed (banked)
		(28) RSSI_BANK_EN_ FAST_PRE_SYNC	(28) RSSI_BANK_EN_ FAST_PRE_SYNC	0x1	Fast mode switching during the preamble reception (banked)
		(27:24) RSSI_BANK_ TAU_RSSI_FILTERING	(27:24) RSSI_BANK_ TAU_RSSI_FILTERING	0x1	Time constant of the RSSI filtering block (banked)
		(20) DECISION_USE_ VIT_SOFT	(20) DECISION_USE_ VIT_SOFT	0x0	Viterbi soft decoding (banked)
		(19:18) DECISION_ VITERBI_LEN	(19:18) DECISION_ VITERBI_LEN	0x2	Set the Viterbi path length (banked)
		(17) DECISION_ VITERBI_POW_NLIN	(17) DECISION_ VITERBI_POW_NLIN	0x1	Viterbi algorithm uses power instead of amplitude to evaluate the error on the path (banked)
		(16) DECISION_EN_ VITERBI_GFSK	(16) DECISION_EN_ VITERBI_GFSK	0x1	Viterbi algorithm for the GFSK decoding (banked)
		(15:8) FSK_FCR_AMP_ 3_FSK_FCR_AMP3	(15:8) FSK_FCR_ AMP_3_FSK_FCR_AMP3	0x44	FSK amplitude high (banked)
		(7:0) FSK_FCR_AMP_ 2_FSK_FCR_AMP2	(7:0) FSK_FCR_AMP_ 2_FSK_FCR_AMP2	0x30	FSK amplitude mid (banked)
0x40040868	RF0_REG1A	(28:24) PA_PWR_PA_ PWR	(28:24) PA_PWR_PA_ PWR	0xC	Signed value that sets the PA power

Address	Register Name	Register Write	Register Read	Default	Description
		(22) RSSI_BANK_ALT_USE_RSSI_ALT	(22) RSSI_BANK_ALT_USE_RSSI_ALT	0x0	Use alternative RRSI configuration (banked)
		(21) RSSI_BANK_ALT_FAST_RSSI_ALT	(21) RSSI_BANK_ALT_FAST_RSSI_ALT	0x0	RSSI filtering speed (banked)
		(19:16) RSSI_BANK_ALT_TAU_RSSI_FILTERING_ALT	(19:16) RSSI_BANK_ALT_TAU_RSSI_FILTERING_ALT	0x3	Time constant of the RSSI filtering block (banked)
		(15:0) CORRECT_CFREQ_IF_CORRECT_CFREQ_IF	(15:0) CORRECT_CFREQ_IF_CORRECT_CFREQ_IF	0x1555	Unsigned value that specifies the IF for the Rx mode (banked)
0x4004086C	RF0_REG1B	(31) PLL_BANK_EN_LOW_CHP_BIAS_TX	(31) PLL_BANK_EN_LOW_CHP_BIAS_TX	0x0	Set the en_low_chp_bias bit in Tx mode (banked)
		(30) PLL_BANK_EN_LOW_CHP_BIAS_RX	(30) PLL_BANK_EN_LOW_CHP_BIAS_RX	0x1	Set the en_low_chp_bias bit in Rx mode (banked)
		(29:28) PLL_BANK_PLL_FILTER_RES_TRIM_TX	(29:28) PLL_BANK_PLL_FILTER_RES_TRIM_TX	0x3	Modify the value of the loop filter resistor R2 when bit 5 is high in Tx mode (banked)
		(27:24) PLL_BANK_IQ_PLL_0_TX	(27:24) PLL_BANK_IQ_PLL_0_TX	0x4	Charge pump bias for Tx case (banked)
		(22) PLL_BANK_LOW_DR_TX	(22) PLL_BANK_LOW_DR_TX	0x0	Enable low data-rate mode in Tx mode (banked)
		(21:20) PLL_BANK_PLL_FILTER_RES_TRIM_RX	(21:20) PLL_BANK_PLL_FILTER_RES_TRIM_RX	0x0	Modify the value of the loop filter resistor R2 when bit 5 is high in Rx mode (banked)
		(19:16) PLL_BANK_IQ_PLL_0_RX	(19:16) PLL_BANK_IQ_PLL_0_RX	0xB	Charge pump bias for Rx (banked)
		(15) ANACLK_USE_NEW_ANACK	(15) ANACLK_USE_NEW_ANACK	0x0	Use the new analog clock generator (banked)
		(13:12) ANACLK_DIV_	(13:12) ANACLK_	0x0	Set the master clock divider for the

Address	Register Name	Register Write	Register Read	Default	Description
		CK_RSSI	DIV_CK_RSSI		RSSI clock (banked)
		(11:10) ANACLK_DIV_CK_FILT	(11:10) ANACLK_DIV_CK_FILT	0x0	Set the master clock divider for the channel filter clock (banked)
		(9:8) ANACLK_DIV_CK_PHADC	(9:8) ANACLK_DIV_CK_PHADC	0x0	Set the master clock divider for the phase ADC clock (banked)
		(7:4) ANACLK_DIV_RSSI	(7:4) ANACLK_DIV_RSSI	0x1	Unsigned value that specifies the division factor for the clock controlling the RSSI (banked)
		(3:0) ANACLK_DIV_FILT	(3:0) ANACLK_DIV_FILT	0x5	Unsigned value that specifies the division factor for the clock controlling the channel filter (banked)
0x40040870	RF0_RSSI_CTRL	(31:30) RSSI_CTRL_AGC_DECAY_TAU	(31:30) RSSI_CTRL_AGC_DECAY_TAU	0x3	Time constant of the decay speed
		(29) RSSI_CTRL_AGC_USE_LNA	(29) RSSI_CTRL_AGC_USE_LNA	0x1	AGC algorithm uses LNA bias
		(28) RSSI_CTRL_AGC_MODE	(28) RSSI_CTRL_AGC_MODE	0x1	AGC algorithm selection
		(27:26) RSSI_CTRL_AGC_WAIT	(27:26) RSSI_CTRL_AGC_WAIT	0x3	Set the wait time of the AGC after switching between state
		(25) RSSI_CTRL_PAYLOAD_BLOCKS_AGC	(25) RSSI_CTRL_PAYLOAD_BLOCKS_AGC	0x1	AGC payload blocking
		(24) RSSI_CTRL_BYPASS_AGC	(24) RSSI_CTRL_BYPASS_AGC	0x0	AGC algorithm bypass
		(20:16) PA_PWR_OFFSET_PA_PWR_OFFSET	(20:16) PA_PWR_OFFSET_PA_PWR_OFFSET	0x0	Signed value that sets the PA power (banked)
		(12:8) FILTER_BIAS_IQ_FI_BW	(12:8) FILTER_BIAS_IQ_FI_BW	0x14	Bias for the bandwidth of the channel filter (banked)

Address	Register Name	Register Write	Register Read	Default	Description
		(4:0) FILTER_BIAS_IQ_FI_FC	(4:0) FILTER_BIAS_IQ_FI_FC	0xB	Bias for the central frequency of the channel filter (banked)
0x40040874	RF0_REG1D	(31:28) AGC_PEAK_DET_PEAK_DET_TAU	(31:28) AGC_PEAK_DET_PEAK_DET_TAU	0x7	Time constant of the peak detector monostable circuit
		(27:26) AGC_PEAK_DET_PEAK_DET_THR_LOW	(27:26) AGC_PEAK_DET_PEAK_DET_THR_LOW	0x0	Threshold for the low level of the peak detector
		(25) AGC_PEAK_DET_PEAK_DET_THR_HIGH	(25) AGC_PEAK_DET_PEAK_DET_THR_HIGH	0x0	Threshold for the high level of the peak detector
		(24) AGC_PEAK_DET_EN_AGC_PEAK	(24) AGC_PEAK_DET_EN_AGC_PEAK	0x1	Enable AGC peak detector
		(23:16) AGC_THR_HIGH_AGC_THR_HIGH	(23:16) AGC_THR_HIGH_AGC_THR_HIGH	0x69	AGC threshold high level (banked)
		(15:8) AGC_THR_LOW_AGC_THR_LOW	(15:8) AGC_THR_LOW_AGC_THR_LOW	0x40	AGC threshold low level (banked)
		(7:4) ATT_CTRL_ATT_CTRL_MAX	(7:4) ATT_CTRL_ATT_CTRL_MAX	0xB	Maximum attenuation level in AGC algorithm
		(3:0) ATT_CTRL_SET_RX_ATT_CTRL	(3:0) ATT_CTRL_SET_RX_ATT_CTRL	0x0	Attenuation level if the AGC is bypassed
0x40040878	RF0_AGC_LUT1	(31:22) AGC_LUT_1_AGC_LEVEL_2_LO	(31:22) AGC_LUT_1_AGC_LEVEL_2_LO	0x280	AGC values level 2 (LSB)
		(21:11) AGC_LUT_1_AGC_LEVEL_1	(21:11) AGC_LUT_1_AGC_LEVEL_1	0x80	AGC values level 1
		(10:0) AGC_LUT_1_AGC_LEVEL_0	(10:0) AGC_LUT_1_AGC_LEVEL_0	0x0	AGC values level 0
0x4004087C	RF0_AGC_LUT2	(31:23) AGC_LUT_2_AGC_LEVEL_5_LO	(31:23) AGC_LUT_2_AGC_LEVEL_5_LO	0x84	AGC values level 5 (LSB)
		(22:12) AGC_LUT_2_	(22:12) AGC_LUT_2_	0x284	AGC values level 4

Address	Register Name	Register Write	Register Read	Default	Description
		AGC_LEVEL_4	AGC_LEVEL_4		
		(11:1) AGC_LUT_2_ AGC_LEVEL_3	(11:1) AGC_LUT_2_ AGC_LEVEL_3	0x480	AGC values level 3
		(0) AGC_LUT_2_AGC_ LEVEL_2_HI	(0) AGC_LUT_2_AGC_ LEVEL_2_HI	0x0	AGC values level 2 (MSB)
0x40040880	RF0_AGC_LUT3	(31:24) AGC_LUT_3_ AGC_LEVEL_8_LO	(31:24) AGC_LUT_3_ AGC_LEVEL_8_LO	0x9D	AGC values level 8 (LSB)
		(23:13) AGC_LUT_3_ AGC_LEVEL_7	(23:13) AGC_LUT_3_ AGC_LEVEL_7	0x495	AGC values level 7
		(12:2) AGC_LUT_3_ AGC_LEVEL_6	(12:2) AGC_LUT_3_ AGC_LEVEL_6	0x485	AGC values level 6
		(1:0) AGC_LUT_3_ AGC_LEVEL_5_HI	(1:0) AGC_LUT_3_ AGC_LEVEL_5_HI	0x2	AGC values level 5 (MSB)
0x40040884	RF0_AGC_LUT4	(31:25) AGC_LUT_4_ AGC_LEVEL_11_LO	(31:25) AGC_LUT_4_ AGC_LEVEL_11_LO	0x7F	AGC values level 11 (LSB)
		(24:14) AGC_LUT_4_ AGC_LEVEL_10	(24:14) AGC_LUT_4_ AGC_LEVEL_10	0x4FF	AGC values level 10
		(13:3) AGC_LUT_4_ AGC_LEVEL_9	(13:3) AGC_LUT_4_ AGC_LEVEL_9	0x49F	AGC values level 9
		(2:0) AGC_LUT_4_ AGC_LEVEL_8_HI	(2:0) AGC_LUT_4_ AGC_LEVEL_8_HI	0x4	AGC values level 8 (MSB)
0x40040888	RF0_AGC_LUT5	(26:25) IEEE802154_ OPTS_CNT_LIM_802154	(26:25) IEEE802154_OPTS_ CNT_LIM_802154	0x2	Set the number of samples to wait before increasing the threshold
		(24:22) IEEE802154_ OPTS_CNT_OK_INC_ 802154	(24:22) IEEE802154_OPTS_ CNT_OK_INC_802154	0x4	Set the increment to the counter that indicates that the correlators peaks are coherent
		(21) IEEE802154_	(21) IEEE802154_	0x1	Enable the new algorithm working in

Address	Register Name	Register Write	Register Read	Default	Description
		OPTS_USE_OS_802154	OPTS_USE_OS_802154		the oversampled domain for the demodulation of the IEEE 802.15.4 protocol
		(20) IEEE802154_ OPTS_EN_DW_TEST	(20) IEEE802154_ OPTS_EN_DW_TEST	0x0	Tx data-whitening before the convolutional code block
		(18:16) IEEE802154_ OPTS_C2B_THR	(18:16) IEEE802154_OPTS_ C2B_THR	0x4	Threshold of the chip2bit correlator of the IEEE 802.15.4 protocol
		(13:12) DATA_ STREAMS_BER_CLK_ MODE	(13:12) DATA_ STREAMS_BER_CLK_ MODE	0x0	Set the clock output mode for BER mode or RW mode
		(10) DATA_STREAMS_ RX_DATA_NOT_SAMPLED	(10) DATA_STREAMS_ RX_DATA_NOT_ SAMPLED	0x0	Signal rx_data in test modes sampling
		(9) DATA_STREAMS_ PHASE_GREY	(9) DATA_STREAMS_ PHASE_GREY	0x0	Phase signal encoding
		(8) DATA_STREAMS_ TX_IN_CLK_TOGGLE	(8) DATA_STREAMS_ TX_IN_CLK_TOGGLE	0x0	Input clock
		(3:0) AGC_LUT_5_ AGC_LEVEL_11_HI	(3:0) AGC_LUT_5_ AGC_LEVEL_11_HI	0xE	AGC values level 11 (MSB)
0x4004088C	RF0_AGC_ATT1	(31:30) AGC_ATT_1_ AGC_ATT_AB_LO	(31:30) AGC_ATT_1_ AGC_ATT_AB_LO	0x3	AGC attenuation step 10/11 (LSB)
		(29:27) AGC_ATT_1_ AGC_ATT_9A	(29:27) AGC_ATT_1_ AGC_ATT_9A	0x5	AGC attenuation step 9/10
		(26:24) AGC_ATT_1_ AGC_ATT_89	(26:24) AGC_ATT_1_ AGC_ATT_89	0x3	AGC attenuation step 8/9
		(23:21) AGC_ATT_1_ AGC_ATT_78	(23:21) AGC_ATT_1_ AGC_ATT_78	0x4	AGC attenuation step 7/8
		(20:18) AGC_ATT_1_	(20:18) AGC_ATT_1_	0x3	AGC attenuation step 6/7

Address	Register Name	Register Write	Register Read	Default	Description
		AGC_ATT_67	AGC_ATT_67		
		(17:15) AGC_ATT_1_ AGC_ATT_56	(17:15) AGC_ATT_1_ AGC_ATT_56	0x2	AGC attenuation step 5/6
		(14:12) AGC_ATT_1_ AGC_ATT_45	(14:12) AGC_ATT_1_ AGC_ATT_45	0x2	AGC attenuation step 4/5
		(11:9) AGC_ATT_1_ AGC_ATT_34	(11:9) AGC_ATT_1_ AGC_ATT_34	0x2	AGC attenuation step 3/4
		(8:6) AGC_ATT_1_ AGC_ATT_23	(8:6) AGC_ATT_1_ AGC_ATT_23	0x1	AGC attenuation step 2/3
		(5:3) AGC_ATT_1_ AGC_ATT_12	(5:3) AGC_ATT_1_ AGC_ATT_12	0x1	AGC attenuation step 1/2
		(2:0) AGC_ATT_1_ AGC_ATT_01	(2:0) AGC_ATT_1_ AGC_ATT_01	0x4	AGC attenuation step 0/1
0x40040890	RF0_AGC_ATT2	(31:28) TIMINGS_3_ T_DLL	(31:28) TIMINGS_3_ T_DLL	0x2	Time needed by the DLL blocks to switch on
		(27:24) TIMINGS_3_ T_PLL_TX	(27:24) TIMINGS_3_ T_PLL_TX	0x2	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	0xC	Time needed by the subband algorithm to calibrate in Tx mode
		(19:16) TIMINGS_2_ T_TX_RF	(19:16) TIMINGS_2_ T_TX_RF	0x1	Time needed by the RF blocks to switch on in Tx mode
		(14:12) TIMINGS_1_ T_GRANULARITY_TX	(14:12) TIMINGS_1_ T_GRANULARITY_TX	0x3	Define the granularity of the timer in Tx mode
		(10:8) TIMINGS_1_T_ GRANULARITY_RX	(10:8) TIMINGS_1_ T_GRANULARITY_RX	0x5	Define the granularity of the timer in Rx mode
		(1) AGC_ATT_2_AGC_ ATT_1DB	(1) AGC_ATT_2_AGC_ ATT_1DB	0x0	Attenuation steps
		(0) AGC_ATT_2_AGC_	(0) AGC_ATT_2_AGC_	0x1	AGC attenuation step 10/11 (MSB)

Address	Register Name	Register Write	Register Read	Default	Description
		ATT_AB_HI	ATT_AB_HI		
0x40040894	RF0_REG25	(31) TIMEOUT_EN_RX_TIMEOUT	(31) TIMEOUT_EN_RX_TIMEOUT	0x0	Timeout of the Rx when the system is on FSM mode
		(30:28) TIMEOUT_T_TIMEOUT_GR	(30:28) TIMEOUT_T_TIMEOUT_GR	0x0	Granularity of the timer in timeout Rx mode
		(27:24) TIMEOUT_T_RX_TIMEOUT	(27:24) TIMEOUT_T_RX_TIMEOUT	0x0	Time that has to occur before the timeout
		(21) TIMING_FAST_RX_EN_FAST_RX_TXFILT	(21) TIMING_FAST_RX_EN_FAST_RX_TXFILT	0x0	Filter Tx configuration for the fast Rx PLL
		(20) TIMING_FAST_RX_EN_FAST_RX	(20) TIMING_FAST_RX_EN_FAST_RX	0x0	Fast Rx PLL
		(19:16) TIMING_FAST_RX_T_RX_FAST_CHP	(19:16) TIMING_FAST_RX_T_RX_FAST_CHP	0x0	Time to switch off the fast CHP in Rx mode
		(15:12) TIMINGS_5_T_RX_RF	(15:12) TIMINGS_5_T_RX_RF	0x0	Time needed by the RF blocks to switch on in Rx mode
		(11:8) TIMINGS_5_T_RX_BB	(11:8) TIMINGS_5_T_RX_BB	0x1	Time needed by the BB blocks to switch on in Rx mode
		(7:4) TIMINGS_4_T_SUBBAND_RX	(7:4) TIMINGS_4_T_SUBBAND_RX	0x5	Time needed by the subband algorithm to calibrate in Rx mode
		(3:0) TIMINGS_4_T_PLL_RX	(3:0) TIMINGS_4_T_PLL_RX	0x1	Time needed by the PLL blocks to switch on in Rx mode
0x40040898	RF0_BIAS_0_2	(31:28) BIAS_2_IQ_RXTX_6	(31:28) BIAS_2_IQ_RXTX_6	0x3	VCOM_MX bias
		(27:24) BIAS_2_IQ_RXTX_5	(27:24) BIAS_2_IQ_RXTX_5	0x8	VCOM_LO bias
		(23:20) BIAS_1_IQ_	(23:20) BIAS_1_IQ_	0x6	PrePA Casc bias

Address	Register Name	Register Write	Register Read	Default	Description
		RXTX_3	RXTX_3		
		(19:16) BIAS_1_IQ_ RXTX_2	(19:16) BIAS_1_IQ_ RXTX_2	0x6	PrePA In bias
		(15:12) BIAS_0_IQ_ RXTX_1	(15:12) BIAS_0_IQ_ RXTX_1	0x7	PA backoff bias
		(11:8) BIAS_0_IQ_ RXTX_0	(11:8) BIAS_0_IQ_ RXTX_0	0x3	PA bias
		(7) INTERFACE_CONF_ EN_SYNC_IFACE	(7) INTERFACE_ CONF_EN_SYNC_IFACE	0x0	Interfaces resynchronization
		(6:4) INTERFACE_ CONF_APB_WAIT_STATE	(6:4) INTERFACE_ CONF_APB_WAIT_ STATE	0x0	Select the number of wait states during the APB transaction
		(1:0) INTERFACE_ CONF_SPI_SELECT	(1:0) INTERFACE_ CONF_SPI_SELECT	0x0	Select the SPI mode
0x4004089C	RF0_BIAS_3_6	(31:28) BIAS_6_IQ_ BB_0	(31:28) BIAS_6_IQ_ BB_0	0x7	ACD_O bias
		(27:24) BIAS_6_IQ_ PLL_3	(27:24) BIAS_6_IQ_ PLL_3	0x7	DLL bias
		(23:20) BIAS_5_IQ_ PLL_4_RX	(23:20) BIAS_5_IQ_ PLL_4_RX	0x8	VCO bias for Rx mode
		(19:16) BIAS_5_IQ_ PLL_4_TX	(19:16) BIAS_5_IQ_ PLL_4_TX	0xA	VCO bias for Tx mode
		(15:12) BIAS_4_IQ_ PLL_2	(15:12) BIAS_4_IQ_ PLL_2	0x7	Sub-band comparator bias
		(11:8) BIAS_4_IQ_ PLL_1	(11:8) BIAS_4_IQ_ PLL_1	0x4	Dynamic divider bias
		(7:4) BIAS_3_IQ_ RXTX_8	(7:4) BIAS_3_IQ_ RXTX_8	0x7	IFA ctrl_c bias

Address	Register Name	Register Write	Register Read	Default	Description
		(3:0) BIAS_3_IQ_RXTX_7	(3:0) BIAS_3_IQ_RXTX_7	0x7	IFA ctrl_r bias
0x400408A0	RF0_BIAS_7_9	(31:28) BIAS_9_IQ_BB_6	(31:28) BIAS_9_IQ_BB_6	0x9	Peak detector threshold bias 0
		(27:24) BIAS_9_IQ_BB_5	(27:24) BIAS_9_IQ_BB_5	0x5	Peak detector bias
		(23:20) SWCAP_FSM_SB_CAP_RX	(23:20) SWCAP_FSM_SB_CAP_RX	0x0	VCO subband selection (FSM in Rx mode)
		(19:16) SWCAP_FSM_SB_CAP_TX	(19:16) SWCAP_FSM_SB_CAP_TX	0x0	VCO subband selection (FSM in Tx mode)
		(15:12) BIAS_8_IQ_BB_4	(15:12) BIAS_8_IQ_BB_4	0x9	RSSI_D bias
		(11:8) BIAS_8_IQ_BB_3	(11:8) BIAS_8_IQ_BB_3	0xF	RSSI_G bias
		(7:4) BIAS_7_IQ_BB_2	(7:4) BIAS_7_IQ_BB_2	0x6	ACD_L bias
		(3:0) BIAS_7_IQ_BB_1	(3:0) BIAS_7_IQ_BB_1	0x6	ACD_C bias
0x400408A4	RF0_BIAS_10_12	(30) SD_MASH_MASH_DITHER_TYPE	(30) SD_MASH_MASH_DITHER_TYPE	0x0	Enable the new dithering scheme
		(29) SD_MASH_MASH_ENABLE	(29) SD_MASH_MASH_ENABLE	0x0	Enable the sigma delta mash
		(28) SD_MASH_MASH_DITHER	(28) SD_MASH_MASH_DITHER	0x1	Enable dithering on the sigma delta mash
		(27:25) SD_MASH_MASH_ORDER	(27:25) SD_MASH_MASH_ORDER	0x3	Order of the sigma delta mash
		(24) SD_MASH_MASH_RSTB	(24) SD_MASH_MASH_RSTB	0x1	Reset of the sigma delta mash (active low)

Address	Register Name	Register Write	Register Read	Default	Description
		(23:20) BIAS_12_ LNA_AGC_BIAS_3	(23:20) BIAS_12_ LNA_AGC_BIAS_3	0x6	LNA bias for AGC level 3
		(19:16) BIAS_12_ LNA_AGC_BIAS_2	(19:16) BIAS_12_ LNA_AGC_BIAS_2	0x7	LNA bias for AGC level 2
		(15:12) BIAS_11_ LNA_AGC_BIAS_1	(15:12) BIAS_11_ LNA_AGC_BIAS_1	0x8	LNA bias for AGC level 1
		(11:8) BIAS_11_LNA_ AGC_BIAS_0	(11:8) BIAS_11_ LNA_AGC_BIAS_0	0x9	LNA bias for AGC level 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	0x6	Peak detector threshold bias 2
0x400408A8	RF0_REG2A	(27:24) SD_MASH_ MASK_MASH_MASK	(27:24) SD_MASH_ MASK_MASH_MASK	0x0	Mask the n LSB of the fractional part of the MASH (debug only)
		(19) BIAS_EN_2_EN_ PTAT	(19) BIAS_EN_2_EN_ PTAT	0x1	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)
0x400408AC	RF0_PLL_CTRL	(26) PLL_CTRL_ DISABLE_CHP_SBS	(26) PLL_CTRL_ DISABLE_CHP_SBS	0x0	Charge-pump disabling during sub-band selection (FLL and frequency ratios)
		(25) PLL_CTRL_PLL_	(25) PLL_CTRL_PLL_	0x1	PLL frequency

Address	Register Name	Register Write	Register Read	Default	Description
		RX_48MEG	RX_48MEG		
		(24) PLL_CTRL_ SWCAP_TX_SAME_RX	(24) PLL_CTRL_ SWCAP_TX_SAME_RX	0x0	Registers for Rx and Tx modes swcap in case of swcap_fsm=1
		(23) PLL_CTRL_ SWCAP_FSM	(23) PLL_CTRL_ SWCAP_FSM	0x1	Selection of the swcap_fsm register
		(22) PLL_CTRL_DLL_ RSTB	(22) PLL_CTRL_DLL_ RSTB	0x1	Reset signal of the DLL (active low)
		(21:18) PLL_CTRL_ VCO_SUBBAND_TRIM	(21:18) PLL_CTRL_ VCO_SUBBAND_TRIM	0x0	VCO sub-band selection bits
		(17) PLL_CTRL_SUB_ SEL_OFFS_EN	(17) PLL_CTRL_SUB_ SEL_OFFS_EN	0x0	Add offset to sub-band selection comparator
		(16) PLL_CTRL_DIV2_ CLKVCO_TEST_EN	(16) PLL_CTRL_ DIV2_CLKVCO_TEST_ EN	0x0	VCO signal divided by the programmable divider
		(15) PLL_CTRL_ VCODIV_CLK_TEST_EN	(15) PLL_CTRL_ VCODIV_CLK_TEST_EN	0x0	Output on GPIO the VCO signal divided by the programmable divider
		(13) PLL_CTRL_CHP_ DEAD_ZONE_EN	(13) PLL_CTRL_CHP_ DEAD_ZONE_EN	0x0	Charge-pump dead zone
		(12:11) PLL_CTRL_ CHP_CURR_OFF_TRIM_ TX	(12:11) PLL_CTRL_ CHP_CURR_OFF_TRIM_ TX	0x3	Charge-pump offset current values selection bits in Tx mode
		(10:9) PLL_CTRL_ CHP_CURR_OFF_TRIM_ RX	(10:9) PLL_CTRL_ CHP_CURR_OFF_TRIM_ RX	0x3	Charge-pump offset current values selection bits in Rx mode
		(8) PLL_CTRL_HIGH_ BW_FILTER_EN_TX	(8) PLL_CTRL_HIGH_ BW_FILTER_EN_TX	0x1	PLL filter high bandwidth needed in Tx mode
		(7) PLL_CTRL_HIGH_ BW_FILTER_EN_RX	(7) PLL_CTRL_HIGH_ BW_FILTER_EN_RX	0x0	PLL filter high bandwidth needed in Rx mode

Address	Register Name	Register Write	Register Read	Default	Description
		(6) PLL_CTRL_FAST_CHP_EN_TX	(6) PLL_CTRL_FAST_CHP_EN_TX	0x1	High current output of the charge-pump for PLL Tx high bandwidth mode
		(5) PLL_CTRL_FAST_CHP_EN_RX	(5) PLL_CTRL_FAST_CHP_EN_RX	0x0	High current output of the charge-pump for PLL Rx high bandwidth mode
		(4:3) PLL_CTRL_CHP_MODE_TRIM	(4:3) PLL_CTRL_CHP_MODE_TRIM	0x0	Select the frequency inside sub-band selection
		(2) PLL_CTRL_CHP_CMC_EN	(2) PLL_CTRL_CHP_CMC_EN	0x1	Common mode control block of the charge-pump
		(1) PLL_CTRL_CHP_CURR_OFF_EN_TX	(1) PLL_CTRL_CHP_CURR_OFF_EN_TX	0x1	Charge-pump offset current in Tx mode
		(0) PLL_CTRL_CHP_CURR_OFF_EN_RX	(0) PLL_CTRL_CHP_CURR_OFF_EN_RX	0x0	Charge-pump offset current in Rx mode
0x400408B0	RF0_DLL_CTRL	(31:29) RSSI_TUN_1_RSSI_TUN_GAIN	(31:29) RSSI_TUN_1_RSSI_TUN_GAIN	0x1	RSSI tuning for gain
		(28:24) RSSI_TUN_1_RSSI_ODD_OFFSET	(28:24) RSSI_TUN_1_RSSI_ODD_OFFSET	0x4	RSSI tuning for odd stages (offset to the even triangular wave)
		(23:20) RSSI_TUN_1_RSSI_EVEN_MAX	(23:20) RSSI_TUN_1_RSSI_EVEN_MAX	0x1	RSSI tuning for even stages (maximum value of the triangular wave)
		(19:16) RSSI_TUN_1_RSSI_EVEN_MIN	(19:16) RSSI_TUN_1_RSSI_EVEN_MIN	0x1	RSSI tuning for even stages (minimum value of the triangular wave)
		(12) DLL_CTRL_CK_LAST_SEL_DELAY	(12) DLL_CTRL_CK_LAST_SEL_DELAY	0x0	Last SEL delay
		(11) DLL_CTRL_CK_FIRST_SEL_DELAY	(11) DLL_CTRL_CK_FIRST_SEL_DELAY	0x0	First SEL delay
		(10) DLL_CTRL_CK_EXT_SEL	(10) DLL_CTRL_CK_EXT_SEL	0x0	Input clock selection
		(9) DLL_CTRL_CK_DIG_EN	(9) DLL_CTRL_CK_DIG_EN	0x0	Alternate ck_dig pin to output the PLL reference clock signal

Address	Register Name	Register Write	Register Read	Default	Description
		(8) DLL_CTRL_CK_TEST_EN	(8) DLL_CTRL_CK_TEST_EN	0x0	Output on GPIO the PLL reference clock signal via ck_test pin
		(7) DLL_CTRL_TOO_FAST_ENB	(7) DLL_CTRL_TOO_FAST_ENB	0x0	Lock range phase detector
		(6) DLL_CTRL_LOCKED_DET_EN	(6) DLL_CTRL_LOCKED_DET_EN	0x1	Reference frequency multiplier locked detector
		(5) DLL_CTRL_LOCKED_AUTO_CHECK_EN	(5) DLL_CTRL_LOCKED_AUTO_CHECK_EN	0x1	Frequency multiplier is out of lock (usually because some input clocks from ck_xtal or ck_ext are missing)
		(4) DLL_CTRL_FAST_ENB	(4) DLL_CTRL_FAST_ENB	0x0	Disable fast mode locking of the reference frequency multiplier
		(3:2) DLL_CTRL_CK_SEL_TX	(3:2) DLL_CTRL_CK_SEL_TX	0x1	Selection of the clock used as frequency reference of the PLL in Tx mode (also to ck_test and ck_dig outputs)
		(1:0) DLL_CTRL_CK_SEL_RX	(1:0) DLL_CTRL_CK_SEL_RX	0x0	Selection of the clock used as frequency reference of the PLL in Rx mode (also to ck_test and ck_dig outputs)
0x400408B4	RF0_REG2D	(29:28) PA_CONF_SW_CN	(29:28) PA_CONF_SW_CN	0x0	Harmonic 2 notch tuning
		(27) PA_CONF_TX_SWITCHPA	(27) PA_CONF_TX_SWITCHPA	0x0	Switch PA
		(26) PA_CONF_TX_0DBM	(26) PA_CONF_TX_0DBM	0x1	Select between PPA and PA
		(25) PA_CONF_LIN_RAMP	(25) PA_CONF_LIN_RAMP	0x0	PA ramp-up linearization
		(24) PA_CONF_MIN_PA_PWR	(24) PA_CONF_MIN_PA_PWR	0x1	Set the minimum power during the PA ramp-up

Address	Register Name	Register Write	Register Read	Default	Description
		(23) CTRL_RX_ SWITCH_LP	(23) CTRL_RX_ SWITCH_LP	0x0	Switch the low-pass filter in the Rx chain
		(22) CTRL_RX_USE_ PEAK_DETECTOR	(22) CTRL_RX_USE_ PEAK_DETECTOR	0x1	Peak detector powering
		(21) CTRL_RX_START_ MIX_ON_CAL	(21) CTRL_RX_ START_MIX_ON_CAL	0x0	Mixer enabling
		(20:16) CTRL_RX_ CTRL_RX	(20:16) CTRL_RX_ CTRL_RX	0xF	Rx control
		(15) CTRL_ADC_ PHADC_THERM_OUT_EN	(15) CTRL_ADC_ PHADC_THERM_OUT_EN	0x1	Enable the buffers of phase ADC thermometric code (banked)
		(14:13) CTRL_ADC_ PHADC_DELLATCH	(14:13) CTRL_ADC_ PHADC_DELLATCH	0x1	Phase ADC delay latch trimming (banked)
		(12:8) CTRL_ADC_ CTRL_ADC	(12:8) CTRL_ADC_ CTRL_ADC	0x5	Phase ADC control (banked)
		(6:5) RSSI_TUN_2_ RSSI_TRI_CK_DIV	(6:5) RSSI_TUN_2_ RSSI_TRI_CK_DIV	0x0	Speed on the RSSI triangular dithering signal (cf reg RSSI_TUN)
		(4) RSSI_TUN_2_ RSSI_ONE_CK_RSSI_ PHADC	(4) RSSI_TUN_2_ RSSI_ONE_CK_RSSI_ PHADC	0x0	RSSI and phase ADC clocks sharing
		(3) RSSI_TUN_2_ RSSI_FULL	(3) RSSI_TUN_2_ RSSI_FULL	0x1	RSSI full scale
		(2) RSSI_TUN_2_ RSSI_1DB	(2) RSSI_TUN_2_ RSSI_1DB	0x0	LSB resolution
		(1:0) RSSI_TUN_2_ RSSI_PRE_ATT	(1:0) RSSI_TUN_2_ RSSI_PRE_ATT	0x3	Pre attenuation of the RSSI signal
0x400408B8	RF0_REG2E	(31:24) XTAL_TRIM_ XTAL_TRIM_INIT	(31:24) XTAL_TRIM_ XTAL_TRIM_INIT	0x60	Initial trimming of the XTAL
		(23:16) XTAL_TRIM_	(23:16) XTAL_TRIM_	0x60	Trimming of the XTAL

Address	Register Name	Register Write	Register Read	Default	Description
		XTAL_TRIM	XTAL_TRIM		
		(12) ENABLES_SEPARATE_PPA_CASC	(12) ENABLES_SEPARATE_PPA_CASC	0x0	PA cascode bit
		(11:6) ENABLES_EN_RXTX	(11:6) ENABLES_EN_RXTX	0x0	Enable signals
		(5:0) ENABLES_EN_BB	(5:0) ENABLES_EN_BB	0x0	Enable signals for the BB
0x400408BC	RF0_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
		(19) XTAL_CTRL_LOW_CLK_READY_TH_EN	(19) XTAL_CTRL_LOW_CLK_READY_TH_EN	0x0	clk_ready threshold
		(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	Clock selection for the digital block
		(16) XTAL_CTRL_XO_EN_B_REG	(16) XTAL_CTRL_XO_EN_B_REG	0x1	XTAL oscillator enable
		(15:14) XTAL_CTRL_XTAL_CKDIV	(15:14) XTAL_CTRL_XTAL_CKDIV	0x0	XTAL trimming speed

Address	Register Name	Register Write	Register Read	Default	Description
		(13) XTAL_CTRL_CLK_OUT_EN_B	(13) XTAL_CTRL_CLK_OUT_EN_B	0x0	Output clock to go to main IP
		(12) XTAL_CTRL_REG_VALUE_SEL	(12) XTAL_CTRL_REG_VALUE_SEL	0x0	Control bits of xtal_reg
		(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
		(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
		(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	Force output clocks on clk_pll, clk_dig and clk_out
		(4) XTAL_CTRL_CLK_DIG_EN_B	(4) XTAL_CTRL_CLK_DIG_EN_B	0x0	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	XTAL buffer disabling
		(2) XTAL_CTRL_HP_MODE	(2) XTAL_CTRL_HP_MODE	0x0	Bias current increase in the clock buffer
		(1) XTAL_CTRL_LP_MODE	(1) XTAL_CTRL_LP_MODE	0x0	Bias current decrease in the clock buffer
		(0) XTAL_CTRL_EXT_CLK_MODE	(0) XTAL_CTRL_EXT_CLK_MODE	0x0	Use XTAL pads as external clock input
0x400408C0	RF0_SUBBAND	(31:24) SUBBAND_OFFSET_SB_OFFSET_RX	(31:24) SUBBAND_OFFSET_SB_OFFSET_RX	0xF1	Offset to add in frequency count in order to compensate the offset of the varicap

Address	Register Name	Register Write	Register Read	Default	Description
		(23:16) SUBBAND_OFFSET_SB_OFFSET	(23:16) SUBBAND_OFFSET_SB_OFFSET	0xD0	Offset to add in frequency count in order to compensate the offset of the varicap
		(15:12) SWCAP_LIM_SB_MAX_VAL	(15:12) SWCAP_LIM_SB_MAX_VAL	0xF	Maximum subband value in linear search subband (freq and comp)
		(11:8) SWCAP_LIM_SB_MIN_VAL	(11:8) SWCAP_LIM_SB_MIN_VAL	0x0	Minimum subband value in linear search subband (freq and comp)
		(7) SUBBAND_CONF_SB_FLL_MODE	(7) SUBBAND_CONF_SB_FLL_MODE	0x1	FLL mode for the subband selection
		(6) SUBBAND_CONF_SB_INV_BAND	(6) SUBBAND_CONF_SB_INV_BAND	0x0	Invert the meaning of sb_high and sb_low
		(5:4) SUBBAND_CONF_SB_FREQ_CNT	(5:4) SUBBAND_CONF_SB_FREQ_CNT	0x0	The length to count in frequency mode
		(3:2) SUBBAND_CONF_SB_WAIT_T	(3:2) SUBBAND_CONF_SB_WAIT_T	0x0	Time to wait to the PLL to settle
		(1:0) SUBBAND_CONF_SB_MODE	(1:0) SUBBAND_CONF_SB_MODE	0x0	Sub-band algorithm mode
0x400408C4	RF0_REG31	(31:30) RSSI_DETECT_RSSI_DET_CR_LEN	(31:30) RSSI_DETECT_RSSI_DET_CR_LEN	0x0	Number of samples to estimate the carrier offset (banked)
		(29:28) RSSI_DETECT_RSSI_DET_WAIT	(29:28) RSSI_DETECT_RSSI_DET_WAIT	0x0	Symbols to wait after the RSSI detection (banked)
		(27:26) RSSI_DETECT_RSSI_DET_DIFF_LL	(27:26) RSSI_DETECT_RSSI_DET_DIFF_LL	0x0	Set the distance between the actual value and the subtracted one (banked)
		(25) RSSI_DETECT_EN_ABS_RSSI_DETECT	(25) RSSI_DETECT_EN_ABS_RSSI_DETECT	0x0	Absolute RSSI detection (banked)
		(24) RSSI_DETECT_	(24) RSSI_DETECT_	0x0	Differential RSSI detection (banked)

Address	Register Name	Register Write	Register Read	Default	Description
		EN_DIFF_RSSI_DETECT	EN_DIFF_RSSI_ DETECT		
		(23) SUBBAND_CORR_ SUBBAND_CORR_EN	(23) SUBBAND_CORR_ SUBBAND_CORR_EN	0x0	Subband correction
		(22:20) SUBBAND_ CORR_SUBBAND_CORR_ RX	(22:20) SUBBAND_ CORR_SUBBAND_CORR_ RX	0x0	Subband correction in Rx
		(18:16) SUBBAND_ CORR_SUBBAND_CORR_ TX	(18:16) SUBBAND_ CORR_SUBBAND_CORR_ TX	0x0	Subband correction in Tx
		(11) TXRX_CONF_INV_ CLK_PLL_TX	(11) TXRX_CONF_ INV_CLK_PLL_TX	0x0	Invert PLL clock when the radio is in Tx mode
		(10) TXRX_CONF_INV_ CLK_DIG_TX	(10) TXRX_CONF_ INV_CLK_DIG_TX	0x0	Invert digital clock when the radio is in Tx mode
		(9:8) TXRX_CONF_SB_ WAIT_T_TX	(9:8) TXRX_CONF_ SB_WAIT_T_TX	0x0	Xor value to apply to sb_wait_t (register SUBBAND_CONF) when the radio is in Tx mode
		(7) PA_RAMPUP_FULL_ PA_RAMPUP	(7) PA_RAMPUP_ FULL_PA_RAMPUP	0x1	PA rampup configuration
		(6:4) PA_RAMPUP_ DEL_PA_RAMPUP	(6:4) PA_RAMPUP_ DEL_PA_RAMPUP	0x4	Time to wait to start the ramp-up after the PA enable is detected
		(3:2) PA_RAMPUP_ TAU_PA_RAMPUP	(3:2) PA_RAMPUP_ TAU_PA_RAMPUP	0x0	Time constant of the ramp-up/ramp-down
		(1) PA_RAMPUP_EN_ PA_RAMPDOWN	(1) PA_RAMPUP_EN_ PA_RAMPDOWN	0x1	PA ramp-down
		(0) PA_RAMPUP_EN_ PA_RAMPUP	(0) PA_RAMPUP_EN_ PA_RAMPUP	0x1	PA ramp-up linearization
0x400408C8	RF0_DEMOD_CTRL	(31) SYNC_WORD_ CORR_EN_SYNC_WORD_	(31) SYNC_WORD_ CORR_EN_SYNC_WORD_	0x1	Sync word bias correction with RSSI detection (banked)

Address	Register Name	Register Write	Register Read	Default	Description
		CORR	CORR		
		(29:24) SYNC_WORD_CORR_SYNC_WORD_BIAS	(29:24) SYNC_WORD_CORR_SYNC_WORD_BIAS	0x8	Set the sync word bias (banked)
		(23:16) RSSI_DETECT_ABS_THR_RSSI_DET_ABS_THR	(23:16) RSSI_DETECT_ABS_THR_RSSI_DET_ABS_THR	0x0	Threshold used for absolute RSSI detection
		(15:8) RSSI_DETECT_DIFF_THR_RSSI_DET_DIFF_THR	(15:8) RSSI_DETECT_DIFF_THR_RSSI_DET_DIFF_THR	0x0	Threshold used for differential RSSI detection
		(7) DEMOD_CTRL_DL_SYNC_NO_DATA	(7) DEMOD_CTRL_DL_SYNC_NO_DATA	0x1	No data going through the demodulator, until the delay line detects the sync word (banked)
		(6) DEMOD_CTRL_EN_DELLINE_SYNC_DET	(6) DEMOD_CTRL_EN_DELLINE_SYNC_DET	0x1	Sync word detection in the delay line (banked)
		(5) DEMOD_CTRL_RSSI_DET_FILT	(5) DEMOD_CTRL_RSSI_DET_FILT	0x0	Additional filtering on the RSSI value (banked)
		(4) DEMOD_CTRL_EN_FAST_CLK_RECOV	(4) DEMOD_CTRL_EN_FAST_CLK_RECOV	0x0	Clock recovery during the resto of the preamble (banked)
		(3) DEMOD_CTRL_EN_MIN_MAX_MF	(3) DEMOD_CTRL_EN_MIN_MAX_MF	0x0	Min max algo after the matched filter (banked)
		(2) DEMOD_CTRL_EN_PRE_SYNC	(2) DEMOD_CTRL_EN_PRE_SYNC	0x0	Sync detection on the non-delayed path (banked)
		(1) DEMOD_CTRL_BLOCK_RSSI_DET	(1) DEMOD_CTRL_BLOCK_RSSI_DET	0x0	RSSI detection during the slow-down period (banked)
		(0) DEMOD_CTRL_EARLY_FINE_RECOV	(0) DEMOD_CTRL_EARLY_FINE_RECOV	0x0	Early fine recovery after the packet detection or pre-sync (banked)
0x400408CC	RF0_REG33	(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

Address	Register Name	Register Write	Register Read	Default	Description
		(23:16) SPARES_SPARES	(23:16) SPARES_SPARES	0x0	Spare bits
		(14) PADS_PE_DS_GPIO_DS	(14) PADS_PE_DS_GPIO_DS	0x0	Increased drive strength of the digital pads
		(13) PADS_PE_DS_GPIO_PE	(13) PADS_PE_DS_GPIO_PE	0x0	Pull-up of the GPIO pads
		(12) PADS_PE_DS_NRESET_PE	(12) PADS_PE_DS_NRESET_PE	0x0	Pull-up of the NRESET pads
		(11) PADS_PE_DS_SPI_MISO_PE	(11) PADS_PE_DS_SPI_MISO_PE	0x0	Pull-up of the SPI MISO pads
		(10) PADS_PE_DS_SPI_MOSI_PE	(10) PADS_PE_DS_SPI_MOSI_PE	0x0	Pull-up of the SPI MOSI pads
		(9) PADS_PE_DS_SPI_SCLK_PE	(9) PADS_PE_DS_SPI_SCLK_PE	0x0	Pull-up of the SPI CLK pads
		(8) PADS_PE_DS_SPI_CS_N_PE	(8) PADS_PE_DS_SPI_CS_N_PE	0x0	Pull-up of the SPI CSN pads
		(7:6) SUBBAND_FLL_SB_FLL_DITHER	(7:6) SUBBAND_FLL_SB_FLL_DITHER	0x0	Select the dithering
		(5:4) SUBBAND_FLL_SB_FLL_CIC_TAU	(5:4) SUBBAND_FLL_SB_FLL_CIC_TAU	0x3	Set the CIC decimator factor
		(3) SUBBAND_FLL_SB_FLL_PH_4_N8	(3) SUBBAND_FLL_SB_FLL_PH_4_N8	0x0	Phases in the frequency detector
		(2:0) SUBBAND_FLL_SB_FLL_WAIT	(2:0) SUBBAND_FLL_SB_FLL_WAIT	0x3	Set the number of CIC samples before stopping the FLL
0x400408D0	RF0_REG34	(29:24) CLK_RECOVERY_CLK_RECOV_CORR	(29:24) CLK_RECOVERY_CLK_RECOV_CORR	0x4	Number of samples that covers the clock recovery correlator
		(23:16) CLK_	(23:16) CLK_	0x80	Time constant for switch the clock

Address	Register Name	Register Write	Register Read	Default	Description
		RECOVERY_CLK_AB_ LIMIT	RECOVERY_CLK_AB_ LIMIT		phase if chosen wrong in clk recovery algorithm
		(15) TX_PRE_DIST_ EN_PRE_DIST	(15) TX_PRE_DIST_ EN_PRE_DIST	0x1	Tx pre-distortion filter (banked)
		(13:8) TX_PRE_DIST_ PRE_DIST_B0	(13:8) TX_PRE_ DIST_PRE_DIST_B0	0x2E	Coefficient b0 of the Tx pre-distortion filter (banked)
		(5:0) TX_PRE_DIST_ PRE_DIST_A0	(5:0) TX_PRE_DIST_ PRE_DIST_A0	0x2F	Coefficient a0 of the Tx pre-distortion filter (banked)
0x400408D4	RF0_BLE_LR	(30:24) BLR_SYNC_ THRESHOLD_BLE_SYNC_ THR	(30:24) BLR_SYNC_ THRESHOLD_BLE_ SYNC_THR	0x38	Threshold for the BLR sync word detector
		(19:16) BLR_ PREAMBLE_BLE_PRE_ THR	(19:16) BLR_ PREAMBLE_BLE_PRE_ THR	0x1	Threshold for the BLR preamble detector
		(15) BLE_LONG_ RANGE_BLR_PUT_RI_ FIFO	(15) BLE_LONG_ RANGE_BLR_PUT_RI_ FIFO	0x1	During the reception the RI (rate indicator) is put into the Rx FIFO (banked)
		(14) BLE_LONG_ RANGE_BLR500_NO_ ROUGH	(14) BLE_LONG_ RANGE_BLR500_NO_ ROUGH	0x1	Rough recovery is stopped during the 500kbps payloads of BLR packets (banked)
		(13) BLE_LONG_ RANGE_BLR_LIN_ FILTER	(13) BLE_LONG_ RANGE_BLR_LIN_ FILTER	0x1	Matched filter (banked)
		(12) BLE_LONG_ RANGE_EN_BLR_FLUSH	(12) BLE_LONG_ RANGE_EN_BLR_FLUSH	0x1	Viterbi path 0 flushing at the end of the packet (banked)
		(11) BLE_LONG_ RANGE_BLR_USE_EXT_ LEN	(11) BLE_LONG_ RANGE_BLR_USE_EXT_ LEN	0x0	BLR_PKT_LEN for flushing out the Viterbi (banked)
		(10) BLE_LONG_	(10) BLE_LONG_	0x0	Long Range feature in Tx mode

Address	Register Name	Register Write	Register Read	Default	Description
		RANGE_DISABLE_BLR_TX	RANGE_DISABLE_BLR_TX		(banked)
		(9) BLE_LONG_RANGE_BLR_500_N125	(9) BLE_LONG_RANGE_BLR_500_N125	0x0	Data rate selection (banked)
		(8) BLE_LONG_RANGE_EN_BLR	(8) BLE_LONG_RANGE_EN_BLR	0x0	BLE long range mode (banked)
		(4) HW_TRIGGER_HW_TRIG_GPIO	(4) HW_TRIGGER_HW_TRIG_GPIO	0x0	HW trigger is mapped on the GPIO instead of the Tx_on signal 0x0
		(3) HW_TRIGGER_HW_TRIG_SUBBAND	(3) HW_TRIGGER_HW_TRIG_SUBBAND	0x0	Activate the sub-band selection during the Tx activation
		(2) HW_TRIGGER_HW_TRIG_TX_NRX	(2) HW_TRIGGER_HW_TRIG_TX_NRX	0x0	Activate the Tx mode
		(1) HW_TRIGGER_HW_TRIG_LOW	(1) HW_TRIGGER_HW_TRIG_LOW	0x0	Set the trigger polarity
		(0) HW_TRIGGER_HW_TRIG_ACTIVE	(0) HW_TRIGGER_HW_TRIG_ACTIVE	0x0	Enable HW trigger
0x400408D8	RF0_REG36	(30:28) IQ_SPARES_EN_BIAS_SPARE	(30:28) IQ_SPARES_EN_BIAS_SPARE	0x0	Enable for IQ spares
		(27:24) IQ_SPARES_IQ_SPARE_2	(27:24) IQ_SPARES_IQ_SPARE_2	0x0	Spare bias 2
		(23:20) IQ_SPARES_IQ_SPARE_1	(23:20) IQ_SPARES_IQ_SPARE_1	0x0	Spare bias 1
		(19:16) IQ_SPARES_IQ_SPARE_0	(19:16) IQ_SPARES_IQ_SPARE_0	0x0	Spare bias 0
		(8) MISC_ISO_VDDA	(8) MISC_ISO_VDDA	0x0	Isolate VDDA signals
		(5) BLR_DEMAPPER_BLR_SEND_DECODED_RI	(5) BLR_DEMAPPER_BLR_SEND_DECODED_RI	0x0	Fully decode the rate indicator

Address	Register Name	Register Write	Register Read	Default	Description
		(4) BLR_DEMAPPER_BLR_USE_EXT_VIT_GFSK	(4) BLR_DEMAPPER_BLR_USE_EXT_VIT_GFSK	0x1	500kbps BLR uses the Viterbi GFSK decision
		(3:2) BLR_DEMAPPER_BLR_500_DPHASE	(3:2) BLR_DEMAPPER_BLR_500_DPHASE	0x3	Set the distance between samples for the phase to frequency conversion in S2 mode
		(1) BLR_DEMAPPER_BLR_500_LOW_GAIN	(1) BLR_DEMAPPER_BLR_500_LOW_GAIN	0x0	Set the low gain in S2 mode
		(0) BLR_DEMAPPER_BLR_125_LOW_GAIN	(0) BLR_DEMAPPER_BLR_125_LOW_GAIN	0x0	Set the low gain in S8 mode
0x400408DC	RF0_PROT_TIMER	(31) PROT_TIMER_CONF_EN_PROT_TIMER	(31) PROT_TIMER_CONF_EN_PROT_TIMER	0x0	Enable the protocol timer
		(29:27) PROT_TIMER_CONF_PT_T_STP_1	(29:27) PROT_TIMER_CONF_PT_T_STP_1	0x0	Configure the time stamp 1
		(26:24) PROT_TIMER_CONF_PT_T_STP_0	(26:24) PROT_TIMER_CONF_PT_T_STP_0	0x0	Configure the time stamp 0
		(22) STAGING_PS_NZ_START_BIT	(22) STAGING_PS_NZ_START_BIT	0x0	Select the frequency offset
		(21) STAGING_PS_NZ_START	(21) STAGING_PS_NZ_START	0x0	Start the pulse shaper with a +/- 250 kHz frequency offset
		(20) STAGING_DEL_PA_RAMPDW	(20) STAGING_DEL_PA_RAMPDW	0x0	Delay the PA ramp-down by 4.5 us
		(19) STAGING_PEAK_DET_TH_SHIFT	(19) STAGING_PEAK_DET_TH_SHIFT	0x0	Peak detector threshold shift
		(18:17) STAGING_AGC_DERIV_LVL	(18:17) STAGING_AGC_DERIV_LVL	0x2	Select the AGC derivative level
		(16) STAGING_AGC_	(16) STAGING_AGC_	0x0	AGC algorithm uses the derivative

Address	Register Name	Register Write	Register Read	Default	Description
		USE_DERIV	USE_DERIV		information to accelerate the AGC settling
		(15:8) BLE_DTM_BLE_DTM_LEN	(15:8) BLE_DTM_BLE_DTM_LEN	0x25	Set the BLE DTM packet length
		(7) BLE_DTM_EN_BLE_DTM	(7) BLE_DTM_EN_BLE_DTM	0x0	Enable the BLE DTM automatic packets
		(3:0) BLE_DTM_BLE_DTM_PKT_TYPE	(3:0) BLE_DTM_BLE_DTM_PKT_TYPE	0x0	Set the BLE DTM packet type (see Bluetooth specification)
0x400408E0	RF0_CTE_OPTS	(29) CTE_OPTS_RECT_PS_CTE	(29) CTE_OPTS_RECT_PS_CTE	0x0	Use rectangular pulse shape during the CTE
		(28) CTE_OPTS_USE_CTE_WO_CP	(28) CTE_OPTS_USE_CTE_WO_CP	0x0	Enable the CTE without reading or inserting the CP
		(27) CTE_OPTS_CTE_AMPL	(27) CTE_OPTS_CTE_AMPL	0x0	Enable the usage of the RSSI values to adapt the amplitude of the IQ signal based to the RSSI value
		(26) CTE_OPTS_DF_AOA_SLOT_TIME	(26) CTE_OPTS_DF_AOA_SLOT_TIME	0x0	Indicate the switching/sampling slot period for AoA
		(25) CTE_OPTS_CP_INSERT	(25) CTE_OPTS_CP_INSERT	0x0	Force the CP bit in the packet header to 1
		(24) CTE_OPTS_EN_READ_CP	(24) CTE_OPTS_EN_READ_CP	0x0	CP bit is read in the packet header (BLE standard)
		(23:16) CTE_OPTS_CTE_INFO	(23:16) CTE_OPTS_CTE_INFO	0x0	Set the CTEInfo field in the packet header while cp_insert is set to 1
		(14:10) ASK_MOD_ASK_MAX	(14:10) ASK_MOD_ASK_MAX	0xC	Set the maximum value for the ASK modulation
		(9:5) ASK_MOD_ASK_MIN	(9:5) ASK_MOD_ASK_MIN	0x0	Set the minimum value for the ASK modulation
		(4:1) ASK_MOD_ASK_	(4:1) ASK_MOD_ASK_	0x7	Set the how long to count for the ASK

Address	Register Name	Register Write	Register Read	Default	Description
		CNT	CNT		modulation
		(0) ASK_MOD_EN_ RSSI_ASK	(0) ASK_MOD_EN_ RSSI_ASK	0x0	PA will perform an ASK modulation
0x400408E4	RF0_PT_DELTA_0	(31:30) PT_DELTA_ TS_0_PT_DELTA_T0_ MULT	(31:30) PT_DELTA_ TS_0_PT_DELTA_T0_ MULT	0x0	Multiplier for the delta t0
		(19:0) PT_DELTA_TS_ 0_PT_DELTA_T0	(19:0) PT_DELTA_ TS_0_PT_DELTA_T0	0x0	Delta t0 for the protocol timer
0x400408E8	RF0_PT_DELTA_1	(31:30) PT_DELTA_ TS_1_PT_DELTA_T1_ MULT	(31:30) PT_DELTA_ TS_1_PT_DELTA_T1_ MULT	0x0	Multiplier for the delta t1
		(19:0) PT_DELTA_TS_ 1_PT_DELTA_T1	(19:0) PT_DELTA_ TS_1_PT_DELTA_T1	0x0	Delta t1 for the protocol timer
0x400408EC	RF0_CTE_IF	(25:16) CTE_CTRL_ DELAY_TX_DF_DELAY_ TX	(25:16) CTE_CTRL_ DELAY_TX_DF_DELAY_ TX	0x0	Delay (in 62.5ns) form the serializer up to the antenna in direction finding (banked)
		(15) ANTENNA_CONF_ DF_IND_PATTERN	(15) ANTENNA_CONF_ DF_IND_PATTERN	0x0	Separate the antenna switching pattern from the reference one
		(14) ANTENNA_CONF_ DF_IND_ANTENNA	(14) ANTENNA_CONF_ DF_IND_ANTENNA	0x0	Make the antenna for DF independent from the rest of the packet
		(13:8) ANTENNA_ CONF_ANT_LUT_M	(13:8) ANTENNA_ CONF_ANT_LUT_M	0x0	Number of states used (-1) in the antenna LUT
		(5) CTE_AUTO_PULL_ EXT_IQ_SMP_TYPE	(5) CTE_AUTO_PULL_ EXT_IQ_SMP_TYPE	0x0	Select the external IQ sample signal qualifier type
		(4) CTE_AUTO_PULL_ IQ_MSB	(4) CTE_AUTO_PULL_ IQ_MSB	0x0	Select which signal is sent over the MSB in case of a 16bits buffers
		(3:2) CTE_AUTO_ PULL_IQ_DATA_BUS_ SIZE	(3:2) CTE_AUTO_ PULL_IQ_DATA_BUS_ SIZE	0x0	Select the bus data size of IQ signals

Address	Register Name	Register Write	Register Read	Default	Description
		(1) CTE_AUTO_PULL_CTE_QUAL	(1) CTE_AUTO_PULL_CTE_QUAL	0x0	Select the CTE data qualifier
		(0) CTE_AUTO_PULL_EN_CTE_AUTO_PULL	(0) CTE_AUTO_PULL_EN_CTE_AUTO_PULL	0x0	Enable the automatic push of CTE data to an external IP
0x400408F0	RF0_CTE_CTRL	(25:16) CTE_CTRL_DELAY_RX_DF_DELAY_SWITCH_RX	(25:16) CTE_CTRL_DELAY_RX_DF_DELAY_SWITCH_RX	0x0	Delay (in 62.5ns) from the antenna up to the deserializer in direction finding (banked)
		(9:0) CTE_CTRL_DELAY_RX_DF_DELAY_SAMPLE_RX	(9:0) CTE_CTRL_DELAY_RX_DF_DELAY_SAMPLE_RX	0x0	Delay (in 62.5ns) from the matched filter up to the deserializer in direction finding (banked)
0x400408F4	RF0_AGC_ADVANCED	(26:16) AGC_SWITCHES_AGC_SHORTS_LUT	(26:16) AGC_SWITCHES_AGC_SHORTS_LUT	0x0	Array of values that indicates if the highpass shorts must be set for the AGC state passage from n -> n+1
		(8) DEBUG_FAKE_IQ_SAMPLES	(8) DEBUG_FAKE_IQ_SAMPLES	0x0	Generate fake IQ samples
		(7:4) AGC_ADVANCED_AGC_TAU_SHORTS	(7:4) AGC_ADVANCED_AGC_TAU_SHORTS	0x0	Time constant that indicates the time that shorts must be on
		(3) AGC_ADVANCED_AGC_EN_SHORT_PHADC	(3) AGC_ADVANCED_AGC_EN_SHORT_PHADC	0x0	Enable the short on the phase ADC highpass filter
		(2) AGC_ADVANCED_AGC_EN_SHORT_IFA	(2) AGC_ADVANCED_AGC_EN_SHORT_IFA	0x0	Enable the short on the IFA highpass filter
		(1) AGC_ADVANCED_AGC_USE_SHORTS	(1) AGC_ADVANCED_AGC_USE_SHORTS	0x0	Enable the usage of the shorts located in the BB path
		(0) AGC_ADVANCED_AGC_FULL_SPEED	(0) AGC_ADVANCED_AGC_FULL_SPEED	0x0	Enable the maximum speed in AGC
0x400408F8	RF0_DATA_STREAMING	(9) DATA_STREAMING_DMA_PHASE_TYPE	(9) DATA_STREAMING_DMA_PHASE_TYPE	0x0	Use the phase after the rescaler instead of the raw phase from phase ADC (banked)

Address	Register Name	Register Write	Register Read	Default	Description
		(8) DATA_STREAMING_DMA_EN_BUS	(8) DATA_STREAMING_DMA_EN_BUS	0x0	Enable the DMA bus on the IP interface (banked)
		(6) DATA_STREAMING_PERIODIC_SAMPLE_AFTER_CTE	(6) DATA_STREAMING_PERIODIC_SAMPLE_AFTER_CTE	0x1	Restart sampling after the CTE period (banked)
		(5) DATA_STREAMING_PERIODIC_SAMPLE_AT_SYNC	(5) DATA_STREAMING_PERIODIC_SAMPLE_AT_SYNC	0x0	Start sampling at the sync detection signal from the delay line (banked)
		(4) DATA_STREAMING_PERIODIC_SAMPLE_OSR_CLK	(4) DATA_STREAMING_PERIODIC_SAMPLE_OSR_CLK	0x0	Oversample (8x) the reference clock of the periodic sample (banked)
		(3:1) DATA_STREAMING_PERIODIC_SAMPLE_OSR	(3:1) DATA_STREAMING_PERIODIC_SAMPLE_OSR	0x3	Division factor (-1) of for the sampling period of the periodic IQ sampling (banked)
		(0) DATA_STREAMING_PERIODIC_SAMPLE_EN_IQ	(0) DATA_STREAMING_PERIODIC_SAMPLE_EN_IQ	0x0	Sample periodically I and Q channels after the matched filter and put into the IQ FIFO (banked)
0x400408FC	RF0_REVISION	–	(31:24) CHIP_ID	0x30	Remapped register of CHIP_ID
0x40040900	RF0_FSM_CTRL	–	(31:30) RXFIFO_STATUS_RX_BIST_ERRORS	0x0	Rx FIFO BIST result
		(31:25) RXFIFO_STATUS_RX_BIST	–	N/A	Start the bist test on the Rx FIFO (code 0x5d)
		–	(29) RXFIFO_	0x0	Rx FIFO near underflow

Address	Register Name	Register Write	Register Read	Default	Description
			STATUS_RX_NEAR_ UNDERFLOW		
		–	(28) RXFIFO_ STATUS_RX_NEAR_ OVERFLOW	0x0	Rx FIFO near overflow
		–	(27) RXFIFO_ STATUS_RX_ UNDERFLOW	0x0	Rx FIFO underflow
		–	(26) RXFIFO_ STATUS_RX_OVERFLOW	0x0	Rx FIFO overflow
		–	(25) RXFIFO_ STATUS_RX_FULL	0x0	Rx FIFO full
		–	(24) RXFIFO_ STATUS_RX_EMPTY	0x0	Rx FIFO empty
		(24) RXFIFO_STATUS_ RX_FLUSH	–	N/A	Rx FIFO flush
		–	(23:22) TXFIFO_ STATUS_TX_BIST_ ERRORS	0x0	Tx FIFO BIST result
		(23:17) TXFIFO_ STATUS_TX_BIST	–	N/A	Start the bist test on the Tx FIFO (code 0x5d)
		–	(21) TXFIFO_ STATUS_TX_NEAR_ UNDERFLOW	0x0	Tx FIFO near underflow
		–	(20) TXFIFO_ STATUS_TX_NEAR_ OVERFLOW	0x0	Tx FIFO near overflow
		–	(19) TXFIFO_ STATUS_TX_	0x0	Tx FIFO underflow

Address	Register Name	Register Write	Register Read	Default	Description
			UNDERFLOW		
		–	(18) TXFIFO_ STATUS_TX_OVERFLOW	0x0	Tx FIFO overflow
		–	(17) TXFIFO_ STATUS_TX_FULL	0x0	Tx FIFO full
		–	(16) TXFIFO_ STATUS_TX_EMPTY	0x0	Tx FIFO empty
		(16) TXFIFO_STATUS_ TX_FLUSH	–	N/A	Tx FIFO flush
		–	(10) FSM_STATUS_ TX_NRX	0x0	Select Rx or Tx mode
		–	(9:8) FSM_STATUS_ STATUS	0x0	Status of the FSM
		(3) FSM_MODE_RESET	–	N/A	FSM reset
		–	(2) FSM_MODE_RX_ MODE	0x0	Rx status
		(2) FSM_MODE_TX_NRX	–	N/A	Set the radio in Tx or Rx mode
		–	(1) FSM_MODE_TX_ MODE	0x0	Tx status
		(1:0) FSM_MODE_MODE	–	N/A	Set the FSM mode
		–	(0) FSM_MODE_N_ IDLE	0x0	FSM status
0x40040904	RF0_IQFIFO_STATUS	–	(24:16) TXFIFO_ COUNT_TX_COUNT	0x0	Number of bytes in the Tx FIFO
		–	(15:8) IQFIFO_ COUNT_IQ_COUNT	0x0	Number of bytes in the IQ FIFO
		–	(7:6) IQFIFO_	0x0	IQ FIFO BIST result

Address	Register Name	Register Write	Register Read	Default	Description
			STATUS_IQ_BIST_ ERRORS		
		(7:1) IQFIFO_ STATUS_IQ_BIST	–	N/A	Start the BIST test on the IQ FIFO (code 0x5d)
		–	(5) IQFIFO_STATUS_ IQ_NEAR_UNDERFLOW	0x0	IQ FIFO near underflow
		–	(4) IQFIFO_STATUS_ IQ_NEAR_OVERFLOW	0x0	IQ FIFO near overflow
		–	(3) IQFIFO_STATUS_ IQ_UNDERFLOW	0x0	IQ FIFO underflow
		–	(2) IQFIFO_STATUS_ IQ_OVERFLOW	0x0	IQ FIFO overflow
		–	(1) IQFIFO_STATUS_ IQ_FULL	0x0	IQ FIFO full
		–	(0) IQFIFO_STATUS_ IQ_EMPTY	0x0	IQ FIFO empty
		(0) IQFIFO_STATUS_ FLUSH	–	N/A	IQ FIFO flush
0x40040908	RF0_TXFIFO	(7:0) TXFIFO_TX_ DATA	–	N/A	Data to be sent
0x4004090C	RF0_RXFIFO	–	(7:0) RXFIFO_RX_ DATA	0x0	Received data
0x40040910	RF0_IQFIFO	–	(7:0) IQFIFO_IQ_ DATA	0x0	IQ data for AoA or AoD
0x40040914	RF0_REG45	–	(25:16) RSSI_AVG_ RSSI_AVG	0x0	Filtered RSSI value
		–	(8:0) RXFIFO_ COUNT_RX_COUNT	0x0	Number of bytes in the Rx FIFO

Address	Register Name	Register Write	Register Read	Default	Description
0x40040918	RF0_DESER_STATUS	–	(7) DESER_STATUS_SIGNAL_RECEIVING	0x0	Deserializer enabling
		–	(6) DESER_STATUS_SYNC_DETECTED	0x0	Sync word detection
		–	(5) DESER_STATUS_WAIT_SYNC	0x0	Deserializer waiting for the sync word
		–	(4) DESER_STATUS_IS_ADDRESS_BR	0x0	Received address
		–	(3) DESER_STATUS_PKT_LEN_ERR	0x0	Packet length
		–	(2) DESER_STATUS_ADDRESS_ERR	0x0	Address error
		–	(1) DESER_STATUS_CRC_ERR	0x0	CRC error
		–	(0) DESER_STATUS_DESER_FINISH	0x0	Deserializer status
0x4004091C	RF0_BLE_AEC_CCM	–	(2) BLE_AES_CCM_BLE_AES_MIC_OK	0x0	AES CCM MIC error
		–	(1) BLE_AES_CCM_BLE_AES_DONE_RX	0x0	AES CCM packet decoding
		–	(0) BLE_AES_CCM_BLE_AES_DONE_TX	0x0	AES CCM packet encoding
0x40040920	RF0_IRQ_STATUS	–	(5) IRQ_STATUS_FLAG_RXFIFO	0x0	IRQ RXFIFO status
		–	(4) IRQ_STATUS_FLAG_TXFIFO	0x0	IRQ TXFIFO status
		–	(3) IRQ_STATUS_FLAG_SYNC	0x0	IRQ SYNC status

Address	Register Name	Register Write	Register Read	Default	Description
		–	(2) IRQ_STATUS_ FLAG_RECEIVED	0x0	IRQ RECEIVED status
		–	(1) IRQ_STATUS_ FLAG_RXSTOP	0x0	IRQ RXSTOP status
		–	(0) IRQ_STATUS_ FLAG_TX	0x0	IRQ Tx status
0x40040924	RF0_RSSI_MIN_MAX	–	(25:16) RSSI_MAX_ RSSI_MAX	0x0	Maximum RSSI value over a filtering period
		–	(9:0) RSSI_MIN_ RSSI_MIN	0x0	Minimum RSSI value over a filtering period
0x40040928	RF0_REG4A	–	(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) DR_ERR_ IND_DR_ERR_IND	0x0	Data-rate error indicator
		–	(9:0) RSSI_PKT_ RSSI_PKT	0x0	Filtered RSSI value sampled during the packet reception
0x4004092C	RF0_FEI	–	(31:16) FEI_PKT_ FEI_PKT	0x0	Frequency error indicator sampled during the packet reception
		–	(15:0) FEI_FEI_OUT	0x0	Frequency error indicator
0x40040930	RF0_REG4C	–	(31:24) LINK_QUAL_ PKT_LINK_QUALITY_ PKT	0x0	Link quality indicator sampled during the packet reception
		–	(23:16) LINK_QUAL_ LINK_QUALITY	0x0	Instantaneous link quality indicator
		–	(15:0) FEI_AFC_	0x0	Frequency error indicator sampled

Address	Register Name	Register Write	Register Read	Default	Description
			FEI_AFC		during the AFC
0x40040934	RF0_ANALOG_INFO	(25:24) BLR_ READOUT_BLR_RATE	–	N/A	Bluetooth LE long range rate indicator
		–	(22:20) PEAK_DET_ VAL_PEAK_DET_FILT	0x0	Distance from the subband center (only available with the FLL method)
		–	(18:16) PEAK_DET_ VAL_PEAK_DET_RAW	0x0	Distance from the subband center (only available with the FLL method)
		–	(15) ANALOG_INFO_ POR_VDDA	0x0	VDDA LDO disable status
		–	(14) ANALOG_INFO_ PLL_UNLOCK	0x0	PLL unlock status
		–	(13) ANALOG_INFO_ XTAL_FINISH	0x0	XTAL algorithm status
		–	(12) ANALOG_INFO_ DLL_LOCKED	0x0	DLL lock status
		–	(11) ANALOG_INFO_ CLK_DIG_READY	0x0	Ready signal of the digital clock
		–	(10) ANALOG_INFO_ CLK_PLL_READY	0x0	PLL clock status
		–	(9:8) ANALOG_INFO_ SUBBAND	0x0	Status of the subband comparator Hi
		–	(7:0) SUBBAND_ERR_ SB_FLL_ERR	0x0	Distance from the subband center (only available with the FLL method)
0x40040938	RF0_SAMPLE_RSSI	(0) SAMPLE_RSSI	–	N/A	Sample the thermometric RSSI
0x4004093C	RF0_RSSI_THERM	–	(29:0) RSSI_THERM	0x0	Thermometric value of the RSSI
0x40040980	RF0_LUT_ANTENNA_ARRAY_ 1	(31:28) LUT_ ANTENNA_ARRAY_1_ ANTENNA_7	(31:28) LUT_ ANTENNA_ARRAY_1_ ANTENNA_7	0x0	Antenna 7 specification

Address	Register Name	Register Write	Register Read	Default	Description
		(27:24) LUT_ ANTENNA_ARRAY_1_ ANTENNA_6	(27:24) LUT_ ANTENNA_ARRAY_1_ ANTENNA_6	0x0	Antenna 6 specification
		(23:20) LUT_ ANTENNA_ARRAY_1_ ANTENNA_5	(23:20) LUT_ ANTENNA_ARRAY_1_ ANTENNA_5	0x0	Antenna 5 specification
		(19:16) LUT_ ANTENNA_ARRAY_1_ ANTENNA_4	(19:16) LUT_ ANTENNA_ARRAY_1_ ANTENNA_4	0x0	Antenna 4 specification
		(15:12) LUT_ ANTENNA_ARRAY_1_ ANTENNA_3	(15:12) LUT_ ANTENNA_ARRAY_1_ ANTENNA_3	0x3	Antenna 3 specification
		(11:8) LUT_ANTENNA_ ARRAY_1_ANTENNA_2	(11:8) LUT_ ANTENNA_ARRAY_1_ ANTENNA_2	0x2	Antenna 2 specification
		(7:4) LUT_ANTENNA_ ARRAY_1_ANTENNA_1	(7:4) LUT_ANTENNA_ ARRAY_1_ANTENNA_1	0x1	Antenna 1 specification
		(3:0) LUT_ANTENNA_ ARRAY_1_ANTENNA_0	(3:0) LUT_ANTENNA_ ARRAY_1_ANTENNA_0	0x0	Antenna 0 specification
0x40040984	RF0_LUT_ANTENNA_ARRAY_ 2	(31:28) LUT_ ANTENNA_ARRAY_2_ ANTENNA_15	(31:28) LUT_ ANTENNA_ARRAY_2_ ANTENNA_15	0x0	Antenna 15 specification
		(27:24) LUT_ ANTENNA_ARRAY_2_ ANTENNA_14	(27:24) LUT_ ANTENNA_ARRAY_2_ ANTENNA_14	0x0	Antenna 14 specification
		(23:20) LUT_ ANTENNA_ARRAY_2_ ANTENNA_13	(23:20) LUT_ ANTENNA_ARRAY_2_ ANTENNA_13	0x0	Antenna 13 specification
		(19:16) LUT_ ANTENNA_ARRAY_2_	(19:16) LUT_ ANTENNA_ARRAY_2_	0x0	Antenna 12 specification

Address	Register Name	Register Write	Register Read	Default	Description
		ANTENNA_12	ANTENNA_12		
		(15:12) LUT_ ANTENNA_ARRAY_2_ ANTENNA_11	(15:12) LUT_ ANTENNA_ARRAY_2_ ANTENNA_11	0x0	Antenna 11 specification
		(11:8) LUT_ANTENNA_ ARRAY_2_ANTENNA_10	(11:8) LUT_ ANTENNA_ARRAY_2_ ANTENNA_10	0x0	Antenna 10 specification
		(7:4) LUT_ANTENNA_ ARRAY_2_ANTENNA_9	(7:4) LUT_ANTENNA_ ARRAY_2_ANTENNA_9	0x0	Antenna 9 specification
		(3:0) LUT_ANTENNA_ ARRAY_2_ANTENNA_8	(3:0) LUT_ANTENNA_ ARRAY_2_ANTENNA_8	0x0	Antenna 8 specification
0x40040988	RF0_LUT_ANTENNA_ARRAY_ 3	(31:28) LUT_ ANTENNA_ARRAY_3_ ANTENNA_23	(31:28) LUT_ ANTENNA_ARRAY_3_ ANTENNA_23	0x0	Antenna 23 specification
		(27:24) LUT_ ANTENNA_ARRAY_3_ ANTENNA_22	(27:24) LUT_ ANTENNA_ARRAY_3_ ANTENNA_22	0x0	Antenna 22 specification
		(23:20) LUT_ ANTENNA_ARRAY_3_ ANTENNA_21	(23:20) LUT_ ANTENNA_ARRAY_3_ ANTENNA_21	0x0	Antenna 21 specification
		(19:16) LUT_ ANTENNA_ARRAY_3_ ANTENNA_20	(19:16) LUT_ ANTENNA_ARRAY_3_ ANTENNA_20	0x0	Antenna 20 specification
		(15:12) LUT_ ANTENNA_ARRAY_3_ ANTENNA_19	(15:12) LUT_ ANTENNA_ARRAY_3_ ANTENNA_19	0x0	Antenna 19 specification
		(11:8) LUT_ANTENNA_ ARRAY_3_ANTENNA_18	(11:8) LUT_ ANTENNA_ARRAY_3_ ANTENNA_18	0x0	Antenna 18 specification

Address	Register Name	Register Write	Register Read	Default	Description
		(7:4) LUT_ANTENNA_ARRAY_3_ANTENNA_17	(7:4) LUT_ANTENNA_ARRAY_3_ANTENNA_17	0x0	Antenna 17 specification
		(3:0) LUT_ANTENNA_ARRAY_3_ANTENNA_16	(3:0) LUT_ANTENNA_ARRAY_3_ANTENNA_16	0x0	Antenna 16 specification
0x4004098C	RF0_LUT_ANTENNA_ARRAY_4	(31:28) LUT_ANTENNA_ARRAY_4_ANTENNA_31	(31:28) LUT_ANTENNA_ARRAY_4_ANTENNA_31	0x0	Antenna 31 specification
		(27:24) LUT_ANTENNA_ARRAY_4_ANTENNA_30	(27:24) LUT_ANTENNA_ARRAY_4_ANTENNA_30	0x0	Antenna 30 specification
		(23:20) LUT_ANTENNA_ARRAY_4_ANTENNA_29	(23:20) LUT_ANTENNA_ARRAY_4_ANTENNA_29	0x0	Antenna 29 specification
		(19:16) LUT_ANTENNA_ARRAY_4_ANTENNA_28	(19:16) LUT_ANTENNA_ARRAY_4_ANTENNA_28	0x0	Antenna 28 specification
		(15:12) LUT_ANTENNA_ARRAY_4_ANTENNA_27	(15:12) LUT_ANTENNA_ARRAY_4_ANTENNA_27	0x0	Antenna 27 specification
		(11:8) LUT_ANTENNA_ARRAY_4_ANTENNA_26	(11:8) LUT_ANTENNA_ARRAY_4_ANTENNA_26	0x0	Antenna 26 specification
		(7:4) LUT_ANTENNA_ARRAY_4_ANTENNA_25	(7:4) LUT_ANTENNA_ARRAY_4_ANTENNA_25	0x0	Antenna 25 specification
		(3:0) LUT_ANTENNA_ARRAY_4_ANTENNA_24	(3:0) LUT_ANTENNA_ARRAY_4_ANTENNA_24	0x0	Antenna 24 specification
0x400409C0	RF0_REG50	–	(27) FEATURES_HAS_BLE_AES	0x0	Bluetooth AES block availability
		–	(26) FEATURES_HAS_	0x1	Bluetooth Direction Finding AoA/AoD

Address	Register Name	Register Write	Register Read	Default	Description
			BLE_DF_AOA_AOD		feature availability
		–	(25) FEATURES_HAS_BLE_LONG_RANGE	0x1	Bluetooth long range feature availability
		–	(24) FEATURES_FEATURES_AVAILABLE	0x1	Features availability
		(23:16) BLR_PKT_LEN_BLR_PKT_LEN	–	N/A	Packet length of the BLR packet
		(15:8) PROT_TIMER_PT_CMD	–	N/A	Protocol timer command
		(0) COMMANDS_START_SUBBAND	–	N/A	Subband selection algorithm
0x400409E0	RF0_REG51	(31:24) FSM_MODE_RM_TX	(31:24) FSM_MODE_RM_TX	0x0	Remapped register of FSM_MODE
		(23:16) PA_PWR_RM	(23:16) PA_PWR_RM	0x0	Remapped register of PA_PWR
		(15:8) CHANNEL_RM_TX	(15:8) CHANNEL_RM_TX	0x0	Remapped register of CHANNEL
		(7:0) RATE_TX	(7:0) RATE_TX	0x0	Remapped register of BANK
0x400409E8	RF0_REG52	(31:24) ACCESS_ADDRESS	(31:24) ACCESS_ADDRESS	0x0	Remapped register of PATTERN
		(23:16) FSM_MODE_RM_RX	(23:16) FSM_MODE_RM_RX	0x0	Remapped register of FSM_MODE
		(15:8) CHANNEL_RM_RX	(15:8) CHANNEL_RM_RX	0x0	Remapped register of CHANNEL
		(7:0) RATE_RX	(7:0) RATE_RX	0x0	Remapped register of BANK
0x400409F0	RF0_REG53	–	(23:16) RSSI_MAX_RM	0x0	Remapped register of RSSI_MAX
		–	(15:8) RSSI_MIN_RM	0x0	Remapped register of RSSI_MIN

Address	Register Name	Register Write	Register Read	Default	Description
		–	(7:0) RSSI_AVG_RM	0x0	Remapped register of RSSI_AVG
0x400409F4	RF0_REG54	(7:0) BLR_PACKET_LEN	–	N/A	Remapped register of BLR_PACKET_LEN
0x400409F8	RF0_REG55	–	(7:0) ITRX_FEATURES	0x0	Remapped register of ITRX_FEATURES
0x400409FC	RF0_REG56	–	(31:24) CHIP_ID_CHIP_ID	0x30	Version of the chip
		–	(23:16) MD5_REGS_MD5_REGS	0x0	MD5 calculated on the register map file
		(15:8) SCAN_2_SCAN_2_PASSWORD	–	N/A	SCAN 2 key
		(7:0) SCAN_1_SCAN_1_PASSWORD	–	N/A	SCAN 1 key
0x40040A00	RF1_REG00	(31) DATAWHITE_BTLE_DW_BTLE	(31) DATAWHITE_BTLE_DW_BTLE	0x1	Data whitening control
		(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	Enable 4FSK coding
		(22:20) FOURFSK_CODING_TX_FOURFSK_CODING	(22:20) FOURFSK_CODING_TX_FOURFSK_CODING	0x0	Set the 4FSK coding (Tx mode)
		(18:16) FOURFSK_CODING_RX_FOURFSK_CODING	(18:16) FOURFSK_CODING_RX_FOURFSK_CODING	0x0	Set the 4FSK decoding (Rx mode)
		(14) MODE2_DIFF_CODING	(14) MODE2_DIFF_CODING	0x0	Differential coding/decoding

Address	Register Name	Register Write	Register Read	Default	Description
		(13) MODE2_PSK_NFSK	(13) MODE2_PSK_NFSK	0x0	FSK/PSK mode selection
		(12:8) MODE2_TESTMODE	(12:8) MODE2_TESTMODE	0x0	Output test mode
		(7) MODE_NOT_TO_IDLE	(7) MODE_NOT_TO_IDLE	0x0	FSM goes in suspend mode after a Tx or Rx packet
		(5) MODE_EN_FSM	(5) MODE_EN_FSM	0x1	Radio FSM control
		(4) MODE_EN_DESERIALIZER	(4) MODE_EN_DESERIALIZER	0x0	Deserializer control
		(3) MODE_EN_SERIALIZER	(3) MODE_EN_SERIALIZER	0x0	Serializer control
		(2) MODE_TX_NRX	(2) MODE_TX_NRX	0x0	Select Tx or Rx mode
		(1:0) MODE_MODE	(1:0) MODE_MODE	0x2	Select the working mode of the digital baseband
0x40040A04	RF1_REG01	(31:24) TAU_PHASE_RECOV_TAU_PHASE_RECOV	(31:24) TAU_PHASE_RECOV_TAU_PHASE_RECOV	0x14	Time constant of the fine carrier recovery block (banked)
		(23:16) TAU_ROUGH_RECOV_TAU_ROUGH_RECOV	(23:16) TAU_ROUGH_RECOV_TAU_ROUGH_RECOV	0xB	Time constant of the rough carrier recovery block (banked)
		(15) CARRIER_RECOVERY_EN_CORRECT_CFREQ_AFC	(15) CARRIER_RECOVERY_EN_CORRECT_CFREQ_AFC	0x0	Automatic AFC correction (banked)
		(14) CARRIER_RECOVERY_CORRECT_CFREQ_IF_NEG	(14) CARRIER_RECOVERY_CORRECT_CFREQ_IF_NEG	0x0	IF correction (banked)
		(13) CARRIER_RECOVERY_EN_CORRECT_CFREQ_IF	(13) CARRIER_RECOVERY_EN_CORRECT_CFREQ_IF	0x1	Automatic IF correction (banked)

Address	Register Name	Register Write	Register Read	Default	Description
		(12) CARRIER_ RECOVERY_AFC_NEG	(12) CARRIER_ RECOVERY_AFC_NEG	0x0	AFC correction (banked)
		(11) CARRIER_ RECOVERY_STARTER_ MODE	(11) CARRIER_ RECOVERY_STARTER_ MODE	0x0	Starter mode (banked)
		(10) CARRIER_ RECOVERY_EN_AFC	(10) CARRIER_ RECOVERY_EN_AFC	0x0	Automatic frequency control (banked)
		(9) CARRIER_ RECOVERY_EN_FINE_ RECOV	(9) CARRIER_ RECOVERY_EN_FINE_ RECOV	0x1	Fine carrier recovery (banked)
		(8) CARRIER_ RECOVERY_EN_ROUGH_ RECOV	(8) CARRIER_ RECOVERY_EN_ROUGH_ RECOV	0x0	Rough carrier recovery (banked)
		(6) MOD_TX_PULSE_ NSYM	(6) MOD_TX_PULSE_ NSYM	0x0	Tx pulse shape function
		(5) MOD_TX_EN_ INTERP	(5) MOD_TX_EN_ INTERP	0x0	Tx CIC interpolator
		(4:0) MOD_TX_CK_TX_ M	(4:0) MOD_TX_CK_ TX_M	0x0	Unsigned value determining the Tx CIC interpolator frequency
0x40040A08	RF1_REG02	(25:24) DATARATE_ OFFSET_DR_LIMIT	(25:24) DATARATE_ OFFSET_DR_LIMIT	0x0	Set the data-rate recovery limits
		(23:16) DATARATE_ OFFSET_DATARATE_ OFFSET	(23:16) DATARATE_ OFFSET_DATARATE_ OFFSET	0x0	Data-rate offset
		(15:8) TAU_ DATARATE_RECOV_TAU_ DATARATE_RECOV	(15:8) TAU_ DATARATE_RECOV_ TAU_DATARATE_RECOV	0x20	Time constant of the data-rate recovery
		(7:0) TAU_CLK_ RECOV_TAU_CLK_RECOV	(7:0) TAU_CLK_ RECOV_TAU_CLK_	0x9	Time constant of the clock recovery (banked)

Address	Register Name	Register Write	Register Read	Default	Description
			RECOV		
0x40040A0C	RF1_REG03	(31:30) MAC_CONF_MAC_TIMER_GR	(31:30) MAC_CONF_MAC_TIMER_GR	0x2	MAC timer granularity
		(29) MAC_CONF_RX_MAC_ACT	(29) MAC_CONF_RX_MAC_ACT	0x0	Switch FSM to Rx or Tx mode after an Rx mode
		(28) MAC_CONF_RX_MAC_TX_NRX	(28) MAC_CONF_RX_MAC_TX_NRX	0x0	Switch FSM to Tx mode after an Rx mode (Rx otherwise)
		(27) MAC_CONF_RX_MAC_START_NSTOP	(27) MAC_CONF_RX_MAC_START_NSTOP	0x0	MAC timer activation after sync word detection
		(26) MAC_CONF_TX_MAC_ACT	(26) MAC_CONF_TX_MAC_ACT	0x0	Switch FSM to Rx or Tx mode after a Tx mode
		(25) MAC_CONF_TX_MAC_TX_NRX	(25) MAC_CONF_TX_MAC_TX_NRX	0x0	Switch FSM to Tx mode after a Tx mode (Rx otherwise)
		(24) MAC_CONF_TX_MAC_START_NSTOP	(24) MAC_CONF_TX_MAC_START_NSTOP	0x0	MAC timer activation after packet transmission
		(23) IRQ_CONF_IRQ_HIGH_Z	(23) IRQ_CONF_IRQ_HIGH_Z	0x0	Pads are set to high-Z when the IRQ is not active
		(22) IRQ_CONF_IRQ_ACTIVE_LOW	(22) IRQ_CONF_IRQ_ACTIVE_LOW	0x1	IRQ are active low
		(21:16) IRQ_CONF_IRQS_MASK	(21:16) IRQ_CONF_IRQS_MASK	0x0	Mask to determine which IRQs are enabled (active high)
		(15:13) FIFO_2_FIFO_THR_TX	(15:13) FIFO_2_FIFO_THR_TX	0x0	Threshold indicating the "almost empty" Tx FIFO state
		(12) FIFO_2_WAIT_TXFIFO_WR	(12) FIFO_2_WAIT_TXFIFO_WR	0x0	FSM will wait a Tx FIFO write before starting the Tx mode in case of an empty Tx FIFO
		(11) FIFO_2_STOP_ON_RXFF_OVFLW	(11) FIFO_2_STOP_ON_RXFF_OVFLW	0x0	Stop the reception in case of a FIFO overflow

Address	Register Name	Register Write	Register Read	Default	Description
		(10) FIFO_2_STOP_ON_TXFF_UNFLW	(10) FIFO_2_STOP_ON_TXFF_UNFLW	0x0	Stop the transmission in case of a FIFO underflow
		(9) FIFO_2_RXFF_FLUSH_ON_START	(9) FIFO_2_RXFF_FLUSH_ON_START	0x1	Flush the Rx FIFO when the Rx mode is enabled in order to receive a packet with an empty FIFO
		(8) FIFO_2_TXFF_FLUSH_ON_STOP	(8) FIFO_2_TXFF_FLUSH_ON_STOP	0x1	Flush the Tx FIFO after the end of a packet transmission in order to have an empty FIFO
		(7) FIFO_FIFO_FLUSH_ON_OVFLW	(7) FIFO_FIFO_FLUSH_ON_OVFLW	0x0	Overflow FIFO flush control
		(6) FIFO_FIFO_FLUSH_ON_ADDR_ERR	(6) FIFO_FIFO_FLUSH_ON_ADDR_ERR	0x0	Address error FIFO flush control
		(5) FIFO_FIFO_FLUSH_ON_PL_ERR	(5) FIFO_FIFO_FLUSH_ON_PL_ERR	0x0	Packet length error FIFO flush control
		(4) FIFO_FIFO_FLUSH_ON_CRC_ERR	(4) FIFO_FIFO_FLUSH_ON_CRC_ERR	0x1	CRC error FIFO flush control
		(3) FIFO_RX_FIFO_ACK	(3) FIFO_RX_FIFO_ACK	0x0	Rx FIFO acknowledgement
		(2:0) FIFO_FIFO_THR	(2:0) FIFO_FIFO_THR	0x0	Threshold indicating the "almost full" Rx FIFO state
0x40040A10	RF1_PADS_03	(28:24) PAD_CONF_1_PAD_3_CONF	(28:24) PAD_CONF_1_PAD_3_CONF	0x0	Configuration of GPIO pad 3
		(20:16) PAD_CONF_1_PAD_2_CONF	(20:16) PAD_CONF_1_PAD_2_CONF	0x0	Configuration of GPIO pad 2
		(12:8) PAD_CONF_1_PAD_1_CONF	(12:8) PAD_CONF_1_PAD_1_CONF	0x0	Configuration of GPIO pad 1
		(4:0) PAD_CONF_1_PAD_0_CONF	(4:0) PAD_CONF_1_PAD_0_CONF	0x0	Configuration of GPIO pad 0

Address	Register Name	Register Write	Register Read	Default	Description
0x40040A14	RF1_PADS_47	(28:24) PAD_CONF_2_PAD_7_CONF	(28:24) PAD_CONF_2_PAD_7_CONF	0x0	Configuration of GPIO pad 7
		(20:16) PAD_CONF_2_PAD_6_CONF	(20:16) PAD_CONF_2_PAD_6_CONF	0x0	Configuration of GPIO pad 6
		(12:8) PAD_CONF_2_PAD_5_CONF	(12:8) PAD_CONF_2_PAD_5_CONF	0x0	Configuration of GPIO pad 5
		(4:0) PAD_CONF_2_PAD_4_CONF	(4:0) PAD_CONF_2_PAD_4_CONF	0x0	Configuration of GPIO pad 4
0x40040A18	RF1_CENTER_FREQ	(31) CENTER_FREQ_ADAPT_CFREQ	(31) CENTER_FREQ_ADAPT_CFREQ	0x1	Frequency adaptation between Tx and Rx modes
		(30) CENTER_FREQ_RX_DIV_5_N6	(30) CENTER_FREQ_RX_DIV_5_N6	0x0	Ratio of the PLL reference between Tx and Rx modes
		(29:0) CENTER_FREQ_CENTER_FREQUENCY	(29:0) CENTER_FREQ_CENTER_FREQUENCY	0x215C71B	Set the center frequency
0x40040A1C	RF1_PADS_89	(31:24) TX_MAC_TIMER_TX_MAC_TIMER	(31:24) TX_MAC_TIMER_TX_MAC_TIMER	0x82	Time to wait after the Tx mode
		(23:16) RX_MAC_TIMER_RX_MAC_TIMER	(23:16) RX_MAC_TIMER_RX_MAC_TIMER	0x23	Time to wait after the Rx mode
		(12:8) PAD_CONF_3_PAD_9_CONF	(12:8) PAD_CONF_3_PAD_9_CONF	0x0	Configuration of GPIO pad 9
		(4:0) PAD_CONF_3_PAD_8_CONF	(4:0) PAD_CONF_3_PAD_8_CONF	0x0	Configuration of GPIO pad 8
0x40040A20	RF1_REG08	(31:30) MOD_INFO_RX_DIV_CK_RX	(31:30) MOD_INFO_RX_DIV_CK_RX	0x0	Set the clock divider for the Rx mode (banked)
		(29) MOD_INFO_RX_SYMBOL_2BIT_RX	(29) MOD_INFO_RX_SYMBOL_2BIT_RX	0x0	Rx symbol bits composition (banked)
		(28:24) MOD_INFO_	(28:24) MOD_INFO_	0x0	Unsigned value determining the

Address	Register Name	Register Write	Register Read	Default	Description
		RX_DR_M_RX	RX_DR_M_RX		oversampling frequency and consequently the data-rate (banked)
		(23:22) MOD_INFO_ TX_DIV_CK_TX	(23:22) MOD_INFO_ TX_DIV_CK_TX	0x0	Set the clock divider for the Tx mode (banked)
		(21) MOD_INFO_TX_ SYMBOL_2BIT_TX	(21) MOD_INFO_TX_ SYMBOL_2BIT_TX	0x0	Tx symbol bits composition (banked)
		(20:16) MOD_INFO_ TX_DR_M_TX	(20:16) MOD_INFO_ TX_DR_M_TX	0x0	Unsigned value determining the oversampling frequency and consequently the data-rate (banked)
		(14) CHANNEL_ SWITCH_IQ	(14) CHANNEL_ SWITCH_IQ	0x0	Switch I and Q channels
		(13:8) CHANNEL_ CHANNEL	(13:8) CHANNEL_ CHANNEL	0x0	Channel number
		(3) BANK_DATARATE_ TX_NRX	(3) BANK_DATARATE_ TX_NRX	0x0	Select the data-rate register
		(2) BANK_STD_BLE_ RATES	(2) BANK_STD_BLE_ RATES	0x0	Select the actual bank behavior
		(1:0) BANK_BANK	(1:0) BANK_BANK	0x0	Select the used bank
0x40040A24	RF1_CODING	(31) CODING_EN_ DATAWHITE	(31) CODING_EN_ DATAWHITE	0x1	Data-whitening enabling (banked)
		(30) CODING_I_NQ_ DELAYED	(30) CODING_I_NQ_ DELAYED	0x0	Channel I delay (banked)
		(29) CODING_OFFSET	(29) CODING_OFFSET	0x0	Offset (delay) introduction (banked)
		(28) CODING_BIT_ INVERT	(28) CODING_BIT_ INVERT	0x0	Bit value inversion in Tx and Rx modes (banked)
		(27) CODING_EVEN_ BEFORE_ODD	(27) CODING_EVEN_ BEFORE_ODD	0x0	Determine the bit order in case of a 2 bits per symbol modulation (banked)
		(26) CODING_EN_	(26) CODING_EN_	0x0	Linear to frequency encoding needed

Address	Register Name	Register Write	Register Read	Default	Description
		802154_L2F	802154_L2F		in order to modulate an OQPSK as an MSK (banked)
		(25) CODING_EN_802154_B2C	(25) CODING_EN_802154_B2C	0x0	Bit to chips encoding used in the IEEE 802.15.4 standard (banked)
		(24) CODING_EN_MANCHESTER	(24) CODING_EN_MANCHESTER	0x0	Manchester encoding (banked)
		(23) CHANNELS_2_EN_CHANNEL_SEL	(23) CHANNELS_2_EN_CHANNEL_SEL	0x1	Definition of channels (banked)
		(22) CHANNELS_2_EN_CHN_BLE	(22) CHANNELS_2_EN_CHN_BLE	0x1	BLE channels index LUT (banked)
		(19:16) CHANNELS_2_CHANNEL_SPACING_HI	(19:16) CHANNELS_2_CHANNEL_SPACING_HI	0x7	Channel spacing MSB (banked)
		(15:0) CHANNELS_1_CHANNEL_SPACING_LO	(15:0) CHANNELS_1_CHANNEL_SPACING_LO	0x1C72	Channel spacing LSB (banked)
0x40040A28	RF1_PACKET_HANDLING	(31:24) PREAMBLE_PREAMBLE	(31:24) PREAMBLE_PREAMBLE	0x55	Preamble to be inserted (banked)
		(22) PACKET_LENGTH_OPTS_EN_PACKET_LEN_FIX	(22) PACKET_LENGTH_OPTS_EN_PACKET_LEN_FIX	0x0	Packet length configuration (banked)
		(21:18) PACKET_LENGTH_OPTS_PACKET_LEN_CORR	(21:18) PACKET_LENGTH_OPTS_PACKET_LEN_CORR	0x0	Signed value specifying the correction to apply to the specified packet length (banked)
		(17:16) PACKET_LENGTH_OPTS_PACKET_LEN_POS	(17:16) PACKET_LENGTH_OPTS_PACKET_LEN_POS	0x1	Unsigned value that specifies the position of the packet length after the pattern (banked)
		(15:8) PACKET_LENGTH_PACKET_LEN	(15:8) PACKET_LENGTH_PACKET_LEN	0xFF	The packet length in the fixed packet length mode (banked)
		(7) PACKET_	(7) PACKET_	0x1	Select LSB or MSB to send first

Address	Register Name	Register Write	Register Read	Default	Description
		HANDLING_LSB_FIRST	HANDLING_LSB_FIRST		(banked)
		(6) PACKET_ HANDLING_EN_CRC	(6) PACKET_ HANDLING_EN_CRC	0x1	Automatic CRC evaluation and insertion (banked)
		(5) PACKET_ HANDLING_EN_CRC_ON_ PKTLEN	(5) PACKET_ HANDLING_EN_CRC_ ON_PKTLEN	0x1	CRC calculation on the packet length part of the packet (banked)
		(4) PACKET_ HANDLING_EN_ PREAMBLE	(4) PACKET_ HANDLING_EN_ PREAMBLE	0x1	Automatic preamble insertion (banked)
		(3) PACKET_ HANDLING_EN_MULTI_ FRAME	(3) PACKET_ HANDLING_EN_MULTI_ FRAME	0x0	Multi-frame packet (banked)
		(2) PACKET_ HANDLING_ENB_DW_ON_ CRC	(2) PACKET_ HANDLING_ENB_DW_ ON_CRC	0x0	Data-whitening on the CRC disabling (banked)
		(1) PACKET_ HANDLING_EN_PATTERN	(1) PACKET_ HANDLING_EN_ PATTERN	0x1	Automatic pattern insertion and recognition (banked)
		(0) PACKET_ HANDLING_EN_PACKET	(0) PACKET_ HANDLING_EN_PACKET	0x1	Packet handler enabling (banked)
0x40040A2C	RF1_SYNC_PATTERN	(31:0) PATTERN	(31:0) PATTERN	0x8E89BED6	Pattern (sync word) to be inserted or recognized (banked)
0x40040A30	RF1_REG0C	(31:16) ADDRESS_ ADDRESS	(31:16) ADDRESS_ ADDRESS	0x0	Address of the node (banked)
		(11) ADDRESS_CONF_ ADDRESS_LEN	(11) ADDRESS_CONF_ ADDRESS_LEN	0x0	Address length selection (banked)
		(10) ADDRESS_CONF_ EN_ADDRESS_RX_BR	(10) ADDRESS_CONF_ EN_ADDRESS_RX_BR	0x0	Broadcast address detection in Rx mode (banked)
		(9) ADDRESS_CONF_	(9) ADDRESS_CONF_	0x0	Address detection in Rx mode

Address	Register Name	Register Write	Register Read	Default	Description
		EN_ADDRESS_RX	EN_ADDRESS_RX		(banked)
		(8) ADDRESS_CONF_ EN_ADDRESS_TX	(8) ADDRESS_CONF_ EN_ADDRESS_TX	0x0	Address insertion in Tx mode (banked)
		(7:0) PREAMBLE_ LENGTH_PREAMBLE_LEN	(7:0) PREAMBLE_ LENGTH_PREAMBLE_ LEN	0x0	Length of the preamble -1 (banked)
0x40040A34	RF1_PACKET_EXTRA	(29:28) CONV_CODES_ CONF_STOP_WORD_LEN	(29:28) CONV_ CODES_CONF_STOP_ WORD_LEN	0x0	Length of the stop word (banked)
		(27:26) CONV_CODES_ CONF_CC_VITERBI_LEN	(27:26) CONV_ CODES_CONF_CC_ VITERBI_LEN	0x2	Set the memory length of the Viterbi decoder (banked)
		(25) CONV_CODES_ CONF_CC_EN_TX_STOP	(25) CONV_CODES_ CONF_CC_EN_TX_STOP	0x0	Stop word at the end of the transmission (banked)
		(24) CONV_CODES_ CONF_EN_CONV_CODE	(24) CONV_CODES_ CONF_EN_CONV_CODE	0x0	Convolutional codes (banked)
		(22) PACKET_EXTRA_ FIFO_REWIND	(22) PACKET_EXTRA_ FIFO_REWIND	0x0	Rewind the FIFO to the initial stage at the end of a Tx transmission (banked)
		(21) PACKET_EXTRA_ BLE_PREAMBLE	(21) PACKET_EXTRA_ BLE_PREAMBLE	0x1	Handle the preamble directly in Tx mode (PREAMBLE register is not used) according to the BLE standard (banked)
		(20) PACKET_EXTRA_ PKT_INFO_PRE_NPOST	(20) PACKET_EXTRA_ PKT_INFO_PRE_NPOST	0x0	Packet information sampling (banked)
		(19:18) PACKET_ EXTRA_PATTERN_MAX_ ERR	(19:18) PACKET_ EXTRA_PATTERN_MAX_ ERR	0x0	Unsigned value that specifies the maximum number of errors in the pattern recognition (banked)
		(17:16) PACKET_ EXTRA_PATTERN_WORD_	(17:16) PACKET_ EXTRA_PATTERN_	0x3	Pattern word length (banked)

Address	Register Name	Register Write	Register Read	Default	Description
		LEN	WORD_LEN		
		(15:0) ADDRESS_BROADCAST_ADDRESS_BR	(15:0) ADDRESS_BROADCAST_ADDRESS_BR	0x0	Broadcast address (banked)
0x40040A38	RF1_CRC_POLYNOMIAL	(31:0) CRC_POLY	(31:0) CRC_POLY	0x80032D	CRC polynomial (banked)
0x40040A3C	RF1_CRC_RST	(31:0) CRC_RST	(31:0) CRC_RST	0x555555	CRC reset value (banked)
0x40040A40	RF1_REG10	(25:21) CONV_CODES_PUNCT_CC_PUNCT_1	(25:21) CONV_CODES_PUNCT_CC_PUNCT_1	0x1	Puncture of the second convolutional code (banked)
		(20:16) CONV_CODES_PUNCT_CC_PUNCT_0	(20:16) CONV_CODES_PUNCT_CC_PUNCT_0	0x1	Puncture of the first convolutional code (banked)
		(11) FRAC_CONF_TX_FRAC_GAIN	(11) FRAC_CONF_TX_FRAC_GAIN	0x0	Additional gain for fractional data-rates in Tx mode (banked)
		(10) FRAC_CONF_RX_FRAC_GAIN	(10) FRAC_CONF_RX_FRAC_GAIN	0x0	Additional gain for fractional data-rates in Rx mode (banked)
		(9) FRAC_CONF_TX_EN_FRAC	(9) FRAC_CONF_TX_EN_FRAC	0x0	Fractional data-rates in Tx mode (banked)
		(8) FRAC_CONF_RX_EN_FRAC	(8) FRAC_CONF_RX_EN_FRAC	0x0	Fractional data-rates in Rx mode (banked)
		(7:4) CONV_CODES_POLY_CC_POLY_1	(7:4) CONV_CODES_POLY_CC_POLY_1	0xD	Second convolutional code (banked)
		(3:0) CONV_CODES_POLY_CC_POLY_0	(3:0) CONV_CODES_POLY_CC_POLY_0	0xF	First convolutional code (banked)
0x40040A44	RF1_REG11	(31) FILTER_GAIN_LIN_FILTER	(31) FILTER_GAIN_LIN_FILTER	0x0	Enable the linear filtering (banked)
		(30) FILTER_GAIN_LOW_LIN_GAIN	(30) FILTER_GAIN_LOW_LIN_GAIN	0x0	Reduce the total gain by two if the linear gain is set (banked)

Address	Register Name	Register Write	Register Read	Default	Description
		(29:27) FILTER_GAIN_GAIN_M	(29:27) FILTER_GAIN_GAIN_M	0x0	Mantissa of the final stage gain of the matched filter (banked)
		(26:24) FILTER_GAIN_GAIN_E	(26:24) FILTER_GAIN_GAIN_E	0x0	Exponent of the final stage gain of the matched filter (banked)
		(23:20) TX_MULT_TX_MULT_EXP	(23:20) TX_MULT_TX_MULT_EXP	0x2	Exponent of the Tx multiplier (banked)
		(19:16) TX_MULT_TX_MULT_MAN	(19:16) TX_MULT_TX_MULT_MAN	0x9	Mantissa of the Tx multiplier (banked)
		(15:12) TX_FRAC_CONF_TX_FRAC_DEN	(15:12) TX_FRAC_CONF_TX_FRAC_DEN	0x0	Denominator of the fractional data-rate in Tx mode (banked)
		(11:8) TX_FRAC_CONF_TX_FRAC_NUM	(11:8) TX_FRAC_CONF_TX_FRAC_NUM	0x0	Numerator of the fractional data-rate in Tx mode (banked)
		(7:4) RX_FRAC_CONF_RX_FRAC_DEN	(7:4) RX_FRAC_CONF_RX_FRAC_DEN	0x0	Denominator of the fractional data-rate in Rx mode (banked)
		(3:0) RX_FRAC_CONF_RX_FRAC_NUM	(3:0) RX_FRAC_CONF_RX_FRAC_NUM	0x0	Numerator of the fractional data-rate in Rx mode (banked)
0x40040A48	RF1_TX_PULSE_SHAPE_1	(31:24) TX_PULSE_SHAPE_1_TX_COEF4	(31:24) TX_PULSE_SHAPE_1_TX_COEF4	0x0	Tx pulse shape coefficient 4 (banked)
		(23:16) TX_PULSE_SHAPE_1_TX_COEF3	(23:16) TX_PULSE_SHAPE_1_TX_COEF3	0x0	Tx pulse shape coefficient 3 (banked)
		(15:8) TX_PULSE_SHAPE_1_TX_COEF2	(15:8) TX_PULSE_SHAPE_1_TX_COEF2	0x0	Tx pulse shape coefficient 2 (banked)
		(7:0) TX_PULSE_SHAPE_1_TX_COEF1	(7:0) TX_PULSE_SHAPE_1_TX_COEF1	0x0	Tx pulse shape coefficient 1 (banked)
0x40040A4C	RF1_TX_PULSE_SHAPE_2	(31:24) TX_PULSE_SHAPE_2_TX_COEF8	(31:24) TX_PULSE_SHAPE_2_TX_COEF8	0x2	Tx pulse shape coefficient 8 (banked)
		(23:16) TX_PULSE_SHAPE_2_TX_COEF7	(23:16) TX_PULSE_SHAPE_2_TX_COEF7	0x1	Tx pulse shape coefficient 7 (banked)

Address	Register Name	Register Write	Register Read	Default	Description
		(15:8) TX_PULSE_SHAPE_2_TX_COEF6	(15:8) TX_PULSE_SHAPE_2_TX_COEF6	0x0	Tx pulse shape coefficient 6 (banked)
		(7:0) TX_PULSE_SHAPE_2_TX_COEF5	(7:0) TX_PULSE_SHAPE_2_TX_COEF5	0x0	Tx pulse shape coefficient 5 (banked)
0x40040A50	RF1_TX_PULSE_SHAPE_3	(31:24) TX_PULSE_SHAPE_3_TX_COEF12	(31:24) TX_PULSE_SHAPE_3_TX_COEF12	0x36	Tx pulse shape coefficient 12 (banked)
		(23:16) TX_PULSE_SHAPE_3_TX_COEF11	(23:16) TX_PULSE_SHAPE_3_TX_COEF11	0x20	Tx pulse shape coefficient 11 (banked)
		(15:8) TX_PULSE_SHAPE_3_TX_COEF10	(15:8) TX_PULSE_SHAPE_3_TX_COEF10	0x10	Tx pulse shape coefficient 10 (banked)
		(7:0) TX_PULSE_SHAPE_3_TX_COEF9	(7:0) TX_PULSE_SHAPE_3_TX_COEF9	0x7	Tx pulse shape coefficient 9 (banked)
0x40040A54	RF1_TX_PULSE_SHAPE_4	(31:24) TX_PULSE_SHAPE_4_TX_COEF16	(31:24) TX_PULSE_SHAPE_4_TX_COEF16	0x7D	Tx pulse shape coefficient 16 (banked)
		(23:16) TX_PULSE_SHAPE_4_TX_COEF15	(23:16) TX_PULSE_SHAPE_4_TX_COEF15	0x75	Tx pulse shape coefficient 15 (banked)
		(15:8) TX_PULSE_SHAPE_4_TX_COEF14	(15:8) TX_PULSE_SHAPE_4_TX_COEF14	0x66	Tx pulse shape coefficient 14 (banked)
		(7:0) TX_PULSE_SHAPE_4_TX_COEF13	(7:0) TX_PULSE_SHAPE_4_TX_COEF13	0x4F	Tx pulse shape coefficient 13 (banked)
0x40040A58	RF1_FRONTEND	(25:16) RX_IF_DIG_IF_DIG	(25:16) RX_IF_DIG_IF_DIG	0x40	IF frequency (banked)
		(14:11) FRONTEND_RESAMPLE_PH_GAIN	(14:11) FRONTEND_RESAMPLE_PH_GAIN	0x6	Gain of the phase resampling block (banked)
		(10:8) FRONTEND_RESAMPLE_RSSI_G2	(10:8) FRONTEND_RESAMPLE_RSSI_G2	0x0	Gain of the decimator in the RSSI resampling block (banked)
		(7:6) FRONTEND_RESAMPLE_RSSI_G1	(7:6) FRONTEND_RESAMPLE_RSSI_G1	0x0	Gain of the interpolator in the RSSI resampling block (banked)

Address	Register Name	Register Write	Register Read	Default	Description
		(5) FRONTEND_EN_RESAMPLE_RSSI	(5) FRONTEND_EN_RESAMPLE_RSSI	0x0	RSSI resampling (banked)
		(4) FRONTEND_EN_RESAMPLE_PHADC	(4) FRONTEND_EN_RESAMPLE_PHADC	0x1	Phase resampling (banked)
		(3:0) FRONTEND_DIV_PHADC	(3:0) FRONTEND_DIV_PHADC	0x0	Unsigned value that specifies the divider to obtain the phase ADC clock and RSSI (banked)
0x40040A5C	RF1_RX_PULSE_SHAPE	(31:28) RX_PULSE_SHAPE_RX_COEF8	(31:28) RX_PULSE_SHAPE_RX_COEF8	0xF	Rx pulse shape coefficient 8 (banked)
		(27:24) RX_PULSE_SHAPE_RX_COEF7	(27:24) RX_PULSE_SHAPE_RX_COEF7	0xE	Rx pulse shape coefficient 7 (banked)
		(23:20) RX_PULSE_SHAPE_RX_COEF6	(23:20) RX_PULSE_SHAPE_RX_COEF6	0xC	Rx pulse shape coefficient 6 (banked)
		(19:16) RX_PULSE_SHAPE_RX_COEF5	(19:16) RX_PULSE_SHAPE_RX_COEF5	0xA	Rx pulse shape coefficient 5 (banked)
		(15:12) RX_PULSE_SHAPE_RX_COEF4	(15:12) RX_PULSE_SHAPE_RX_COEF4	0x7	Rx pulse shape coefficient 4 (banked)
		(11:8) RX_PULSE_SHAPE_RX_COEF3	(11:8) RX_PULSE_SHAPE_RX_COEF3	0x4	Rx pulse shape coefficient 3 (banked)
		(7:4) RX_PULSE_SHAPE_RX_COEF2	(7:4) RX_PULSE_SHAPE_RX_COEF2	0x2	Rx pulse shape coefficient 2 (banked)
		(3:0) RX_PULSE_SHAPE_RX_COEF1	(3:0) RX_PULSE_SHAPE_RX_COEF1	0x1	Rx pulse shape coefficient 1 (banked)
0x40040A60	RF1_REG18	(28) DELAY_LINE_CONF_MULTI_SYNC	(28) DELAY_LINE_CONF_MULTI_SYNC	0x0	Detect multiple syncs (banked)
		(27:25) DELAY_LINE_CONF_DL_ISI_THR	(27:25) DELAY_LINE_CONF_DL_ISI_THR	0x1	Threshold bias for ISI compensation in the delay line sync word comparator (banked)

Address	Register Name	Register Write	Register Read	Default	Description
		(22) DELAY_LINE_CONF_EN_SYNC_OK_DELAY_LINE	(22) DELAY_LINE_CONF_EN_SYNC_OK_DELAY_LINE	0x1	Use pattern_ok signal in delay line to synchronize the deserializer (banked)
		(21:20) DELAY_LINE_CONF_MAX_ERR_IN_DL_SYNC	(21:20) DELAY_LINE_CONF_MAX_ERR_IN_DL_SYNC	0x0	Set the maximum errors in the delay line sync detection (banked)
		(19) DELAY_LINE_CONF_EN_NOT_CAUSAL	(19) DELAY_LINE_CONF_EN_NOT_CAUSAL	0x0	Non causal processing (banked)
		(18:16) DELAY_LINE_CONF_NC_SEL_OUT	(18:16) DELAY_LINE_CONF_NC_SEL_OUT	0x0	Select the output position for the non causal processing (banked)
		(15:8) FSK_FCR_AMP_1_FSK_FCR_AMP1	(15:8) FSK_FCR_AMP_1_FSK_FCR_AMP1	0x1B	FSK amplitude low (banked)
		(6:4) CARRIER_RECOVERY_EXTRA_FREQ_LIMIT_MAN	(6:4) CARRIER_RECOVERY_EXTRA_FREQ_LIMIT_MAN	0x5	Mantissa of the carrier recovery frequency limit (banked)
		(2:0) CARRIER_RECOVERY_EXTRA_FREQ_LIMIT_EXP	(2:0) CARRIER_RECOVERY_EXTRA_FREQ_LIMIT_EXP	0x0	Exponent of the carrier recovery frequency limit (banked)
0x40040A64	RF1_REG19	(30) RSSI_BANK_EN_RSSI_DITHER	(30) RSSI_BANK_EN_RSSI_DITHER	0x0	Speed on the RSSI triangular dithering signal (banked)
		(29) RSSI_BANK_FAST_RSSI	(29) RSSI_BANK_FAST_RSSI	0x0	RSSI filtering speed (banked)
		(28) RSSI_BANK_EN_FAST_PRE_SYNC	(28) RSSI_BANK_EN_FAST_PRE_SYNC	0x1	Fast mode switching during the preamble reception (banked)
		(27:24) RSSI_BANK_TAU_RSSI_FILTERING	(27:24) RSSI_BANK_TAU_RSSI_FILTERING	0x1	Time constant of the RSSI filtering block (banked)
		(20) DECISION_USE_VIT_SOFT	(20) DECISION_USE_VIT_SOFT	0x0	Viterbi soft decoding (banked)

Address	Register Name	Register Write	Register Read	Default	Description
		(19:18) DECISION_ VITERBI_LEN	(19:18) DECISION_ VITERBI_LEN	0x2	Set the Viterbi path length (banked)
		(17) DECISION_ VITERBI_POW_NLIN	(17) DECISION_ VITERBI_POW_NLIN	0x1	Viterbi algorithm uses power instead of amplitude to evaluate the error on the path (banked)
		(16) DECISION_EN_ VITERBI_GFSK	(16) DECISION_EN_ VITERBI_GFSK	0x1	Viterbi algorithm for the GFSK decoding (banked)
		(15:8) FSK_FCR_AMP_ 3_FSK_FCR_AMP3	(15:8) FSK_FCR_ AMP_3_FSK_FCR_AMP3	0x44	FSK amplitude high (banked)
		(7:0) FSK_FCR_AMP_ 2_FSK_FCR_AMP2	(7:0) FSK_FCR_AMP_ 2_FSK_FCR_AMP2	0x30	FSK amplitude mid (banked)
0x40040A68	RF1_REG1A	(28:24) PA_PWR_PA_ PWR	(28:24) PA_PWR_PA_ PWR	0xC	Signed value that sets the PA power
		(22) RSSI_BANK_ALT_ USE_RSSI_ALT	(22) RSSI_BANK_ ALT_USE_RSSI_ALT	0x0	Use alternative RRSI configuration (banked)
		(21) RSSI_BANK_ALT_ FAST_RSSI_ALT	(21) RSSI_BANK_ ALT_FAST_RSSI_ALT	0x0	RSSI filtering speed (banked)
		(19:16) RSSI_BANK_ ALT_TAU_RSSI_ FILTERING_ALT	(19:16) RSSI_BANK_ ALT_TAU_RSSI_ FILTERING_ALT	0x3	Time constant of the RSSI filtering block (banked)
		(15:0) CORRECT_ CFREQ_IF_CORRECT_ CFREQ_IF	(15:0) CORRECT_ CFREQ_IF_CORRECT_ CFREQ_IF	0x1555	Unsigned value that specifies the IF for the Rx mode (banked)
0x40040A6C	RF1_REG1B	(31) PLL_BANK_EN_ LOW_CHP_BIAS_TX	(31) PLL_BANK_EN_ LOW_CHP_BIAS_TX	0x0	Set the en_low_chp_bias bit in Tx mode (banked)
		(30) PLL_BANK_EN_ LOW_CHP_BIAS_RX	(30) PLL_BANK_EN_ LOW_CHP_BIAS_RX	0x1	Set the en_low_chp_bias bit in Rx mode (banked)
		(29:28) PLL_BANK_ PLL_FILTER_RES_	(29:28) PLL_BANK_ PLL_FILTER_RES_	0x3	Modify the value of the loop filter resistor R2 when bit 5 is high in Tx

Address	Register Name	Register Write	Register Read	Default	Description
		TRIM_TX	TRIM_TX		mode (banked)
		(27:24) PLL_BANK_IQ_PLL_0_TX	(27:24) PLL_BANK_IQ_PLL_0_TX	0x4	Charge pump bias for Tx case (banked)
		(22) PLL_BANK_LOW_DR_TX	(22) PLL_BANK_LOW_DR_TX	0x0	Enable low data-rate mode in Tx mode (banked)
		(21:20) PLL_BANK_PLL_FILTER_RES_TRIM_RX	(21:20) PLL_BANK_PLL_FILTER_RES_TRIM_RX	0x0	Modify the value of the loop filter resistor R2 when bit 5 is high in Rx mode (banked)
		(19:16) PLL_BANK_IQ_PLL_0_RX	(19:16) PLL_BANK_IQ_PLL_0_RX	0xB	Charge pump bias for Rx (banked)
		(15) ANACLK_USE_NEW_ANACK	(15) ANACLK_USE_NEW_ANACK	0x0	Use the new analog clock generator (banked)
		(13:12) ANACLK_DIV_CK_RSSI	(13:12) ANACLK_DIV_CK_RSSI	0x0	Set the master clock divider for the RSSI clock (banked)
		(11:10) ANACLK_DIV_CK_FILT	(11:10) ANACLK_DIV_CK_FILT	0x0	Set the master clock divider for the channel filter clock (banked)
		(9:8) ANACLK_DIV_CK_PHADC	(9:8) ANACLK_DIV_CK_PHADC	0x0	Set the master clock divider for the phase ADC clock (banked)
		(7:4) ANACLK_DIV_RSSI	(7:4) ANACLK_DIV_RSSI	0x1	Unsigned value that specifies the division factor for the clock controlling the RSSI (banked)
		(3:0) ANACLK_DIV_FILT	(3:0) ANACLK_DIV_FILT	0x5	Unsigned value that specifies the division factor for the clock controlling the channel filter (banked)
0x40040A70	RF1_RSSI_CTRL	(31:30) RSSI_CTRL_AGC_DECAY_TAU	(31:30) RSSI_CTRL_AGC_DECAY_TAU	0x3	Time constant of the decay speed
		(29) RSSI_CTRL_AGC_USE_LNA	(29) RSSI_CTRL_AGC_USE_LNA	0x1	AGC algorithm uses LNA bias

Address	Register Name	Register Write	Register Read	Default	Description
		(28) RSSI_CTRL_AGC_MODE	(28) RSSI_CTRL_AGC_MODE	0x1	AGC algorithm selection
		(27:26) RSSI_CTRL_AGC_WAIT	(27:26) RSSI_CTRL_AGC_WAIT	0x3	Set the wait time of the AGC after switching between state
		(25) RSSI_CTRL_PAYLOAD_BLOCKS_AGC	(25) RSSI_CTRL_PAYLOAD_BLOCKS_AGC	0x1	AGC payload blocking
		(24) RSSI_CTRL_BYPASS_AGC	(24) RSSI_CTRL_BYPASS_AGC	0x0	AGC algorithm bypass
		(20:16) PA_PWR_OFFSET_PA_PWR_OFFSET	(20:16) PA_PWR_OFFSET_PA_PWR_OFFSET	0x0	Signed value that sets the PA power (banked)
		(12:8) FILTER_BIAS_IQ_FI_BW	(12:8) FILTER_BIAS_IQ_FI_BW	0x14	Bias for the bandwidth of the channel filter (banked)
		(4:0) FILTER_BIAS_IQ_FI_FC	(4:0) FILTER_BIAS_IQ_FI_FC	0xB	Bias for the central frequency of the channel filter (banked)
0x40040A74	RF1_REG1D	(31:28) AGC_PEAK_DET_PEAK_DET_TAU	(31:28) AGC_PEAK_DET_PEAK_DET_TAU	0x7	Time constant of the peak detector monostable circuit
		(27:26) AGC_PEAK_DET_PEAK_DET_THR_LOW	(27:26) AGC_PEAK_DET_PEAK_DET_THR_LOW	0x0	Threshold for the low level of the peak detector
		(25) AGC_PEAK_DET_PEAK_DET_THR_HIGH	(25) AGC_PEAK_DET_PEAK_DET_THR_HIGH	0x0	Threshold for the high level of the peak detector
		(24) AGC_PEAK_DET_EN_AGC_PEAK	(24) AGC_PEAK_DET_EN_AGC_PEAK	0x1	Enable AGC peak detector
		(23:16) AGC_THR_HIGH_AGC_THR_HIGH	(23:16) AGC_THR_HIGH_AGC_THR_HIGH	0x69	AGC threshold high level (banked)
		(15:8) AGC_THR_LOW_AGC_THR_LOW	(15:8) AGC_THR_LOW_AGC_THR_LOW	0x40	AGC threshold low level (banked)

Address	Register Name	Register Write	Register Read	Default	Description
		(7:4) ATT_CTRL_ATT_CTRL_MAX	(7:4) ATT_CTRL_ATT_CTRL_MAX	0xB	Maximum attenuation level in AGC algorithm
		(3:0) ATT_CTRL_SET_RX_ATT_CTRL	(3:0) ATT_CTRL_SET_RX_ATT_CTRL	0x0	Attenuation level if the AGC is bypassed
0x40040A78	RF1_AGC_LUT1	(31:22) AGC_LUT_1_AGC_LEVEL_2_LO	(31:22) AGC_LUT_1_AGC_LEVEL_2_LO	0x280	AGC values level 2 (LSB)
		(21:11) AGC_LUT_1_AGC_LEVEL_1	(21:11) AGC_LUT_1_AGC_LEVEL_1	0x80	AGC values level 1
		(10:0) AGC_LUT_1_AGC_LEVEL_0	(10:0) AGC_LUT_1_AGC_LEVEL_0	0x0	AGC values level 0
0x40040A7C	RF1_AGC_LUT2	(31:23) AGC_LUT_2_AGC_LEVEL_5_LO	(31:23) AGC_LUT_2_AGC_LEVEL_5_LO	0x84	AGC values level 5 (LSB)
		(22:12) AGC_LUT_2_AGC_LEVEL_4	(22:12) AGC_LUT_2_AGC_LEVEL_4	0x284	AGC values level 4
		(11:1) AGC_LUT_2_AGC_LEVEL_3	(11:1) AGC_LUT_2_AGC_LEVEL_3	0x480	AGC values level 3
		(0) AGC_LUT_2_AGC_LEVEL_2_HI	(0) AGC_LUT_2_AGC_LEVEL_2_HI	0x0	AGC values level 2 (MSB)
0x40040A80	RF1_AGC_LUT3	(31:24) AGC_LUT_3_AGC_LEVEL_8_LO	(31:24) AGC_LUT_3_AGC_LEVEL_8_LO	0x9D	AGC values level 8 (LSB)
		(23:13) AGC_LUT_3_AGC_LEVEL_7	(23:13) AGC_LUT_3_AGC_LEVEL_7	0x495	AGC values level 7
		(12:2) AGC_LUT_3_AGC_LEVEL_6	(12:2) AGC_LUT_3_AGC_LEVEL_6	0x485	AGC values level 6
		(1:0) AGC_LUT_3_AGC_LEVEL_5_HI	(1:0) AGC_LUT_3_AGC_LEVEL_5_HI	0x2	AGC values level 5 (MSB)
0x40040A84	RF1_AGC_LUT4	(31:25) AGC_LUT_4_AGC_LEVEL_11_LO	(31:25) AGC_LUT_4_AGC_LEVEL_11_LO	0x7F	AGC values level 11 (LSB)

Address	Register Name	Register Write	Register Read	Default	Description
		(24:14) AGC_LUT_4_AGC_LEVEL_10	(24:14) AGC_LUT_4_AGC_LEVEL_10	0x4FF	AGC values level 10
		(13:3) AGC_LUT_4_AGC_LEVEL_9	(13:3) AGC_LUT_4_AGC_LEVEL_9	0x49F	AGC values level 9
		(2:0) AGC_LUT_4_AGC_LEVEL_8_HI	(2:0) AGC_LUT_4_AGC_LEVEL_8_HI	0x4	AGC values level 8 (MSB)
0x40040A88	RF1_AGC_LUT5	(26:25) IEEE802154_OPTS_CNT_LIM_802154	(26:25) IEEE802154_OPTS_CNT_LIM_802154	0x2	Set the number of samples to wait before increasing the threshold
		(24:22) IEEE802154_OPTS_CNT_OK_INC_802154	(24:22) IEEE802154_OPTS_CNT_OK_INC_802154	0x4	Set the increment to the counter that indicates that the correlators peaks are coherent
		(21) IEEE802154_OPTS_USE_OS_802154	(21) IEEE802154_OPTS_USE_OS_802154	0x1	Enable the new algorithm working in the oversampled domain for the demodulation of the IEEE 802.15.4 protocol
		(20) IEEE802154_OPTS_EN_DW_TEST	(20) IEEE802154_OPTS_EN_DW_TEST	0x0	Tx data-whitening before the convolutional code block
		(18:16) IEEE802154_OPTS_C2B_THR	(18:16) IEEE802154_OPTS_C2B_THR	0x4	Threshold of the chip2bit correlator of the IEEE 802.15.4 protocol
		(13:12) DATA_STREAMS_BER_CLK_MODE	(13:12) DATA_STREAMS_BER_CLK_MODE	0x0	Set the clock output mode for BER mode or RW mode
		(10) DATA_STREAMS_RX_DATA_NOT_SAMPLED	(10) DATA_STREAMS_RX_DATA_NOT_SAMPLED	0x0	Signal rx_data in test modes sampling
		(9) DATA_STREAMS_PHASE_GREY	(9) DATA_STREAMS_PHASE_GREY	0x0	Phase signal encoding

Address	Register Name	Register Write	Register Read	Default	Description
		(8) DATA_STREAMS_ TX_IN_CLK_TOGGLE	(8) DATA_STREAMS_ TX_IN_CLK_TOGGLE	0x0	Input clock
		(3:0) AGC_LUT_5_ AGC_LEVEL_11_HI	(3:0) AGC_LUT_5_ AGC_LEVEL_11_HI	0xE	AGC values level 11 (MSB)
0x40040A8C	RF1_AGC_ATT1	(31:30) AGC_ATT_1_ AGC_ATT_AB_LO	(31:30) AGC_ATT_1_ AGC_ATT_AB_LO	0x3	AGC attenuation step 10/11 (LSB)
		(29:27) AGC_ATT_1_ AGC_ATT_9A	(29:27) AGC_ATT_1_ AGC_ATT_9A	0x5	AGC attenuation step 9/10
		(26:24) AGC_ATT_1_ AGC_ATT_89	(26:24) AGC_ATT_1_ AGC_ATT_89	0x3	AGC attenuation step 8/9
		(23:21) AGC_ATT_1_ AGC_ATT_78	(23:21) AGC_ATT_1_ AGC_ATT_78	0x4	AGC attenuation step 7/8
		(20:18) AGC_ATT_1_ AGC_ATT_67	(20:18) AGC_ATT_1_ AGC_ATT_67	0x3	AGC attenuation step 6/7
		(17:15) AGC_ATT_1_ AGC_ATT_56	(17:15) AGC_ATT_1_ AGC_ATT_56	0x2	AGC attenuation step 5/6
		(14:12) AGC_ATT_1_ AGC_ATT_45	(14:12) AGC_ATT_1_ AGC_ATT_45	0x2	AGC attenuation step 4/5
		(11:9) AGC_ATT_1_ AGC_ATT_34	(11:9) AGC_ATT_1_ AGC_ATT_34	0x2	AGC attenuation step 3/4
		(8:6) AGC_ATT_1_ AGC_ATT_23	(8:6) AGC_ATT_1_ AGC_ATT_23	0x1	AGC attenuation step 2/3
		(5:3) AGC_ATT_1_ AGC_ATT_12	(5:3) AGC_ATT_1_ AGC_ATT_12	0x1	AGC attenuation step 1/2
		(2:0) AGC_ATT_1_ AGC_ATT_01	(2:0) AGC_ATT_1_ AGC_ATT_01	0x4	AGC attenuation step 0/1
0x40040A90	RF1_AGC_ATT2	(31:28) TIMINGS_3_ T_DLL	(31:28) TIMINGS_3_ T_DLL	0x2	Time needed by the DLL blocks to switch on

Address	Register Name	Register Write	Register Read	Default	Description
		(27:24) TIMINGS_3_T_PLL_TX	(27:24) TIMINGS_3_T_PLL_TX	0x2	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	0xC	Time needed by the subband algorithm to calibrate in Tx mode
		(19:16) TIMINGS_2_T_TX_RF	(19:16) TIMINGS_2_T_TX_RF	0x1	Time needed by the RF blocks to switch on in Tx mode
		(14:12) TIMINGS_1_T_GRANULARITY_TX	(14:12) TIMINGS_1_T_GRANULARITY_TX	0x3	Define the granularity of the timer in Tx mode
		(10:8) TIMINGS_1_T_GRANULARITY_RX	(10:8) TIMINGS_1_T_GRANULARITY_RX	0x5	Define the granularity of the timer in Rx mode
		(1) AGC_ATT_2_AGC_ATT_1DB	(1) AGC_ATT_2_AGC_ATT_1DB	0x0	Attenuation steps
		(0) AGC_ATT_2_AGC_ATT_AB_HI	(0) AGC_ATT_2_AGC_ATT_AB_HI	0x1	AGC attenuation step 10/11 (MSB)
0x40040A94	RF1_REG25	(31) TIMEOUT_EN_RX_TIMEOUT	(31) TIMEOUT_EN_RX_TIMEOUT	0x0	Timeout of the Rx when the system is on FSM mode
		(30:28) TIMEOUT_T_TIMEOUT_GR	(30:28) TIMEOUT_T_TIMEOUT_GR	0x0	Granularity of the timer in timeout Rx mode
		(27:24) TIMEOUT_T_RX_TIMEOUT	(27:24) TIMEOUT_T_RX_TIMEOUT	0x0	Time that has to occur before the timeout
		(21) TIMING_FAST_RX_EN_FAST_RX_TXFILT	(21) TIMING_FAST_RX_EN_FAST_RX_TXFILT	0x0	Filter Tx configuration for the fast Rx PLL
		(20) TIMING_FAST_RX_EN_FAST_RX	(20) TIMING_FAST_RX_EN_FAST_RX	0x0	Fast Rx PLL
		(19:16) TIMING_FAST_RX_T_RX_FAST_CHP	(19:16) TIMING_FAST_RX_T_RX_FAST_CHP	0x0	Time to switch off the fast CHP in Rx mode

Address	Register Name	Register Write	Register Read	Default	Description
		(15:12) TIMINGS_5_T_RX_RF	(15:12) TIMINGS_5_T_RX_RF	0x0	Time needed by the RF blocks to switch on in Rx mode
		(11:8) TIMINGS_5_T_RX_BB	(11:8) TIMINGS_5_T_RX_BB	0x1	Time needed by the BB blocks to switch on in Rx mode
		(7:4) TIMINGS_4_T_SUBBAND_RX	(7:4) TIMINGS_4_T_SUBBAND_RX	0x5	Time needed by the subband algorithm to calibrate in Rx mode
		(3:0) TIMINGS_4_T_PLL_RX	(3:0) TIMINGS_4_T_PLL_RX	0x1	Time needed by the PLL blocks to switch on in Rx mode
0x40040A98	RF1_BIAS_0_2	(31:28) BIAS_2_IQ_RXTX_6	(31:28) BIAS_2_IQ_RXTX_6	0x3	VCOM_MX bias
		(27:24) BIAS_2_IQ_RXTX_5	(27:24) BIAS_2_IQ_RXTX_5	0x8	VCOM_LO bias
		(23:20) BIAS_1_IQ_RXTX_3	(23:20) BIAS_1_IQ_RXTX_3	0x6	PrePA Casc bias
		(19:16) BIAS_1_IQ_RXTX_2	(19:16) BIAS_1_IQ_RXTX_2	0x6	PrePA In bias
		(15:12) BIAS_0_IQ_RXTX_1	(15:12) BIAS_0_IQ_RXTX_1	0x7	PA backoff bias
		(11:8) BIAS_0_IQ_RXTX_0	(11:8) BIAS_0_IQ_RXTX_0	0x3	PA bias
		(7) INTERFACE_CONF_EN_SYNC_IFACE	(7) INTERFACE_CONF_EN_SYNC_IFACE	0x0	Interfaces resynchronization
		(6:4) INTERFACE_CONF_APB_WAIT_STATE	(6:4) INTERFACE_CONF_APB_WAIT_STATE	0x0	Select the number of wait states during the APB transaction
		(1:0) INTERFACE_CONF_SPI_SELECT	(1:0) INTERFACE_CONF_SPI_SELECT	0x0	Select the SPI mode
0x40040A9C	RF1_BIAS_3_6	(31:28) BIAS_6_IQ_	(31:28) BIAS_6_IQ_	0x7	ACD_O bias

Address	Register Name	Register Write	Register Read	Default	Description
		BB_0	BB_0		
		(27:24) BIAS_6_IQ_PLL_3	(27:24) BIAS_6_IQ_PLL_3	0x7	DLL bias
		(23:20) BIAS_5_IQ_PLL_4_RX	(23:20) BIAS_5_IQ_PLL_4_RX	0x8	VCO bias for Rx mode
		(19:16) BIAS_5_IQ_PLL_4_TX	(19:16) BIAS_5_IQ_PLL_4_TX	0xA	VCO bias for Tx mode
		(15:12) BIAS_4_IQ_PLL_2	(15:12) BIAS_4_IQ_PLL_2	0x7	Sub-band comparator bias
		(11:8) BIAS_4_IQ_PLL_1	(11:8) BIAS_4_IQ_PLL_1	0x4	Dynamic divider bias
		(7:4) BIAS_3_IQ_RXTX_8	(7:4) BIAS_3_IQ_RXTX_8	0x7	IFA ctrl_c bias
		(3:0) BIAS_3_IQ_RXTX_7	(3:0) BIAS_3_IQ_RXTX_7	0x7	IFA ctrl_r bias
0x40040AA0	RF1_BIAS_7_9	(31:28) BIAS_9_IQ_BB_6	(31:28) BIAS_9_IQ_BB_6	0x9	Peak detector threshold bias 0
		(27:24) BIAS_9_IQ_BB_5	(27:24) BIAS_9_IQ_BB_5	0x5	Peak detector bias
		(23:20) SWCAP_FSM_SB_CAP_RX	(23:20) SWCAP_FSM_SB_CAP_RX	0x0	VCO subband selection (FSM in Rx mode)
		(19:16) SWCAP_FSM_SB_CAP_TX	(19:16) SWCAP_FSM_SB_CAP_TX	0x0	VCO subband selection (FSM in Tx mode)
		(15:12) BIAS_8_IQ_BB_4	(15:12) BIAS_8_IQ_BB_4	0x9	RSSI_D bias
		(11:8) BIAS_8_IQ_BB_3	(11:8) BIAS_8_IQ_BB_3	0xF	RSSI_G bias
		(7:4) BIAS_7_IQ_BB_	(7:4) BIAS_7_IQ_	0x6	ACD_L bias

Address	Register Name	Register Write	Register Read	Default	Description
		2	BB_2		
		(3:0) BIAS_7_IQ_BB_1	(3:0) BIAS_7_IQ_BB_1	0x6	ACD_C bias
0x40040AA4	RF1_BIAS_10_12	(30) SD_MASH_MASH_DITHER_TYPE	(30) SD_MASH_MASH_DITHER_TYPE	0x0	Enable the new dithering scheme
		(29) SD_MASH_MASH_ENABLE	(29) SD_MASH_MASH_ENABLE	0x0	Enable the sigma delta mash
		(28) SD_MASH_MASH_DITHER	(28) SD_MASH_MASH_DITHER	0x1	Enable dithering on the sigma delta mash
		(27:25) SD_MASH_MASH_ORDER	(27:25) SD_MASH_MASH_ORDER	0x3	Order of the sigma delta mash
		(24) SD_MASH_MASH_RSTB	(24) SD_MASH_MASH_RSTB	0x1	Reset of the sigma delta mash (active low)
		(23:20) BIAS_12_LNA_AGC_BIAS_3	(23:20) BIAS_12_LNA_AGC_BIAS_3	0x6	LNA bias for AGC level 3
		(19:16) BIAS_12_LNA_AGC_BIAS_2	(19:16) BIAS_12_LNA_AGC_BIAS_2	0x7	LNA bias for AGC level 2
		(15:12) BIAS_11_LNA_AGC_BIAS_1	(15:12) BIAS_11_LNA_AGC_BIAS_1	0x8	LNA bias for AGC level 1
		(11:8) BIAS_11_LNA_AGC_BIAS_0	(11:8) BIAS_11_LNA_AGC_BIAS_0	0x9	LNA bias for AGC level 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	0x6	Peak detector threshold bias 2
0x40040AA8	RF1_REG2A	(27:24) SD_MASH_MASK_MASH_MASK	(27:24) SD_MASH_MASK_MASH_MASK	0x0	Mask the n LSB of the fractional part of the MASH (debug only)
		(19) BIAS_EN_2_EN_	(19) BIAS_EN_2_EN_	0x1	Enable PTAT

Address	Register Name	Register Write	Register Read	Default	Description
		PTAT	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)
0x40040AAC	RF1_PLL_CTRL	(26) PLL_CTRL_DISABLE_CHP_SBS	(26) PLL_CTRL_DISABLE_CHP_SBS	0x0	Charge-pump disabling during sub-band selection (FLL and frequency ratios)
		(25) PLL_CTRL_PLL_RX_48MEG	(25) PLL_CTRL_PLL_RX_48MEG	0x1	PLL frequency
		(24) PLL_CTRL_SWCAP_TX_SAME_RX	(24) PLL_CTRL_SWCAP_TX_SAME_RX	0x0	Registers for Rx and Tx modes swcap in case of swcap_fsm=1
		(23) PLL_CTRL_SWCAP_FSM	(23) PLL_CTRL_SWCAP_FSM	0x1	Selection of the swcap_fsm register
		(22) PLL_CTRL_DLL_RSTB	(22) PLL_CTRL_DLL_RSTB	0x1	Reset signal of the DLL (active low)
		(21:18) PLL_CTRL_VCO_SUBBAND_TRIM	(21:18) PLL_CTRL_VCO_SUBBAND_TRIM	0x0	VCO sub-band selection bits
		(17) PLL_CTRL_SUB_SEL_OFFS_EN	(17) PLL_CTRL_SUB_SEL_OFFS_EN	0x0	Add offset to sub-band selection comparator
		(16) PLL_CTRL_DIV2_CLKVCO_TEST_EN	(16) PLL_CTRL_DIV2_CLKVCO_TEST_EN	0x0	VCO signal divided by the programmable divider
		(15) PLL_CTRL_	(15) PLL_CTRL_	0x0	Output on GPIO the VCO signal

Address	Register Name	Register Write	Register Read	Default	Description
		VCODIV_CLK_TEST_EN	VCODIV_CLK_TEST_EN		divided by the programmable divider
		(13) PLL_CTRL_CHP_DEAD_ZONE_EN	(13) PLL_CTRL_CHP_DEAD_ZONE_EN	0x0	Charge-pump dead zone
		(12:11) PLL_CTRL_CHP_CURR_OFF_TRIM_TX	(12:11) PLL_CTRL_CHP_CURR_OFF_TRIM_TX	0x3	Charge-pump offset current values selection bits in Tx mode
		(10:9) PLL_CTRL_CHP_CURR_OFF_TRIM_RX	(10:9) PLL_CTRL_CHP_CURR_OFF_TRIM_RX	0x3	Charge-pump offset current values selection bits in Rx mode
		(8) PLL_CTRL_HIGH_BW_FILTER_EN_TX	(8) PLL_CTRL_HIGH_BW_FILTER_EN_TX	0x1	PLL filter high bandwidth needed in Tx mode
		(7) PLL_CTRL_HIGH_BW_FILTER_EN_RX	(7) PLL_CTRL_HIGH_BW_FILTER_EN_RX	0x0	PLL filter high bandwidth needed in Rx mode
		(6) PLL_CTRL_FAST_CHP_EN_TX	(6) PLL_CTRL_FAST_CHP_EN_TX	0x1	High current output of the charge-pump for PLL Tx high bandwidth mode
		(5) PLL_CTRL_FAST_CHP_EN_RX	(5) PLL_CTRL_FAST_CHP_EN_RX	0x0	High current output of the charge-pump for PLL Rx high bandwidth mode
		(4:3) PLL_CTRL_CHP_MODE_TRIM	(4:3) PLL_CTRL_CHP_MODE_TRIM	0x0	Select the frequency inside sub-band selection
		(2) PLL_CTRL_CHP_CMC_EN	(2) PLL_CTRL_CHP_CMC_EN	0x1	Common mode control block of the charge-pump
		(1) PLL_CTRL_CHP_CURR_OFF_EN_TX	(1) PLL_CTRL_CHP_CURR_OFF_EN_TX	0x1	Charge-pump offset current in Tx mode
		(0) PLL_CTRL_CHP_CURR_OFF_EN_RX	(0) PLL_CTRL_CHP_CURR_OFF_EN_RX	0x0	Charge-pump offset current in Rx mode
0x40040AB0	RF1_DLL_CTRL	(31:29) RSSI_TUN_1_RSSI_TUN_GAIN	(31:29) RSSI_TUN_1_RSSI_TUN_GAIN	0x1	RSSI tuning for gain
		(28:24) RSSI_TUN_1_	(28:24) RSSI_TUN_	0x4	RSSI tuning for odd stages (offset to

Address	Register Name	Register Write	Register Read	Default	Description
		RSSI_ODD_OFFSET	1_RSSI_ODD_OFFSET		the even triangular wave)
		(23:20) RSSI_TUN_1_ RSSI_EVEN_MAX	(23:20) RSSI_TUN_ 1_RSSI_EVEN_MAX	0x1	RSSI tuning for even stages (maximum value of the triangular wave)
		(19:16) RSSI_TUN_1_ RSSI_EVEN_MIN	(19:16) RSSI_TUN_ 1_RSSI_EVEN_MIN	0x1	RSSI tuning for even stages (minimum value of the triangular wave)
		(12) DLL_CTRL_CK_ LAST_SEL_DELAY	(12) DLL_CTRL_CK_ LAST_SEL_DELAY	0x0	Last SEL delay
		(11) DLL_CTRL_CK_ FIRST_SEL_DELAY	(11) DLL_CTRL_CK_ FIRST_SEL_DELAY	0x0	First SEL delay
		(10) DLL_CTRL_CK_ EXT_SEL	(10) DLL_CTRL_CK_ EXT_SEL	0x0	Input clock selection
		(9) DLL_CTRL_CK_ DIG_EN	(9) DLL_CTRL_CK_ DIG_EN	0x0	Alternate ck_dig pin to output the PLL reference clock signal
		(8) DLL_CTRL_CK_ TEST_EN	(8) DLL_CTRL_CK_ TEST_EN	0x0	Output on GPIO the PLL reference clock signal via ck_test pin
		(7) DLL_CTRL_TOO_ FAST_ENB	(7) DLL_CTRL_TOO_ FAST_ENB	0x0	Lock range phase detector
		(6) DLL_CTRL_ LOCKED_DET_EN	(6) DLL_CTRL_ LOCKED_DET_EN	0x1	Reference frequency multiplier locked detector
		(5) DLL_CTRL_ LOCKED_AUTO_CHECK_ EN	(5) DLL_CTRL_ LOCKED_AUTO_CHECK_ EN	0x1	Frequency multiplier is out of lock (usually because some input clocks from ck_xtal or ck_ext are missing)
		(4) DLL_CTRL_FAST_ ENB	(4) DLL_CTRL_FAST_ ENB	0x0	Disable fast mode locking of the reference frequency multiplier
		(3:2) DLL_CTRL_CK_ SEL_TX	(3:2) DLL_CTRL_CK_ SEL_TX	0x1	Selection of the clock used as frequency reference of the PLL in Tx mode (also to ck_test and ck_dig outputs)

Address	Register Name	Register Write	Register Read	Default	Description
		(1:0) DLL_CTRL_CK_SEL_RX	(1:0) DLL_CTRL_CK_SEL_RX	0x0	Selection of the clock used as frequency reference of the PLL in Rx mode (also to ck_test and ck_dig outputs)
0x40040AB4	RF1_REG2D	(29:28) PA_CONF_SW_CN	(29:28) PA_CONF_SW_CN	0x0	Harmonic 2 notch tuning
		(27) PA_CONF_TX_SWITCHPA	(27) PA_CONF_TX_SWITCHPA	0x0	Switch PA
		(26) PA_CONF_TX_0DBM	(26) PA_CONF_TX_0DBM	0x1	Select between PPA and PA
		(25) PA_CONF_LIN_RAMP	(25) PA_CONF_LIN_RAMP	0x0	PA ramp-up linearization
		(24) PA_CONF_MIN_PA_PWR	(24) PA_CONF_MIN_PA_PWR	0x1	Set the minimum power during the PA ramp-up
		(23) CTRL_RX_SWITCH_LP	(23) CTRL_RX_SWITCH_LP	0x0	Switch the low-pass filter in the Rx chain
		(22) CTRL_RX_USE_PEAK_DETECTOR	(22) CTRL_RX_USE_PEAK_DETECTOR	0x1	Peak detector powering
		(21) CTRL_RX_START_MIX_ON_CAL	(21) CTRL_RX_START_MIX_ON_CAL	0x0	Mixer enabling
		(20:16) CTRL_RX_CTRL_RX	(20:16) CTRL_RX_CTRL_RX	0xF	Rx control
		(15) CTRL_ADC_PHADC_THERM_OUT_EN	(15) CTRL_ADC_PHADC_THERM_OUT_EN	0x1	Enable the buffers of phase ADC thermometric code (banked)
		(14:13) CTRL_ADC_PHADC_DELLATCH	(14:13) CTRL_ADC_PHADC_DELLATCH	0x1	Phase ADC delay latch trimming (banked)
		(12:8) CTRL_ADC_CTRL_ADC	(12:8) CTRL_ADC_CTRL_ADC	0x5	Phase ADC control (banked)

Address	Register Name	Register Write	Register Read	Default	Description
		(6:5) RSSI_TUN_2_ RSSI_TRI_CK_DIV	(6:5) RSSI_TUN_2_ RSSI_TRI_CK_DIV	0x0	Speed on the RSSI triangular dithering signal (cf reg RSSI_TUN)
		(4) RSSI_TUN_2_ RSSI_ONE_CK_RSSI_ PHADC	(4) RSSI_TUN_2_ RSSI_ONE_CK_RSSI_ PHADC	0x0	RSSI and phase ADC clocks sharing
		(3) RSSI_TUN_2_ RSSI_FULL	(3) RSSI_TUN_2_ RSSI_FULL	0x1	RSSI full scale
		(2) RSSI_TUN_2_ RSSI_1DB	(2) RSSI_TUN_2_ RSSI_1DB	0x0	LSB resolution
		(1:0) RSSI_TUN_2_ RSSI_PRE_ATT	(1:0) RSSI_TUN_2_ RSSI_PRE_ATT	0x3	Pre attenuation of the RSSI signal
0x40040AB8	RF1_REG2E	(31:24) XTAL_TRIM_ XTAL_TRIM_INIT	(31:24) XTAL_TRIM_ XTAL_TRIM_INIT	0x60	Initial trimming of the XTAL
		(23:16) XTAL_TRIM_ XTAL_TRIM	(23:16) XTAL_TRIM_ XTAL_TRIM	0x60	Trimming of the XTAL
		(12) ENABLES_ SEPARATE_PPA_CASC	(12) ENABLES_ SEPARATE_PPA_CASC	0x0	PA cascode bit
		(11:6) ENABLES_EN_ RXTX	(11:6) ENABLES_EN_ RXTX	0x0	Enable signals
		(5:0) ENABLES_EN_BB	(5:0) ENABLES_EN_ BB	0x0	Enable signals for the BB
0x40040ABC	RF1_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

Address	Register Name	Register Write	Register Read	Default	Description
		(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
		(19) XTAL_CTRL_LOW_CLK_READY_TH_EN	(19) XTAL_CTRL_LOW_CLK_READY_TH_EN	0x0	clk_ready threshold
		(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	Clock selection for the digital block
		(16) XTAL_CTRL_XO_EN_B_REG	(16) XTAL_CTRL_XO_EN_B_REG	0x1	XTAL oscillator enable
		(15:14) XTAL_CTRL_XTAL_CKDIV	(15:14) XTAL_CTRL_XTAL_CKDIV	0x0	XTAL trimming speed
		(13) XTAL_CTRL_CLK_OUT_EN_B	(13) XTAL_CTRL_CLK_OUT_EN_B	0x0	Output clock to go to main IP
		(12) XTAL_CTRL_REG_VALUE_SEL	(12) XTAL_CTRL_REG_VALUE_SEL	0x0	Control bits of xtal_reg
		(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
		(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
		(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_	(5) XTAL_CTRL_	0x0	Force output clocks on clk_pll, clk_dig

Address	Register Name	Register Write	Register Read	Default	Description
		FORCE_CLK_READY	FORCE_CLK_READY		and clk_out
		(4) XTAL_CTRL_CLK_DIG_EN_B	(4) XTAL_CTRL_CLK_DIG_EN_B	0x0	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	XTAL buffer disabling
		(2) XTAL_CTRL_HP_MODE	(2) XTAL_CTRL_HP_MODE	0x0	Bias current increase in the clock buffer
		(1) XTAL_CTRL_LP_MODE	(1) XTAL_CTRL_LP_MODE	0x0	Bias current decrease in the clock buffer
		(0) XTAL_CTRL_EXT_CLK_MODE	(0) XTAL_CTRL_EXT_CLK_MODE	0x0	Use XTAL pads as external clock input
0x40040AC0	RF1_SUBBAND	(31:24) SUBBAND_OFFSET_SB_OFFSET_RX	(31:24) SUBBAND_OFFSET_SB_OFFSET_RX	0xF1	Offset to add in frequency count in order to compensate the offset of the varicap
		(23:16) SUBBAND_OFFSET_SB_OFFSET	(23:16) SUBBAND_OFFSET_SB_OFFSET	0xD0	Offset to add in frequency count in order to compensate the offset of the varicap
		(15:12) SWCAP_LIM_SB_MAX_VAL	(15:12) SWCAP_LIM_SB_MAX_VAL	0xF	Maximum subband value in linear search subband (freq and comp)
		(11:8) SWCAP_LIM_SB_MIN_VAL	(11:8) SWCAP_LIM_SB_MIN_VAL	0x0	Minimum subband value in linear search subband (freq and comp)
		(7) SUBBAND_CONF_SB_FLL_MODE	(7) SUBBAND_CONF_SB_FLL_MODE	0x1	FLL mode for the subband selection
		(6) SUBBAND_CONF_SB_INV_BAND	(6) SUBBAND_CONF_SB_INV_BAND	0x0	Invert the meaning of sb_high and sb_low
		(5:4) SUBBAND_CONF_SB_FREQ_CNT	(5:4) SUBBAND_CONF_SB_FREQ_CNT	0x0	The length to count in frequency mode
		(3:2) SUBBAND_CONF_	(3:2) SUBBAND_	0x0	Time to wait to the PLL to settle

Address	Register Name	Register Write	Register Read	Default	Description
		SB_WAIT_T	CONF_SB_WAIT_T		
		(1:0) SUBBAND_CONF_SB_MODE	(1:0) SUBBAND_CONF_SB_MODE	0x0	Sub-band algorithm mode
0x40040AC4	RF1_REG31	(31:30) RSSI_DETECT_RSSI_DET_CR_LEN	(31:30) RSSI_DETECT_RSSI_DET_CR_LEN	0x0	Number of samples to estimate the carrier offset (banked)
		(29:28) RSSI_DETECT_RSSI_DET_WAIT	(29:28) RSSI_DETECT_RSSI_DET_WAIT	0x0	Symbols to wait after the RSSI detection (banked)
		(27:26) RSSI_DETECT_RSSI_DET_DIFF_LL	(27:26) RSSI_DETECT_RSSI_DET_DIFF_LL	0x0	Set the distance between the actual value and the subtracted one (banked)
		(25) RSSI_DETECT_EN_ABS_RSSI_DETECT	(25) RSSI_DETECT_EN_ABS_RSSI_DETECT	0x0	Absolute RSSI detection (banked)
		(24) RSSI_DETECT_EN_DIFF_RSSI_DETECT	(24) RSSI_DETECT_EN_DIFF_RSSI_DETECT	0x0	Differential RSSI detection (banked)
		(23) SUBBAND_CORR_SUBBAND_CORR_EN	(23) SUBBAND_CORR_SUBBAND_CORR_EN	0x0	Subband correction
		(22:20) SUBBAND_CORR_SUBBAND_CORR_RX	(22:20) SUBBAND_CORR_SUBBAND_CORR_RX	0x0	Subband correction in Rx
		(18:16) SUBBAND_CORR_SUBBAND_CORR_TX	(18:16) SUBBAND_CORR_SUBBAND_CORR_TX	0x0	Subband correction in Tx
		(11) TXRX_CONF_INV_CLK_PLL_TX	(11) TXRX_CONF_INV_CLK_PLL_TX	0x0	Invert PLL clock when the radio is in Tx mode
		(10) TXRX_CONF_INV_CLK_DIG_TX	(10) TXRX_CONF_INV_CLK_DIG_TX	0x0	Invert digital clock when the radio is in Tx mode

Address	Register Name	Register Write	Register Read	Default	Description
		(9:8) TXRX_CONF_SB_WAIT_T_TX	(9:8) TXRX_CONF_SB_WAIT_T_TX	0x0	Xor value to apply to sb_wait_t (register SUBBAND_CONF) when the radio is in Tx mode
		(7) PA_RAMPUP_FULL_PA_RAMPUP	(7) PA_RAMPUP_FULL_PA_RAMPUP	0x1	PA rampup configuration
		(6:4) PA_RAMPUP_DEL_PA_RAMPUP	(6:4) PA_RAMPUP_DEL_PA_RAMPUP	0x4	Time to wait to start the ramp-up after the PA enable is detected
		(3:2) PA_RAMPUP_TAU_PA_RAMPUP	(3:2) PA_RAMPUP_TAU_PA_RAMPUP	0x0	Time constant of the ramp-up/ramp-down
		(1) PA_RAMPUP_EN_PA_RAMPDOWN	(1) PA_RAMPUP_EN_PA_RAMPDOWN	0x1	PA ramp-down
		(0) PA_RAMPUP_EN_PA_RAMPUP	(0) PA_RAMPUP_EN_PA_RAMPUP	0x1	PA ramp-up linearization
0x40040AC8	RF1_DEMOD_CTRL	(31) SYNC_WORD_CORR_EN_SYNC_WORD_CORR	(31) SYNC_WORD_CORR_EN_SYNC_WORD_CORR	0x1	Sync word bias correction with RSSI detection (banked)
		(29:24) SYNC_WORD_CORR_SYNC_WORD_BIAS	(29:24) SYNC_WORD_CORR_SYNC_WORD_BIAS	0x8	Set the sync word bias (banked)
		(23:16) RSSI_DETECT_ABS_THR_RSSI_DET_ABS_THR	(23:16) RSSI_DETECT_ABS_THR_RSSI_DET_ABS_THR	0x0	Threshold used for absolute RSSI detection
		(15:8) RSSI_DETECT_DIFF_THR_RSSI_DET_DIFF_THR	(15:8) RSSI_DETECT_DIFF_THR_RSSI_DET_DIFF_THR	0x0	Threshold used for differential RSSI detection
		(7) DEMOD_CTRL_DL_SYNC_NO_DATA	(7) DEMOD_CTRL_DL_SYNC_NO_DATA	0x1	No data going through the demodulator, until the delay line detects the sync word (banked)
		(6) DEMOD_CTRL_EN_	(6) DEMOD_CTRL_EN_	0x1	Sync word detection in the delay line

Address	Register Name	Register Write	Register Read	Default	Description
		DELLINE_SYNC_DET	DELLINE_SYNC_DET		(banked)
		(5) DEMOD_CTRL_ RSSI_DET_FILT	(5) DEMOD_CTRL_ RSSI_DET_FILT	0x0	Additional filtering on the RSSI value (banked)
		(4) DEMOD_CTRL_EN_ FAST_CLK_RECOV	(4) DEMOD_CTRL_EN_ FAST_CLK_RECOV	0x0	Clock recovery during the resto of the preamble (banked)
		(3) DEMOD_CTRL_EN_ MIN_MAX_MF	(3) DEMOD_CTRL_EN_ MIN_MAX_MF	0x0	Min max algo after the matched filter (banked)
		(2) DEMOD_CTRL_EN_ PRE_SYNC	(2) DEMOD_CTRL_EN_ PRE_SYNC	0x0	Sync detection on the non-delayed path (banked)
		(1) DEMOD_CTRL_ BLOCK_RSSI_DET	(1) DEMOD_CTRL_ BLOCK_RSSI_DET	0x0	RSSI detection during the slow-down period (banked)
		(0) DEMOD_CTRL_ EARLY_FINE_RECOV	(0) DEMOD_CTRL_ EARLY_FINE_RECOV	0x0	Early fine recovery after the packet detection or pre-sync (banked)
0x40040ACC	RF1_REG33	(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
		(23:16) SPARES_ SPARES	(23:16) SPARES_ SPARES	0x0	Spare bits
		(14) PADS_PE_DS_ GPIO_DS	(14) PADS_PE_DS_ GPIO_DS	0x0	Increased drive strength of the digital pads
		(13) PADS_PE_DS_ GPIO_PE	(13) PADS_PE_DS_ GPIO_PE	0x0	Pull-up of the GPIO pads
		(12) PADS_PE_DS_ NRESET_PE	(12) PADS_PE_DS_ NRESET_PE	0x0	Pull-up of the NRESET pads
		(11) PADS_PE_DS_ SPI_MISO_PE	(11) PADS_PE_DS_ SPI_MISO_PE	0x0	Pull-up of the SPI MISO pads
		(10) PADS_PE_DS_ SPI_MOSI_PE	(10) PADS_PE_DS_ SPI_MOSI_PE	0x0	Pull-up of the SPI MOSI pads
		(9) PADS_PE_DS_SPI_	(9) PADS_PE_DS_	0x0	Pull-up of the SPI CLK pads

Address	Register Name	Register Write	Register Read	Default	Description
		SCLK_PE	SPI_SCLK_PE		
		(8) PADS_PE_DS_SPI_CS_N_PE	(8) PADS_PE_DS_SPI_CS_N_PE	0x0	Pull-up of the SPI CSN pads
		(7:6) SUBBAND_FLL_SB_FLL_DITHER	(7:6) SUBBAND_FLL_SB_FLL_DITHER	0x0	Select the dithering
		(5:4) SUBBAND_FLL_SB_FLL_CIC_TAU	(5:4) SUBBAND_FLL_SB_FLL_CIC_TAU	0x3	Set the CIC decimator factor
		(3) SUBBAND_FLL_SB_FLL_PH_4_N8	(3) SUBBAND_FLL_SB_FLL_PH_4_N8	0x0	Phases in the frequency detector
		(2:0) SUBBAND_FLL_SB_FLL_WAIT	(2:0) SUBBAND_FLL_SB_FLL_WAIT	0x3	Set the number of CIC samples before stopping the FLL
0x40040AD0	RF1_REG34	(29:24) CLK_RECOVERY_CLK_RECOV_CORR	(29:24) CLK_RECOVERY_CLK_RECOV_CORR	0x4	Number of samples that covers the clock recovery correlator
		(23:16) CLK_RECOVERY_CLK_AB_LIMIT	(23:16) CLK_RECOVERY_CLK_AB_LIMIT	0x80	Time constant for switch the clock phase if chosen wrong in clk recovery algorithm
		(15) TX_PRE_DIST_EN_PRE_DIST	(15) TX_PRE_DIST_EN_PRE_DIST	0x1	Tx pre-distortion filter (banked)
		(13:8) TX_PRE_DIST_PRE_DIST_B0	(13:8) TX_PRE_DIST_PRE_DIST_B0	0x2E	Coefficient b0 of the Tx pre-distortion filter (banked)
		(5:0) TX_PRE_DIST_PRE_DIST_A0	(5:0) TX_PRE_DIST_PRE_DIST_A0	0x2F	Coefficient a0 of the Tx pre-distortion filter (banked)
0x40040AD4	RF1_BLE_LR	(30:24) BLR_SYNC_THRESHOLD_BLE_SYNC_THR	(30:24) BLR_SYNC_THRESHOLD_BLE_SYNC_THR	0x38	Threshold for the BLR sync word detector
		(19:16) BLR_PREAMBLE_BLE_PRE_THR	(19:16) BLR_PREAMBLE_BLE_PRE_THR	0x1	Threshold for the BLR preamble detector

Address	Register Name	Register Write	Register Read	Default	Description
		(15) BLE_LONG_ RANGE_BLR_PUT_RI_ FIFO	(15) BLE_LONG_ RANGE_BLR_PUT_RI_ FIFO	0x1	During the reception the RI (rate indicator) is put into the Rx FIFO (banked)
		(14) BLE_LONG_ RANGE_BLR500_NO_ ROUGH	(14) BLE_LONG_ RANGE_BLR500_NO_ ROUGH	0x1	Rough recovery is stopped during the 500kbps payloads of BLR packets (banked)
		(13) BLE_LONG_ RANGE_BLR_LIN_ FILTER	(13) BLE_LONG_ RANGE_BLR_LIN_ FILTER	0x1	Matched filter (banked)
		(12) BLE_LONG_ RANGE_EN_BLR_FLUSH	(12) BLE_LONG_ RANGE_EN_BLR_FLUSH	0x1	Viterbi path 0 flushing at the end of the packet (banked)
		(11) BLE_LONG_ RANGE_BLR_USE_EXT_ LEN	(11) BLE_LONG_ RANGE_BLR_USE_EXT_ LEN	0x0	BLR_PKT_LEN for flushing out the Viterbi (banked)
		(10) BLE_LONG_ RANGE_DISABLE_BLR_ TX	(10) BLE_LONG_ RANGE_DISABLE_BLR_ TX	0x0	Long Range feature in Tx mode (banked)
		(9) BLE_LONG_RANGE_ BLR_500_N125	(9) BLE_LONG_ RANGE_BLR_500_N125	0x0	Data rate selection (banked)
		(8) BLE_LONG_RANGE_ EN_BLR	(8) BLE_LONG_ RANGE_EN_BLR	0x0	BLE long range mode (banked)
		(4) HW_TRIGGER_HW_ TRIG_GPIO	(4) HW_TRIGGER_HW_ TRIG_GPIO	0x0	HW trigger is mapped on the GPIO instead of the Tx_on signal 0x0
		(3) HW_TRIGGER_HW_ TRIG_SUBBAND	(3) HW_TRIGGER_HW_ TRIG_SUBBAND	0x0	Activate the sub-band selection during the Tx activation
		(2) HW_TRIGGER_HW_ TRIG_TX_NRX	(2) HW_TRIGGER_HW_ TRIG_TX_NRX	0x0	Activate the Tx mode
		(1) HW_TRIGGER_HW_ TRIG_LOW	(1) HW_TRIGGER_HW_ TRIG_LOW	0x0	Set the trigger polarity

Address	Register Name	Register Write	Register Read	Default	Description
		(0) HW_TRIGGER_HW_TRIG_ACTIVE	(0) HW_TRIGGER_HW_TRIG_ACTIVE	0x0	Enable HW trigger
0x40040AD8	RF1_REG36	(30:28) IQ_SPARES_EN_BIAS_SPARE	(30:28) IQ_SPARES_EN_BIAS_SPARE	0x0	Enable for IQ spares
		(27:24) IQ_SPARES_IQ_SPARE_2	(27:24) IQ_SPARES_IQ_SPARE_2	0x0	Spare bias 2
		(23:20) IQ_SPARES_IQ_SPARE_1	(23:20) IQ_SPARES_IQ_SPARE_1	0x0	Spare bias 1
		(19:16) IQ_SPARES_IQ_SPARE_0	(19:16) IQ_SPARES_IQ_SPARE_0	0x0	Spare bias 0
		(8) MISC_ISO_VDDA	(8) MISC_ISO_VDDA	0x0	Isolate VDDA signals
		(5) BLR_DEMAPPER_BLR_SEND_DECODED_RI	(5) BLR_DEMAPPER_BLR_SEND_DECODED_RI	0x0	Fully decode the rate indicator
		(4) BLR_DEMAPPER_BLR_USE_EXT_VIT_GFSK	(4) BLR_DEMAPPER_BLR_USE_EXT_VIT_GFSK	0x1	500kbps BLR uses the Viterbi GFSK decision
		(3:2) BLR_DEMAPPER_BLR_500_DPHASE	(3:2) BLR_DEMAPPER_BLR_500_DPHASE	0x3	Set the distance between samples for the phase to frequency conversion in S2 mode
		(1) BLR_DEMAPPER_BLR_500_LOW_GAIN	(1) BLR_DEMAPPER_BLR_500_LOW_GAIN	0x0	Set the low gain in S2 mode
		(0) BLR_DEMAPPER_BLR_125_LOW_GAIN	(0) BLR_DEMAPPER_BLR_125_LOW_GAIN	0x0	Set the low gain in S8 mode
0x40040ADC	RF1_PROT_TIMER	(31) PROT_TIMER_CONF_EN_PROT_TIMER	(31) PROT_TIMER_CONF_EN_PROT_TIMER	0x0	Enable the protocol timer
		(29:27) PROT_TIMER_CONF_PT_T_STP_1	(29:27) PROT_TIMER_CONF_PT_T_STP_1	0x0	Configure the time stamp 1

Address	Register Name	Register Write	Register Read	Default	Description
		(26:24) PROT_TIMER_CONF_PT_T_STP_0	(26:24) PROT_TIMER_CONF_PT_T_STP_0	0x0	Configure the time stamp 0
		(22) STAGING_PS_NZ_START_BIT	(22) STAGING_PS_NZ_START_BIT	0x0	Select the frequency offset
		(21) STAGING_PS_NZ_START	(21) STAGING_PS_NZ_START	0x0	Start the pulse shaper with a +/- 250 kHz frequency offset
		(20) STAGING_DEL_PA_RAMPDW	(20) STAGING_DEL_PA_RAMPDW	0x0	Delay the PA ramp-down by 4.5 us
		(19) STAGING_PEAK_DET_TH_SHIFT	(19) STAGING_PEAK_DET_TH_SHIFT	0x0	Peak detector threshold shift
		(18:17) STAGING_AGC_DERIV_LVL	(18:17) STAGING_AGC_DERIV_LVL	0x2	Select the AGC derivative level
		(16) STAGING_AGC_USE_DERIV	(16) STAGING_AGC_USE_DERIV	0x0	AGC algorithm uses the derivative information to accelerate the AGC settling
		(15:8) BLE_DTM_BLE_DTM_LEN	(15:8) BLE_DTM_BLE_DTM_LEN	0x25	Set the BLE DTM packet length
		(7) BLE_DTM_EN_BLE_DTM	(7) BLE_DTM_EN_BLE_DTM	0x0	Enable the BLE DTM automatic packets
		(3:0) BLE_DTM_BLE_DTM_PKT_TYPE	(3:0) BLE_DTM_BLE_DTM_PKT_TYPE	0x0	Set the BLE DTM packet type (see Bluetooth specification)
0x40040AE0	RF1_CTE_OPTS	(29) CTE_OPTS_RECT_PS_CTE	(29) CTE_OPTS_RECT_PS_CTE	0x0	Use rectangular pulse shape during the CTE
		(28) CTE_OPTS_USE_CTE_WO_CP	(28) CTE_OPTS_USE_CTE_WO_CP	0x0	Enable the CTE without reading or inserting the CP
		(27) CTE_OPTS_CTE_AMPL	(27) CTE_OPTS_CTE_AMPL	0x0	Enable the usage of the RSSI values to adapt the amplitude of the IQ signal based to the RSSI value

Address	Register Name	Register Write	Register Read	Default	Description
		(26) CTE_OPTS_DF_AOA_SLOT_TIME	(26) CTE_OPTS_DF_AOA_SLOT_TIME	0x0	Indicate the switching/sampling slot period for AoA
		(25) CTE_OPTS_CP_INSERT	(25) CTE_OPTS_CP_INSERT	0x0	Force the CP bit in the packet header to 1
		(24) CTE_OPTS_EN_READ_CP	(24) CTE_OPTS_EN_READ_CP	0x0	CP bit is read in the packet header (BLE standard)
		(23:16) CTE_OPTS_CTE_INFO	(23:16) CTE_OPTS_CTE_INFO	0x0	Set the CTEInfo field in the packet header while cp_insert is set to 1
		(14:10) ASK_MOD_ASK_MAX	(14:10) ASK_MOD_ASK_MAX	0xC	Set the maximum value for the ASK modulation
		(9:5) ASK_MOD_ASK_MIN	(9:5) ASK_MOD_ASK_MIN	0x0	Set the minimum value for the ASK modulation
		(4:1) ASK_MOD_ASK_CNT	(4:1) ASK_MOD_ASK_CNT	0x7	Set the how long to count for the ASK modulation
		(0) ASK_MOD_EN_RSSI_ASK	(0) ASK_MOD_EN_RSSI_ASK	0x0	PA will perform an ASK modulation
0x40040AE4	RF1_PT_DELTA_0	(31:30) PT_DELTA_TS_0_PT_DELTA_T0_MULT	(31:30) PT_DELTA_TS_0_PT_DELTA_T0_MULT	0x0	Multiplier for the delta t0
		(19:0) PT_DELTA_TS_0_PT_DELTA_T0	(19:0) PT_DELTA_TS_0_PT_DELTA_T0	0x0	Delta t0 for the protocol timer
0x40040AE8	RF1_PT_DELTA_1	(31:30) PT_DELTA_TS_1_PT_DELTA_T1_MULT	(31:30) PT_DELTA_TS_1_PT_DELTA_T1_MULT	0x0	Multiplier for the delta t1
		(19:0) PT_DELTA_TS_1_PT_DELTA_T1	(19:0) PT_DELTA_TS_1_PT_DELTA_T1	0x0	Delta t1 for the protocol timer
0x40040AEC	RF1_CTE_IF	(25:16) CTE_CTRL_DELAY_TX_DF_DELAY_TX	(25:16) CTE_CTRL_DELAY_TX_DF_DELAY_TX	0x0	Delay (in 62.5ns) form the serializer up to the antenna in direction finding (banked)

Address	Register Name	Register Write	Register Read	Default	Description
		(15) ANTENNA_CONF_DF_IND_PATTERN	(15) ANTENNA_CONF_DF_IND_PATTERN	0x0	Separate the antenna switching pattern from the reference one
		(14) ANTENNA_CONF_DF_IND_ANTENNA	(14) ANTENNA_CONF_DF_IND_ANTENNA	0x0	Make the antenna for DF independent from the rest of the packet
		(13:8) ANTENNA_CONF_ANT_LUT_M	(13:8) ANTENNA_CONF_ANT_LUT_M	0x0	Number of states used (-1) in the antenna LUT
		(5) CTE_AUTO_PULL_EXT_IQ_SMP_TYPE	(5) CTE_AUTO_PULL_EXT_IQ_SMP_TYPE	0x0	Select the external IQ sample signal qualifier type
		(4) CTE_AUTO_PULL_IQ_MSB	(4) CTE_AUTO_PULL_IQ_MSB	0x0	Select which signal is sent over the MSB in case of a 16bits buffers
		(3:2) CTE_AUTO_PULL_IQ_DATA_BUS_SIZE	(3:2) CTE_AUTO_PULL_IQ_DATA_BUS_SIZE	0x0	Select the bus data size of IQ signals
		(1) CTE_AUTO_PULL_CTE_QUAL	(1) CTE_AUTO_PULL_CTE_QUAL	0x0	Select the CTE data qualifier
		(0) CTE_AUTO_PULL_EN_CTE_AUTO_PULL	(0) CTE_AUTO_PULL_EN_CTE_AUTO_PULL	0x0	Enable the automatic push of CTE data to an external IP
0x40040AF0	RF1_CTE_CTRL	(25:16) CTE_CTRL_DELAY_RX_DF_DELAY_SWITCH_RX	(25:16) CTE_CTRL_DELAY_RX_DF_DELAY_SWITCH_RX	0x0	Delay (in 62.5ns) from the antenna up to the deserializer in direction finding (banked)
		(9:0) CTE_CTRL_DELAY_RX_DF_DELAY_SAMPLE_RX	(9:0) CTE_CTRL_DELAY_RX_DF_DELAY_SAMPLE_RX	0x0	Delay (in 62.5ns) from the matched filter up to the deserializer in direction finding (banked)
0x40040AF4	RF1_AGC_ADVANCED	(26:16) AGC_SWITCHES_AGC_SHORTS_LUT	(26:16) AGC_SWITCHES_AGC_SHORTS_LUT	0x0	Array of values that indicates if the highpass shorts must be set for the AGC state passage from n -> n+1
		(8) DEBUG_FAKE_IQ_SAMPLES	(8) DEBUG_FAKE_IQ_SAMPLES	0x0	Generate fake IQ samples
		(7:4) AGC_ADVANCED_	(7:4) AGC_	0x0	Time constant that indicates the time

Address	Register Name	Register Write	Register Read	Default	Description
		AGC_TAU_SHORTS	ADVANCED_AGC_TAU_SHORTS		that shorts must be on
		(3) AGC_ADVANCED_AGC_EN_SHORT_PHADC	(3) AGC_ADVANCED_AGC_EN_SHORT_PHADC	0x0	Enable the short on the phase ADC highpass filter
		(2) AGC_ADVANCED_AGC_EN_SHORT_IFA	(2) AGC_ADVANCED_AGC_EN_SHORT_IFA	0x0	Enable the short on the IFA highpass filter
		(1) AGC_ADVANCED_AGC_USE_SHORTS	(1) AGC_ADVANCED_AGC_USE_SHORTS	0x0	Enable the usage of the shorts located in the BB path
		(0) AGC_ADVANCED_AGC_FULL_SPEED	(0) AGC_ADVANCED_AGC_FULL_SPEED	0x0	Enable the maximum speed in AGC
0x40040AF8	RF1_DATA_STREAMING	(9) DATA_STREAMING_DMA_PHASE_TYPE	(9) DATA_STREAMING_DMA_PHASE_TYPE	0x0	Use the phase after the rescaler instead of the raw phase from phase ADC (banked)
		(8) DATA_STREAMING_DMA_EN_BUS	(8) DATA_STREAMING_DMA_EN_BUS	0x0	Enable the DMA bus on the IP interface (banked)
		(6) DATA_STREAMING_PERIODIC_SAMPLE_AFTER_CTE	(6) DATA_STREAMING_PERIODIC_SAMPLE_AFTER_CTE	0x1	Restart sampling after the CTE period (banked)
		(5) DATA_STREAMING_PERIODIC_SAMPLE_AT_SYNC	(5) DATA_STREAMING_PERIODIC_SAMPLE_AT_SYNC	0x0	Start sampling at the sync detection signal from the delay line (banked)
		(4) DATA_STREAMING_PERIODIC_SAMPLE_OSR_CLK	(4) DATA_STREAMING_PERIODIC_SAMPLE_OSR_CLK	0x0	Oversample (8x) the reference clock of the periodic sample (banked)
		(3:1) DATA_STREAMING_PERIODIC_	(3:1) DATA_STREAMING_	0x3	Division factor (-1) of for the sampling period of the periodic IQ sampling

Address	Register Name	Register Write	Register Read	Default	Description
		SAMPLE_OSR	PERIODIC_SAMPLE_OSR		(banked)
		(0) DATA_STREAMING_PERIODIC_SAMPLE_EN_IQ	(0) DATA_STREAMING_PERIODIC_SAMPLE_EN_IQ	0x0	Sample periodically I and Q channels after the matched filter and put into the IQ FIFO (banked)
0x40040AFC	RF1_REVISION	–	(31:24) CHIP_ID	0x30	Remapped register of CHIP_ID
0x40040B00	RF1_FSM_CTRL	–	(31:30) RXFIFO_STATUS_RX_BIST_ERRORS	0x0	Rx FIFO BIST result
		(31:25) RXFIFO_STATUS_RX_BIST	–	N/A	Start the bist test on the Rx FIFO (code 0x5d)
		–	(29) RXFIFO_STATUS_RX_NEAR_UNDERFLOW	0x0	Rx FIFO near underflow
		–	(28) RXFIFO_STATUS_RX_NEAR_OVERFLOW	0x0	Rx FIFO near overflow
		–	(27) RXFIFO_STATUS_RX_UNDERFLOW	0x0	Rx FIFO underflow
		–	(26) RXFIFO_STATUS_RX_OVERFLOW	0x0	Rx FIFO overflow
		–	(25) RXFIFO_STATUS_RX_FULL	0x0	Rx FIFO full
		–	(24) RXFIFO_STATUS_RX_EMPTY	0x0	Rx FIFO empty
		(24) RXFIFO_STATUS_RX_FLUSH	–	N/A	Rx FIFO flush

Address	Register Name	Register Write	Register Read	Default	Description
		–	(23:22) TXFIFO_ STATUS_TX_BIST_ ERRORS	0x0	Tx FIFO BIST result
		(23:17) TXFIFO_ STATUS_TX_BIST	–	N/A	Start the bist test on the Tx FIFO (code 0x5d)
		–	(21) TXFIFO_ STATUS_TX_NEAR_ UNDERFLOW	0x0	Tx FIFO near underflow
		–	(20) TXFIFO_ STATUS_TX_NEAR_ OVERFLOW	0x0	Tx FIFO near overflow
		–	(19) TXFIFO_ STATUS_TX_ UNDERFLOW	0x0	Tx FIFO underflow
		–	(18) TXFIFO_ STATUS_TX_OVERFLOW	0x0	Tx FIFO overflow
		–	(17) TXFIFO_ STATUS_TX_FULL	0x0	Tx FIFO full
		–	(16) TXFIFO_ STATUS_TX_EMPTY	0x0	Tx FIFO empty
		(16) TXFIFO_STATUS_ TX_FLUSH	–	N/A	Tx FIFO flush
		–	(10) FSM_STATUS_ TX_NRX	0x0	Select Rx or Tx mode
		–	(9:8) FSM_STATUS_ STATUS	0x0	Status of the FSM
		(3) FSM_MODE_RESET	–	N/A	FSM reset
		–	(2) FSM_MODE_RX_ MODE	0x0	Rx status

Address	Register Name	Register Write	Register Read	Default	Description
		(2) FSM_MODE_TX_NRX	-	N/A	Set the radio in Tx or Rx mode
		-	(1) FSM_MODE_TX_MODE	0x0	Tx status
		(1:0) FSM_MODE_MODE	-	N/A	Set the FSM mode
		-	(0) FSM_MODE_N_IDLE	0x0	FSM status
0x40040B04	RF1_IQFIFO_STATUS	-	(24:16) TXFIFO_COUNT_TX_COUNT	0x0	Number of bytes in the Tx FIFO
		-	(15:8) IQFIFO_COUNT_IQ_COUNT	0x0	Number of bytes in the IQ FIFO
		-	(7:6) IQFIFO_STATUS_IQ_BIST_ERRORS	0x0	IQ FIFO BIST result
		(7:1) IQFIFO_STATUS_IQ_BIST	-	N/A	Start the BIST test on the IQ FIFO (code 0x5d)
		-	(5) IQFIFO_STATUS_IQ_NEAR_UNDERFLOW	0x0	IQ FIFO near underflow
		-	(4) IQFIFO_STATUS_IQ_NEAR_OVERFLOW	0x0	IQ FIFO near overflow
		-	(3) IQFIFO_STATUS_IQ_UNDERFLOW	0x0	IQ FIFO underflow
		-	(2) IQFIFO_STATUS_IQ_OVERFLOW	0x0	IQ FIFO overflow
		-	(1) IQFIFO_STATUS_IQ_FULL	0x0	IQ FIFO full
		-	(0) IQFIFO_STATUS_IQ_EMPTY	0x0	IQ FIFO empty
		(0) IQFIFO_STATUS_	-	N/A	IQ FIFO flush

Address	Register Name	Register Write	Register Read	Default	Description
		FLUSH			
0x40040B08	RF1_TXFIFO	(7:0) TXFIFO_TX_DATA	–	N/A	Data to be sent
0x40040B0C	RF1_RXFIFO	–	(7:0) RXFIFO_RX_DATA	0x0	Received data
0x40040B10	RF1_IQFIFO	–	(7:0) IQFIFO_IQ_DATA	0x0	IQ data for AoA or AoD
0x40040B14	RF1_REG45	–	(25:16) RSSI_AVG_RSSI_AVG	0x0	Filtered RSSI value
		–	(8:0) RXFIFO_COUNT_RX_COUNT	0x0	Number of bytes in the Rx FIFO
0x40040B18	RF1_DESER_STATUS	–	(7) DESER_STATUS_SIGNAL_RECEIVING	0x0	Deserializer enabling
		–	(6) DESER_STATUS_SYNC_DETECTED	0x0	Sync word detection
		–	(5) DESER_STATUS_WAIT_SYNC	0x0	Deserializer waiting for the sync word
		–	(4) DESER_STATUS_IS_ADDRESS_BR	0x0	Received address
		–	(3) DESER_STATUS_PKT_LEN_ERR	0x0	Packet length
		–	(2) DESER_STATUS_ADDRESS_ERR	0x0	Address error
		–	(1) DESER_STATUS_CRC_ERR	0x0	CRC error
		–	(0) DESER_STATUS_DESER_FINISH	0x0	Deserializer status
0x40040B1C	RF1_BLE_AEC_CCM	–	(2) BLE_AES_CCM_	0x0	AES CCM MIC error

Address	Register Name	Register Write	Register Read	Default	Description
			BLE_AES_MIC_OK		
		–	(1) BLE_AES_CCM_ BLE_AES_DONE_RX	0x0	AES CCM packet decoding
		–	(0) BLE_AES_CCM_ BLE_AES_DONE_TX	0x0	AES CCM packet encoding
0x40040B20	RF1_IRQ_STATUS	–	(5) IRQ_STATUS_ FLAG_RXFIFO	0x0	IRQ RXFIFO status
		–	(4) IRQ_STATUS_ FLAG_TXFIFO	0x0	IRQ TXFIFO status
		–	(3) IRQ_STATUS_ FLAG_SYNC	0x0	IRQ SYNC status
		–	(2) IRQ_STATUS_ FLAG_RECEIVED	0x0	IRQ RECEIVED status
		–	(1) IRQ_STATUS_ FLAG_RXSTOP	0x0	IRQ RXSTOP status
		–	(0) IRQ_STATUS_ FLAG_TX	0x0	IRQ Tx status
0x40040B24	RF1_RSSI_MIN_MAX	–	(25:16) RSSI_MAX_ RSSI_MAX	0x0	Maximum RSSI value over a filtering period
		–	(9:0) RSSI_MIN_ RSSI_MIN	0x0	Minimum RSSI value over a filtering period
0x40040B28	RF1_REG4A	–	(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) DR_ERR_ IND_DR_ERR_IND	0x0	Data-rate error indicator

Address	Register Name	Register Write	Register Read	Default	Description
		–	(9:0) RSSI_PKT_RSSI_PKT	0x0	Filtered RSSI value sampled during the packet reception
0x40040B2C	RF1_FEI	–	(31:16) FEI_PKT_FEI_PKT	0x0	Frequency error indicator sampled during the packet reception
		–	(15:0) FEI_FEI_OUT	0x0	Frequency error indicator
0x40040B30	RF1_REG4C	–	(31:24) LINK_QUAL_PKT_LINK_QUALITY_PKT	0x0	Link quality indicator sampled during the packet reception
		–	(23:16) LINK_QUAL_LINK_QUALITY	0x0	Instantaneous link quality indicator
		–	(15:0) FEI_AFC_FEI_AFC	0x0	Frequency error indicator sampled during the AFC
0x40040B34	RF1_ANALOG_INFO	(25:24) BLR_READOUT_BLR_RATE	–	N/A	Bluetooth LE long range rate indicator
		–	(22:20) PEAK_DET_VAL_PEAK_DET_FILT	0x0	Distance from the subband center (only available with the FLL method)
		–	(18:16) PEAK_DET_VAL_PEAK_DET_RAW	0x0	Distance from the subband center (only available with the FLL method)
		–	(15) ANALOG_INFO_POR_VDDA	0x0	VDDA LDO disable status
		–	(14) ANALOG_INFO_PLL_UNLOCK	0x0	PLL unlock status
		–	(13) ANALOG_INFO_XTAL_FINISH	0x0	XTAL algorithm status
		–	(12) ANALOG_INFO_DLL_LOCKED	0x0	DLL lock status
		–	(11) ANALOG_INFO_CLK_DIG_READY	0x0	Ready signal of the digital clock

Address	Register Name	Register Write	Register Read	Default	Description
		–	(10) ANALOG_INFO_CLK_PLL_READY	0x0	PLL clock status
		–	(9:8) ANALOG_INFO_SUBBAND	0x0	Status of the subband comparator Hi
		–	(7:0) SUBBAND_ERR_SB_FLL_ERR	0x0	Distance from the subband center (only available with the FLL method)
0x40040B38	RF1_SAMPLE_RSSI	(0) SAMPLE_RSSI	–	N/A	Sample the thermometric RSSI
0x40040B3C	RF1_RSSI_THERM	–	(29:0) RSSI_THERM	0x0	Thermometric value of the RSSI
0x40040B80	RF1_LUT_ANTENNA_ARRAY_1	(31:28) LUT_ANTENNA_ARRAY_1_ANTENNA_7	(31:28) LUT_ANTENNA_ARRAY_1_ANTENNA_7	0x0	Antenna 7 specification
		(27:24) LUT_ANTENNA_ARRAY_1_ANTENNA_6	(27:24) LUT_ANTENNA_ARRAY_1_ANTENNA_6	0x0	Antenna 6 specification
		(23:20) LUT_ANTENNA_ARRAY_1_ANTENNA_5	(23:20) LUT_ANTENNA_ARRAY_1_ANTENNA_5	0x0	Antenna 5 specification
		(19:16) LUT_ANTENNA_ARRAY_1_ANTENNA_4	(19:16) LUT_ANTENNA_ARRAY_1_ANTENNA_4	0x0	Antenna 4 specification
		(15:12) LUT_ANTENNA_ARRAY_1_ANTENNA_3	(15:12) LUT_ANTENNA_ARRAY_1_ANTENNA_3	0x3	Antenna 3 specification
		(11:8) LUT_ANTENNA_ARRAY_1_ANTENNA_2	(11:8) LUT_ANTENNA_ARRAY_1_ANTENNA_2	0x2	Antenna 2 specification
		(7:4) LUT_ANTENNA_ARRAY_1_ANTENNA_1	(7:4) LUT_ANTENNA_ARRAY_1_ANTENNA_1	0x1	Antenna 1 specification
		(3:0) LUT_ANTENNA_	(3:0) LUT_ANTENNA_	0x0	Antenna 0 specification

Address	Register Name	Register Write	Register Read	Default	Description
		ARRAY_1_ANTENNA_0	ARRAY_1_ANTENNA_0		
0x40040B84	RF1_LUT_ANTENNA_ARRAY_2	(31:28) LUT_ANTENNA_ARRAY_2_ANTENNA_15	(31:28) LUT_ANTENNA_ARRAY_2_ANTENNA_15	0x0	Antenna 15 specification
		(27:24) LUT_ANTENNA_ARRAY_2_ANTENNA_14	(27:24) LUT_ANTENNA_ARRAY_2_ANTENNA_14	0x0	Antenna 14 specification
		(23:20) LUT_ANTENNA_ARRAY_2_ANTENNA_13	(23:20) LUT_ANTENNA_ARRAY_2_ANTENNA_13	0x0	Antenna 13 specification
		(19:16) LUT_ANTENNA_ARRAY_2_ANTENNA_12	(19:16) LUT_ANTENNA_ARRAY_2_ANTENNA_12	0x0	Antenna 12 specification
		(15:12) LUT_ANTENNA_ARRAY_2_ANTENNA_11	(15:12) LUT_ANTENNA_ARRAY_2_ANTENNA_11	0x0	Antenna 11 specification
		(11:8) LUT_ANTENNA_ARRAY_2_ANTENNA_10	(11:8) LUT_ANTENNA_ARRAY_2_ANTENNA_10	0x0	Antenna 10 specification
		(7:4) LUT_ANTENNA_ARRAY_2_ANTENNA_9	(7:4) LUT_ANTENNA_ARRAY_2_ANTENNA_9	0x0	Antenna 9 specification
		(3:0) LUT_ANTENNA_ARRAY_2_ANTENNA_8	(3:0) LUT_ANTENNA_ARRAY_2_ANTENNA_8	0x0	Antenna 8 specification
0x40040B88	RF1_LUT_ANTENNA_ARRAY_3	(31:28) LUT_ANTENNA_ARRAY_3_ANTENNA_23	(31:28) LUT_ANTENNA_ARRAY_3_ANTENNA_23	0x0	Antenna 23 specification
		(27:24) LUT_ANTENNA_ARRAY_3_ANTENNA_22	(27:24) LUT_ANTENNA_ARRAY_3_ANTENNA_22	0x0	Antenna 22 specification

Address	Register Name	Register Write	Register Read	Default	Description
		(23:20) LUT_ ANTENNA_ARRAY_3_ ANTENNA_21	(23:20) LUT_ ANTENNA_ARRAY_3_ ANTENNA_21	0x0	Antenna 21 specification
		(19:16) LUT_ ANTENNA_ARRAY_3_ ANTENNA_20	(19:16) LUT_ ANTENNA_ARRAY_3_ ANTENNA_20	0x0	Antenna 20 specification
		(15:12) LUT_ ANTENNA_ARRAY_3_ ANTENNA_19	(15:12) LUT_ ANTENNA_ARRAY_3_ ANTENNA_19	0x0	Antenna 19 specification
		(11:8) LUT_ANTENNA_ ARRAY_3_ANTENNA_18	(11:8) LUT_ ANTENNA_ARRAY_3_ ANTENNA_18	0x0	Antenna 18 specification
		(7:4) LUT_ANTENNA_ ARRAY_3_ANTENNA_17	(7:4) LUT_ANTENNA_ ARRAY_3_ANTENNA_17	0x0	Antenna 17 specification
		(3:0) LUT_ANTENNA_ ARRAY_3_ANTENNA_16	(3:0) LUT_ANTENNA_ ARRAY_3_ANTENNA_16	0x0	Antenna 16 specification
0x40040B8C	RF1_LUT_ANTENNA_ARRAY_ 4	(31:28) LUT_ ANTENNA_ARRAY_4_ ANTENNA_31	(31:28) LUT_ ANTENNA_ARRAY_4_ ANTENNA_31	0x0	Antenna 31 specification
		(27:24) LUT_ ANTENNA_ARRAY_4_ ANTENNA_30	(27:24) LUT_ ANTENNA_ARRAY_4_ ANTENNA_30	0x0	Antenna 30 specification
		(23:20) LUT_ ANTENNA_ARRAY_4_ ANTENNA_29	(23:20) LUT_ ANTENNA_ARRAY_4_ ANTENNA_29	0x0	Antenna 29 specification
		(19:16) LUT_ ANTENNA_ARRAY_4_ ANTENNA_28	(19:16) LUT_ ANTENNA_ARRAY_4_ ANTENNA_28	0x0	Antenna 28 specification
		(15:12) LUT_ ANTENNA_ARRAY_4_	(15:12) LUT_ ANTENNA_ARRAY_4_	0x0	Antenna 27 specification

Address	Register Name	Register Write	Register Read	Default	Description
		ANTENNA_27	ANTENNA_27		
		(11:8) LUT_ANTENNA_ARRAY_4_ANTENNA_26	(11:8) LUT_ANTENNA_ARRAY_4_ANTENNA_26	0x0	Antenna 26 specification
		(7:4) LUT_ANTENNA_ARRAY_4_ANTENNA_25	(7:4) LUT_ANTENNA_ARRAY_4_ANTENNA_25	0x0	Antenna 25 specification
		(3:0) LUT_ANTENNA_ARRAY_4_ANTENNA_24	(3:0) LUT_ANTENNA_ARRAY_4_ANTENNA_24	0x0	Antenna 24 specification
0x40040BC0	RF1_REG50	–	(27) FEATURES_HAS_BLE_AES	0x0	Bluetooth AES block availability
		–	(26) FEATURES_HAS_BLE_DF_AOA_AOD	0x1	Bluetooth Direction Finding AoA/AoD feature availability
		–	(25) FEATURES_HAS_BLE_LONG_RANGE	0x1	Bluetooth long range feature availability
		–	(24) FEATURES_FEATURES_AVAILABLE	0x1	Features availability
		(23:16) BLR_PKT_LEN_BLR_PKT_LEN	–	N/A	Packet length of the BLR packet
		(15:8) PROT_TIMER_PT_CMD	–	N/A	Protocol timer command
		(0) COMMANDS_START_SUBBAND	–	N/A	Subband selection algorithm
0x40040BE0	RF1_REG51	(31:24) FSM_MODE_RM_TX	(31:24) FSM_MODE_RM_TX	0x0	Remapped register of FSM_MODE
		(23:16) PA_PWR_RM	(23:16) PA_PWR_RM	0x0	Remapped register of PA_PWR
		(15:8) CHANNEL_RM_TX	(15:8) CHANNEL_RM_TX	0x0	Remapped register of CHANNEL

Address	Register Name	Register Write	Register Read	Default	Description
		(7:0) RATE_TX	(7:0) RATE_TX	0x0	Remapped register of BANK
0x40040BE8	RF1_REG52	(31:24) ACCESS_ ADDRESS	(31:24) ACCESS_ ADDRESS	0x0	Remapped register of PATTERN
		(23:16) FSM_MODE_ RM_RX	(23:16) FSM_MODE_ RM_RX	0x0	Remapped register of FSM_MODE
		(15:8) CHANNEL_RM_ RX	(15:8) CHANNEL_RM_ RX	0x0	Remapped register of CHANNEL
		(7:0) RATE_RX	(7:0) RATE_RX	0x0	Remapped register of BANK
0x40040BF0	RF1_REG53	-	(23:16) RSSI_MAX_ RM	0x0	Remapped register of RSSI_MAX
		-	(15:8) RSSI_MIN_RM	0x0	Remapped register of RSSI_MIN
		-	(7:0) RSSI_AVG_RM	0x0	Remapped register of RSSI_AVG
0x40040BF4	RF1_REG54	(7:0) BLR_PACKET_ LEN	-	N/A	Remapped register of BLR_PACKET_ LEN
0x40040BF8	RF1_REG55	-	(7:0) ITRX_ FEATURES	0x0	Remapped register of ITRX_ FEATURES
0x40040BFC	RF1_REG56	-	(31:24) CHIP_ID_ CHIP_ID	0x30	Version of the chip
		-	(23:16) MD5_REGS_ MD5_REGS	0x0	MD5 calculated on the register map file
		(15:8) SCAN_2_SCAN_ 2_PASSWORD	-	N/A	SCAN 2 key
		(7:0) SCAN_1_SCAN_ 1_PASSWORD	-	N/A	SCAN 1 key
0x40040C00	RF2_REG00	(31) DATAWHITE_ BTLE_DW_BTLE	(31) DATAWHITE_ BTLE_DW_BTLE	0x1	Data whitening control
		(30:24) DATAWHITE_	(30:24) DATAWHITE_	0x0	Reset value to put on the Bluetooth LE

Address	Register Name	Register Write	Register Read	Default	Description
		BTLE_DW_BTLE_RST	BTLE_DW_BTLE_RST		data whitening shift register
		(23) FOURFSK_CODING_EN_FOURFSK_CODING	(23) FOURFSK_CODING_EN_FOURFSK_CODING	0x0	Enable 4FSK coding
		(22:20) FOURFSK_CODING_TX_FOURFSK_CODING	(22:20) FOURFSK_CODING_TX_FOURFSK_CODING	0x0	Set the 4FSK coding (Tx mode)
		(18:16) FOURFSK_CODING_RX_FOURFSK_CODING	(18:16) FOURFSK_CODING_RX_FOURFSK_CODING	0x0	Set the 4FSK decoding (Rx mode)
		(14) MODE2_DIFF_CODING	(14) MODE2_DIFF_CODING	0x0	Differential coding/decoding
		(13) MODE2_PSK_NFSK	(13) MODE2_PSK_NFSK	0x0	FSK/PSK mode selection
		(12:8) MODE2_TESTMODE	(12:8) MODE2_TESTMODE	0x0	Output test mode
		(7) MODE_NOT_TO_IDLE	(7) MODE_NOT_TO_IDLE	0x0	FSM goes in suspend mode after a Tx or Rx packet
		(5) MODE_EN_FSM	(5) MODE_EN_FSM	0x1	Radio FSM control
		(4) MODE_EN_DESERIALIZER	(4) MODE_EN_DESERIALIZER	0x0	Deserializer control
		(3) MODE_EN_SERIALIZER	(3) MODE_EN_SERIALIZER	0x0	Serializer control
		(2) MODE_TX_NRX	(2) MODE_TX_NRX	0x0	Select Tx or Rx mode
		(1:0) MODE_MODE	(1:0) MODE_MODE	0x2	Select the working mode of the digital baseband
0x40040C04	RF2_REG01	(31:24) TAU_PHASE_RECOV_TAU_PHASE_	(31:24) TAU_PHASE_RECOV_TAU_PHASE_	0x14	Time constant of the fine carrier recovery block (banked)

Address	Register Name	Register Write	Register Read	Default	Description
		RECOV	RECOV		
		(23:16) TAU_ROUGH_ RECOV_TAU_ROUGH_ RECOV	(23:16) TAU_ROUGH_ RECOV_TAU_ROUGH_ RECOV	0xB	Time constant of the rough carrier recovery block (banked)
		(15) CARRIER_ RECOVERY_EN_ CORRECT_CFREQ_AFC	(15) CARRIER_ RECOVERY_EN_ CORRECT_CFREQ_AFC	0x0	Automatic AFC correction (banked)
		(14) CARRIER_ RECOVERY_CORRECT_ CFREQ_IF_NEG	(14) CARRIER_ RECOVERY_CORRECT_ CFREQ_IF_NEG	0x0	IF correction (banked)
		(13) CARRIER_ RECOVERY_EN_ CORRECT_CFREQ_IF	(13) CARRIER_ RECOVERY_EN_ CORRECT_CFREQ_IF	0x1	Automatic IF correction (banked)
		(12) CARRIER_ RECOVERY_AFC_NEG	(12) CARRIER_ RECOVERY_AFC_NEG	0x0	AFC correction (banked)
		(11) CARRIER_ RECOVERY_STARTER_ MODE	(11) CARRIER_ RECOVERY_STARTER_ MODE	0x0	Starter mode (banked)
		(10) CARRIER_ RECOVERY_EN_AFC	(10) CARRIER_ RECOVERY_EN_AFC	0x0	Automatic frequency control (banked)
		(9) CARRIER_ RECOVERY_EN_FINE_ RECOV	(9) CARRIER_ RECOVERY_EN_FINE_ RECOV	0x1	Fine carrier recovery (banked)
		(8) CARRIER_ RECOVERY_EN_ROUGH_ RECOV	(8) CARRIER_ RECOVERY_EN_ROUGH_ RECOV	0x0	Rough carrier recovery (banked)
		(6) MOD_TX_PULSE_ NSYM	(6) MOD_TX_PULSE_ NSYM	0x0	Tx pulse shape function
		(5) MOD_TX_EN_	(5) MOD_TX_EN_	0x0	Tx CIC interpolator

Address	Register Name	Register Write	Register Read	Default	Description
		INTERP	INTERP		
		(4:0) MOD_TX_CK_TX_M	(4:0) MOD_TX_CK_TX_M	0x0	Unsigned value determining the Tx CIC interpolator frequency
0x40040C08	RF2_REG02	(25:24) DATARATE_OFFSET_DR_LIMIT	(25:24) DATARATE_OFFSET_DR_LIMIT	0x0	Set the data-rate recovery limits
		(23:16) DATARATE_OFFSET_DATARATE_OFFSET	(23:16) DATARATE_OFFSET_DATARATE_OFFSET	0x0	Data-rate offset
		(15:8) TAU_DATARATE_RECOV_TAU_DATARATE_RECOV	(15:8) TAU_DATARATE_RECOV_TAU_DATARATE_RECOV	0x20	Time constant of the data-rate recovery
		(7:0) TAU_CLK_RECOV_TAU_CLK_RECOV	(7:0) TAU_CLK_RECOV_TAU_CLK_RECOV	0x9	Time constant of the clock recovery (banked)
0x40040C0C	RF2_REG03	(31:30) MAC_CONF_MAC_TIMER_GR	(31:30) MAC_CONF_MAC_TIMER_GR	0x2	MAC timer granularity
		(29) MAC_CONF_RX_MAC_ACT	(29) MAC_CONF_RX_MAC_ACT	0x0	Switch FSM to Rx or Tx mode after an Rx mode
		(28) MAC_CONF_RX_MAC_TX_NRX	(28) MAC_CONF_RX_MAC_TX_NRX	0x0	Switch FSM to Tx mode after an Rx mode (Rx otherwise)
		(27) MAC_CONF_RX_MAC_START_NSTOP	(27) MAC_CONF_RX_MAC_START_NSTOP	0x0	MAC timer activation after sync word detection
		(26) MAC_CONF_TX_MAC_ACT	(26) MAC_CONF_TX_MAC_ACT	0x0	Switch FSM to Rx or Tx mode after a Tx mode
		(25) MAC_CONF_TX_MAC_TX_NRX	(25) MAC_CONF_TX_MAC_TX_NRX	0x0	Switch FSM to Tx mode after a Tx mode (Rx otherwise)
		(24) MAC_CONF_TX_MAC_START_NSTOP	(24) MAC_CONF_TX_MAC_START_NSTOP	0x0	MAC timer activation after packet transmission

Address	Register Name	Register Write	Register Read	Default	Description
		(23) IRQ_CONF_IRQ_HIGH_Z	(23) IRQ_CONF_IRQ_HIGH_Z	0x0	Pads are set to high-Z when the IRQ is not active
		(22) IRQ_CONF_IRQ_ACTIVE_LOW	(22) IRQ_CONF_IRQ_ACTIVE_LOW	0x1	IRQ are active low
		(21:16) IRQ_CONF_IRQS_MASK	(21:16) IRQ_CONF_IRQS_MASK	0x0	Mask to determine which IRQs are enabled (active high)
		(15:13) FIFO_2_FIFO_THR_TX	(15:13) FIFO_2_FIFO_THR_TX	0x0	Threshold indicating the "almost empty" Tx FIFO state
		(12) FIFO_2_WAIT_TXFIFO_WR	(12) FIFO_2_WAIT_TXFIFO_WR	0x0	FSM will wait a Tx FIFO write before starting the Tx mode in case of an empty Tx FIFO
		(11) FIFO_2_STOP_ON_RXFF_OVFLW	(11) FIFO_2_STOP_ON_RXFF_OVFLW	0x0	Stop the reception in case of a FIFO overflow
		(10) FIFO_2_STOP_ON_TXFF_UNFLW	(10) FIFO_2_STOP_ON_TXFF_UNFLW	0x0	Stop the transmission in case of a FIFO underflow
		(9) FIFO_2_RXFF_FLUSH_ON_START	(9) FIFO_2_RXFF_FLUSH_ON_START	0x1	Flush the Rx FIFO when the Rx mode is enabled in order to receive a packet with an empty FIFO
		(8) FIFO_2_TXFF_FLUSH_ON_STOP	(8) FIFO_2_TXFF_FLUSH_ON_STOP	0x1	Flush the Tx FIFO after the end of a packet transmission in order to have an empty FIFO
		(7) FIFO_FIFO_FLUSH_ON_OVFLW	(7) FIFO_FIFO_FLUSH_ON_OVFLW	0x0	Overflow FIFO flush control
		(6) FIFO_FIFO_FLUSH_ON_ADDR_ERR	(6) FIFO_FIFO_FLUSH_ON_ADDR_ERR	0x0	Address error FIFO flush control
		(5) FIFO_FIFO_FLUSH_ON_PL_ERR	(5) FIFO_FIFO_FLUSH_ON_PL_ERR	0x0	Packet length error FIFO flush control
		(4) FIFO_FIFO_FLUSH_ON_CRC_ERR	(4) FIFO_FIFO_FLUSH_ON_CRC_ERR	0x1	CRC error FIFO flush control

Address	Register Name	Register Write	Register Read	Default	Description
		(3) FIFO_RX_FIFO_ACK	(3) FIFO_RX_FIFO_ACK	0x0	Rx FIFO acknowledgement
		(2:0) FIFO_FIFO_THR	(2:0) FIFO_FIFO_THR	0x0	Threshold indicating the "almost full" Rx FIFO state
0x40040C10	RF2_PADS_03	(28:24) PAD_CONF_1_PAD_3_CONF	(28:24) PAD_CONF_1_PAD_3_CONF	0x0	Configuration of GPIO pad 3
		(20:16) PAD_CONF_1_PAD_2_CONF	(20:16) PAD_CONF_1_PAD_2_CONF	0x0	Configuration of GPIO pad 2
		(12:8) PAD_CONF_1_PAD_1_CONF	(12:8) PAD_CONF_1_PAD_1_CONF	0x0	Configuration of GPIO pad 1
		(4:0) PAD_CONF_1_PAD_0_CONF	(4:0) PAD_CONF_1_PAD_0_CONF	0x0	Configuration of GPIO pad 0
0x40040C14	RF2_PADS_47	(28:24) PAD_CONF_2_PAD_7_CONF	(28:24) PAD_CONF_2_PAD_7_CONF	0x0	Configuration of GPIO pad 7
		(20:16) PAD_CONF_2_PAD_6_CONF	(20:16) PAD_CONF_2_PAD_6_CONF	0x0	Configuration of GPIO pad 6
		(12:8) PAD_CONF_2_PAD_5_CONF	(12:8) PAD_CONF_2_PAD_5_CONF	0x0	Configuration of GPIO pad 5
		(4:0) PAD_CONF_2_PAD_4_CONF	(4:0) PAD_CONF_2_PAD_4_CONF	0x0	Configuration of GPIO pad 4
0x40040C18	RF2_CENTER_FREQ	(31) CENTER_FREQ_ADAPT_CFREQ	(31) CENTER_FREQ_ADAPT_CFREQ	0x1	Frequency adaptation between Tx and Rx modes
		(30) CENTER_FREQ_RX_DIV_5_N6	(30) CENTER_FREQ_RX_DIV_5_N6	0x0	Ratio of the PLL reference between Tx and Rx modes
		(29:0) CENTER_FREQ_CENTER_FREQUENCY	(29:0) CENTER_FREQ_CENTER_FREQUENCY	0x215C71B	Set the center frequency
0x40040C1C	RF2_PADS_89	(31:24) TX_MAC_	(31:24) TX_MAC_	0x82	Time to wait after the Tx mode

Address	Register Name	Register Write	Register Read	Default	Description
		TIMER_TX_MAC_TIMER	TIMER_TX_MAC_TIMER		
		(23:16) RX_MAC_ TIMER_RX_MAC_TIMER	(23:16) RX_MAC_ TIMER_RX_MAC_TIMER	0x23	Time to wait after the Rx mode
		(12:8) PAD_CONF_3_ PAD_9_CONF	(12:8) PAD_CONF_3_ PAD_9_CONF	0x0	Configuration of GPIO pad 9
		(4:0) PAD_CONF_3_ PAD_8_CONF	(4:0) PAD_CONF_3_ PAD_8_CONF	0x0	Configuration of GPIO pad 8
0x40040C20	RF2_REG08	(31:30) MOD_INFO_ RX_DIV_CK_RX	(31:30) MOD_INFO_ RX_DIV_CK_RX	0x0	Set the clock divider for the Rx mode (banked)
		(29) MOD_INFO_RX_ SYMBOL_2BIT_RX	(29) MOD_INFO_RX_ SYMBOL_2BIT_RX	0x0	Rx symbol bits composition (banked)
		(28:24) MOD_INFO_ RX_DR_M_RX	(28:24) MOD_INFO_ RX_DR_M_RX	0x0	Unsigned value determining the oversampling frequency and consequently the data-rate (banked)
		(23:22) MOD_INFO_ TX_DIV_CK_TX	(23:22) MOD_INFO_ TX_DIV_CK_TX	0x0	Set the clock divider for the Tx mode (banked)
		(21) MOD_INFO_TX_ SYMBOL_2BIT_TX	(21) MOD_INFO_TX_ SYMBOL_2BIT_TX	0x0	Tx symbol bits composition (banked)
		(20:16) MOD_INFO_ TX_DR_M_TX	(20:16) MOD_INFO_ TX_DR_M_TX	0x0	Unsigned value determining the oversampling frequency and consequently the data-rate (banked)
		(14) CHANNEL_ SWITCH_IQ	(14) CHANNEL_ SWITCH_IQ	0x0	Switch I and Q channels
		(13:8) CHANNEL_ CHANNEL	(13:8) CHANNEL_ CHANNEL	0x0	Channel number
		(3) BANK_DATARATE_ TX_NRX	(3) BANK_DATARATE_ TX_NRX	0x0	Select the data-rate register
		(2) BANK_STD_BLE_	(2) BANK_STD_BLE_	0x0	Select the actual bank behavior

Address	Register Name	Register Write	Register Read	Default	Description
		RATES	RATES		
		(1:0) BANK_BANK	(1:0) BANK_BANK	0x0	Select the used bank
0x40040C24	RF2_CODING	(31) CODING_EN_ DATAWHITE	(31) CODING_EN_ DATAWHITE	0x1	Data-whitening enabling (banked)
		(30) CODING_I_NQ_ DELAYED	(30) CODING_I_NQ_ DELAYED	0x0	Channel I delay (banked)
		(29) CODING_OFFSET	(29) CODING_OFFSET	0x0	Offset (delay) introduction (banked)
		(28) CODING_BIT_ INVERT	(28) CODING_BIT_ INVERT	0x0	Bit value inversion in Tx and Rx modes (banked)
		(27) CODING_EVEN_ BEFORE_ODD	(27) CODING_EVEN_ BEFORE_ODD	0x0	Determine the bit order in case of a 2 bits per symbol modulation (banked)
		(26) CODING_EN_ 802154_L2F	(26) CODING_EN_ 802154_L2F	0x0	Linear to frequency encoding needed in order to modulate an OQPSK as an MSK (banked)
		(25) CODING_EN_ 802154_B2C	(25) CODING_EN_ 802154_B2C	0x0	Bit to chips encoding used in the IEEE 802.15.4 standard (banked)
		(24) CODING_EN_ MANCHESTER	(24) CODING_EN_ MANCHESTER	0x0	Manchester encoding (banked)
		(23) CHANNELS_2_EN_ CHANNEL_SEL	(23) CHANNELS_2_ EN_CHANNEL_SEL	0x1	Definition of channels (banked)
		(22) CHANNELS_2_EN_ CHN_BLE	(22) CHANNELS_2_ EN_CHN_BLE	0x1	BLE channels index LUT (banked)
		(19:16) CHANNELS_2_ CHANNEL_SPACING_HI	(19:16) CHANNELS_ 2_CHANNEL_SPACING_ HI	0x7	Channel spacing MSB (banked)
		(15:0) CHANNELS_1_ CHANNEL_SPACING_LO	(15:0) CHANNELS_1_ CHANNEL_SPACING_LO	0x1C72	Channel spacing LSB (banked)
0x40040C28	RF2_PACKET_HANDLING	(31:24) PREAMBLE_	(31:24) PREAMBLE_	0x55	Preamble to be inserted (banked)

Address	Register Name	Register Write	Register Read	Default	Description
		PREAMBLE	PREAMBLE		
		(22) PACKET_LENGTH_OPTS_EN_PACKET_LEN_FIX	(22) PACKET_LENGTH_OPTS_EN_PACKET_LEN_FIX	0x0	Packet length configuration (banked)
		(21:18) PACKET_LENGTH_OPTS_PACKET_LEN_CORR	(21:18) PACKET_LENGTH_OPTS_PACKET_LEN_CORR	0x0	Signed value specifying the correction to apply to the specified packet length (banked)
		(17:16) PACKET_LENGTH_OPTS_PACKET_LEN_POS	(17:16) PACKET_LENGTH_OPTS_PACKET_LEN_POS	0x1	Unsigned value that specifies the position of the packet length after the pattern (banked)
		(15:8) PACKET_LENGTH_PACKET_LEN	(15:8) PACKET_LENGTH_PACKET_LEN	0xFF	The packet length in the fixed packet length mode (banked)
		(7) PACKET_HANDLING_LSB_FIRST	(7) PACKET_HANDLING_LSB_FIRST	0x1	Select LSB or MSB to send first (banked)
		(6) PACKET_HANDLING_EN_CRC	(6) PACKET_HANDLING_EN_CRC	0x1	Automatic CRC evaluation and insertion (banked)
		(5) PACKET_HANDLING_EN_CRC_ON_PKTLEN	(5) PACKET_HANDLING_EN_CRC_ON_PKTLEN	0x1	CRC calculation on the packet length part of the packet (banked)
		(4) PACKET_HANDLING_EN_PREAMBLE	(4) PACKET_HANDLING_EN_PREAMBLE	0x1	Automatic preamble insertion (banked)
		(3) PACKET_HANDLING_EN_MULTI_FRAME	(3) PACKET_HANDLING_EN_MULTI_FRAME	0x0	Multi-frame packet (banked)
		(2) PACKET_HANDLING_ENB_DW_ON_CRC	(2) PACKET_HANDLING_ENB_DW_ON_CRC	0x0	Data-whitening on the CRC disabling (banked)
		(1) PACKET_	(1) PACKET_	0x1	Automatic pattern insertion and

Address	Register Name	Register Write	Register Read	Default	Description
		HANDLING_EN_PATTERN	HANDLING_EN_ PATTERN		recognition (banked)
		(0) PACKET_ HANDLING_EN_PACKET	(0) PACKET_ HANDLING_EN_PACKET	0x1	Packet handler enabling (banked)
0x40040C2C	RF2_SYNC_PATTERN	(31:0) PATTERN	(31:0) PATTERN	0x8E89BED6	Pattern (sync word) to be inserted or recognized (banked)
0x40040C30	RF2_REG0C	(31:16) ADDRESS_ ADDRESS	(31:16) ADDRESS_ ADDRESS	0x0	Address of the node (banked)
		(11) ADDRESS_CONF_ ADDRESS_LEN	(11) ADDRESS_CONF_ ADDRESS_LEN	0x0	Address length selection (banked)
		(10) ADDRESS_CONF_ EN_ADDRESS_RX_BR	(10) ADDRESS_CONF_ EN_ADDRESS_RX_BR	0x0	Broadcast address detection in Rx mode (banked)
		(9) ADDRESS_CONF_ EN_ADDRESS_RX	(9) ADDRESS_CONF_ EN_ADDRESS_RX	0x0	Address detection in Rx mode (banked)
		(8) ADDRESS_CONF_ EN_ADDRESS_TX	(8) ADDRESS_CONF_ EN_ADDRESS_TX	0x0	Address insertion in Tx mode (banked)
		(7:0) PREAMBLE_ LENGTH_PREAMBLE_LEN	(7:0) PREAMBLE_ LENGTH_PREAMBLE_ LEN	0x0	Length of the preamble -1 (banked)
0x40040C34	RF2_PACKET_EXTRA	(29:28) CONV_CODES_ CONF_STOP_WORD_LEN	(29:28) CONV_ CODES_CONF_STOP_ WORD_LEN	0x0	Length of the stop word (banked)
		(27:26) CONV_CODES_ CONF_CC_VITERBI_LEN	(27:26) CONV_ CODES_CONF_CC_ VITERBI_LEN	0x2	Set the memory length of the Viterbi decoder (banked)
		(25) CONV_CODES_ CONF_CC_EN_TX_STOP	(25) CONV_CODES_ CONF_CC_EN_TX_STOP	0x0	Stop word at the end of the transmission (banked)
		(24) CONV_CODES_ CONF_EN_CONV_CODE	(24) CONV_CODES_ CONF_EN_CONV_CODE	0x0	Convolutional codes (banked)

Address	Register Name	Register Write	Register Read	Default	Description
		(22) PACKET_EXTRA_FIFO_REWIND	(22) PACKET_EXTRA_FIFO_REWIND	0x0	Rewind the FIFO to the initial stage at the end of a Tx transmission (banked)
		(21) PACKET_EXTRA_BLE_PREAMBLE	(21) PACKET_EXTRA_BLE_PREAMBLE	0x1	Handle the preamble directly in Tx mode (PREAMBLE register is not used) according to the BLE standard (banked)
		(20) PACKET_EXTRA_PKT_INFO_PRE_NPOST	(20) PACKET_EXTRA_PKT_INFO_PRE_NPOST	0x0	Packet information sampling (banked)
		(19:18) PACKET_EXTRA_PATTERN_MAX_ERR	(19:18) PACKET_EXTRA_PATTERN_MAX_ERR	0x0	Unsigned value that specifies the maximum number of errors in the pattern recognition (banked)
		(17:16) PACKET_EXTRA_PATTERN_WORD_LEN	(17:16) PACKET_EXTRA_PATTERN_WORD_LEN	0x3	Pattern word length (banked)
		(15:0) ADDRESS_BROADCAST_ADDRESS_BR	(15:0) ADDRESS_BROADCAST_ADDRESS_BR	0x0	Broadcast address (banked)
0x40040C38	RF2_CRC_POLYNOMIAL	(31:0) CRC_POLY	(31:0) CRC_POLY	0x80032D	CRC polynomial (banked)
0x40040C3C	RF2_CRC_RST	(31:0) CRC_RST	(31:0) CRC_RST	0x555555	CRC reset value (banked)
0x40040C40	RF2_REG10	(25:21) CONV_CODES_PUNCT_CC_PUNCT_1	(25:21) CONV_CODES_PUNCT_CC_PUNCT_1	0x1	Puncture of the second convolutional code (banked)
		(20:16) CONV_CODES_PUNCT_CC_PUNCT_0	(20:16) CONV_CODES_PUNCT_CC_PUNCT_0	0x1	Puncture of the first convolutional code (banked)
		(11) FRAC_CONF_TX_FRAC_GAIN	(11) FRAC_CONF_TX_FRAC_GAIN	0x0	Additional gain for fractional data-rates in Tx mode (banked)
		(10) FRAC_CONF_RX_FRAC_GAIN	(10) FRAC_CONF_RX_FRAC_GAIN	0x0	Additional gain for fractional data-rates in Rx mode (banked)

Address	Register Name	Register Write	Register Read	Default	Description
		(9) FRAC_CONF_TX_EN_FRAC	(9) FRAC_CONF_TX_EN_FRAC	0x0	Fractional data-rates in Tx mode (banked)
		(8) FRAC_CONF_RX_EN_FRAC	(8) FRAC_CONF_RX_EN_FRAC	0x0	Fractional data-rates in Rx mode (banked)
		(7:4) CONV_CODES_POLY_CC_POLY_1	(7:4) CONV_CODES_POLY_CC_POLY_1	0xD	Second convolutional code (banked)
		(3:0) CONV_CODES_POLY_CC_POLY_0	(3:0) CONV_CODES_POLY_CC_POLY_0	0xF	First convolutional code (banked)
0x40040C44	RF2_REG11	(31) FILTER_GAIN_LIN_FILTER	(31) FILTER_GAIN_LIN_FILTER	0x0	Enable the linear filtering (banked)
		(30) FILTER_GAIN_LOW_LIN_GAIN	(30) FILTER_GAIN_LOW_LIN_GAIN	0x0	Reduce the total gain by two if the linear gain is set (banked)
		(29:27) FILTER_GAIN_GAIN_M	(29:27) FILTER_GAIN_GAIN_M	0x0	Mantissa of the final stage gain of the matched filter (banked)
		(26:24) FILTER_GAIN_GAIN_E	(26:24) FILTER_GAIN_GAIN_E	0x0	Exponent of the final stage gain of the matched filter (banked)
		(23:20) TX_MULT_TX_MULT_EXP	(23:20) TX_MULT_TX_MULT_EXP	0x2	Exponent of the Tx multiplier (banked)
		(19:16) TX_MULT_TX_MULT_MAN	(19:16) TX_MULT_TX_MULT_MAN	0x9	Mantissa of the Tx multiplier (banked)
		(15:12) TX_FRAC_CONF_TX_FRAC_DEN	(15:12) TX_FRAC_CONF_TX_FRAC_DEN	0x0	Denominator of the fractional data-rate in Tx mode (banked)
		(11:8) TX_FRAC_CONF_TX_FRAC_NUM	(11:8) TX_FRAC_CONF_TX_FRAC_NUM	0x0	Numerator of the fractional data-rate in Tx mode (banked)
		(7:4) RX_FRAC_CONF_RX_FRAC_DEN	(7:4) RX_FRAC_CONF_RX_FRAC_DEN	0x0	Denominator of the fractional data-rate in Rx mode (banked)
		(3:0) RX_FRAC_CONF_RX_FRAC_NUM	(3:0) RX_FRAC_CONF_RX_FRAC_NUM	0x0	Numerator of the fractional data-rate in Rx mode (banked)

Address	Register Name	Register Write	Register Read	Default	Description
0x40040C48	RF2_TX_PULSE_SHAPE_1	(31:24) TX_PULSE_ SHAPE_1_TX_COEF4	(31:24) TX_PULSE_ SHAPE_1_TX_COEF4	0x0	Tx pulse shape coefficient 4 (banked)
		(23:16) TX_PULSE_ SHAPE_1_TX_COEF3	(23:16) TX_PULSE_ SHAPE_1_TX_COEF3	0x0	Tx pulse shape coefficient 3 (banked)
		(15:8) TX_PULSE_ SHAPE_1_TX_COEF2	(15:8) TX_PULSE_ SHAPE_1_TX_COEF2	0x0	Tx pulse shape coefficient 2 (banked)
		(7:0) TX_PULSE_ SHAPE_1_TX_COEF1	(7:0) TX_PULSE_ SHAPE_1_TX_COEF1	0x0	Tx pulse shape coefficient 1 (banked)
0x40040C4C	RF2_TX_PULSE_SHAPE_2	(31:24) TX_PULSE_ SHAPE_2_TX_COEF8	(31:24) TX_PULSE_ SHAPE_2_TX_COEF8	0x2	Tx pulse shape coefficient 8 (banked)
		(23:16) TX_PULSE_ SHAPE_2_TX_COEF7	(23:16) TX_PULSE_ SHAPE_2_TX_COEF7	0x1	Tx pulse shape coefficient 7 (banked)
		(15:8) TX_PULSE_ SHAPE_2_TX_COEF6	(15:8) TX_PULSE_ SHAPE_2_TX_COEF6	0x0	Tx pulse shape coefficient 6 (banked)
		(7:0) TX_PULSE_ SHAPE_2_TX_COEF5	(7:0) TX_PULSE_ SHAPE_2_TX_COEF5	0x0	Tx pulse shape coefficient 5 (banked)
0x40040C50	RF2_TX_PULSE_SHAPE_3	(31:24) TX_PULSE_ SHAPE_3_TX_COEF12	(31:24) TX_PULSE_ SHAPE_3_TX_COEF12	0x36	Tx pulse shape coefficient 12 (banked)
		(23:16) TX_PULSE_ SHAPE_3_TX_COEF11	(23:16) TX_PULSE_ SHAPE_3_TX_COEF11	0x20	Tx pulse shape coefficient 11 (banked)
		(15:8) TX_PULSE_ SHAPE_3_TX_COEF10	(15:8) TX_PULSE_ SHAPE_3_TX_COEF10	0x10	Tx pulse shape coefficient 10 (banked)
		(7:0) TX_PULSE_ SHAPE_3_TX_COEF9	(7:0) TX_PULSE_ SHAPE_3_TX_COEF9	0x7	Tx pulse shape coefficient 9 (banked)
0x40040C54	RF2_TX_PULSE_SHAPE_4	(31:24) TX_PULSE_ SHAPE_4_TX_COEF16	(31:24) TX_PULSE_ SHAPE_4_TX_COEF16	0x7D	Tx pulse shape coefficient 16 (banked)
		(23:16) TX_PULSE_ SHAPE_4_TX_COEF15	(23:16) TX_PULSE_ SHAPE_4_TX_COEF15	0x75	Tx pulse shape coefficient 15 (banked)

Address	Register Name	Register Write	Register Read	Default	Description
		(15:8) TX_PULSE_ SHAPE_4_TX_COEF14	(15:8) TX_PULSE_ SHAPE_4_TX_COEF14	0x66	Tx pulse shape coefficient 14 (banked)
		(7:0) TX_PULSE_ SHAPE_4_TX_COEF13	(7:0) TX_PULSE_ SHAPE_4_TX_COEF13	0x4F	Tx pulse shape coefficient 13 (banked)
0x40040C58	RF2_FRONTEND	(25:16) RX_IF_DIG_ IF_DIG	(25:16) RX_IF_DIG_ IF_DIG	0x40	IF frequency (banked)
		(14:11) FRONTEND_ RESAMPLE_PH_GAIN	(14:11) FRONTEND_ RESAMPLE_PH_GAIN	0x6	Gain of the phase resampling block (banked)
		(10:8) FRONTEND_ RESAMPLE_RSSI_G2	(10:8) FRONTEND_ RESAMPLE_RSSI_G2	0x0	Gain of the decimator in the RSSI resampling block (banked)
		(7:6) FRONTEND_ RESAMPLE_RSSI_G1	(7:6) FRONTEND_ RESAMPLE_RSSI_G1	0x0	Gain of the interpolator in the RSSI resampling block (banked)
		(5) FRONTEND_EN_ RESAMPLE_RSSI	(5) FRONTEND_EN_ RESAMPLE_RSSI	0x0	RSSI resampling (banked)
		(4) FRONTEND_EN_ RESAMPLE_PHADC	(4) FRONTEND_EN_ RESAMPLE_PHADC	0x1	Phase resampling (banked)
		(3:0) FRONTEND_DIV_ PHADC	(3:0) FRONTEND_ DIV_PHADC	0x0	Unsigned value that specifies the divider to obtain the phase ADC clock and RSSI (banked)
0x40040C5C	RF2_RX_PULSE_SHAPE	(31:28) RX_PULSE_ SHAPE_RX_COEF8	(31:28) RX_PULSE_ SHAPE_RX_COEF8	0xF	Rx pulse shape coefficient 8 (banked)
		(27:24) RX_PULSE_ SHAPE_RX_COEF7	(27:24) RX_PULSE_ SHAPE_RX_COEF7	0xE	Rx pulse shape coefficient 7 (banked)
		(23:20) RX_PULSE_ SHAPE_RX_COEF6	(23:20) RX_PULSE_ SHAPE_RX_COEF6	0xC	Rx pulse shape coefficient 6 (banked)
		(19:16) RX_PULSE_ SHAPE_RX_COEF5	(19:16) RX_PULSE_ SHAPE_RX_COEF5	0xA	Rx pulse shape coefficient 5 (banked)
		(15:12) RX_PULSE_	(15:12) RX_PULSE_	0x7	Rx pulse shape coefficient 4 (banked)

Address	Register Name	Register Write	Register Read	Default	Description
		SHAPE_RX_COEF4	SHAPE_RX_COEF4		
		(11:8) RX_PULSE_ SHAPE_RX_COEF3	(11:8) RX_PULSE_ SHAPE_RX_COEF3	0x4	Rx pulse shape coefficient 3 (banked)
		(7:4) RX_PULSE_ SHAPE_RX_COEF2	(7:4) RX_PULSE_ SHAPE_RX_COEF2	0x2	Rx pulse shape coefficient 2 (banked)
		(3:0) RX_PULSE_ SHAPE_RX_COEF1	(3:0) RX_PULSE_ SHAPE_RX_COEF1	0x1	Rx pulse shape coefficient 1 (banked)
0x40040C60	RF2_REG18	(28) DELAY_LINE_ CONF_MULTI_SYNC	(28) DELAY_LINE_ CONF_MULTI_SYNC	0x0	Detect multiple syncs (banked)
		(27:25) DELAY_LINE_ CONF_DL_ISI_THR	(27:25) DELAY_ LINE_CONF_DL_ISI_ THR	0x1	Threshold bias for ISI compensation in the delay line sync word comparator (banked)
		(22) DELAY_LINE_ CONF_EN_SYNC_OK_ DELAY_LINE	(22) DELAY_LINE_ CONF_EN_SYNC_OK_ DELAY_LINE	0x1	Use pattern_ok signal in delay line to synchronize the deserializer (banked)
		(21:20) DELAY_LINE_ CONF_MAX_ERR_IN_DL_ SYNC	(21:20) DELAY_ LINE_CONF_MAX_ERR_ IN_DL_SYNC	0x0	Set the maximum errors in the delay line sync detection (banked)
		(19) DELAY_LINE_ CONF_EN_NOT_CAUSAL	(19) DELAY_LINE_ CONF_EN_NOT_CAUSAL	0x0	Non causal processing (banked)
		(18:16) DELAY_LINE_ CONF_NC_SEL_OUT	(18:16) DELAY_ LINE_CONF_NC_SEL_ OUT	0x0	Select the output position for the non causal processing (banked)
		(15:8) FSK_FCR_AMP_ 1_FSK_FCR_AMP1	(15:8) FSK_FCR_ AMP_1_FSK_FCR_AMP1	0x1B	FSK amplitude low (banked)
		(6:4) CARRIER_ RECOVERY_EXTRA_ FREQ_LIMIT_MAN	(6:4) CARRIER_ RECOVERY_EXTRA_ FREQ_LIMIT_MAN	0x5	Mantissa of the carrier recovery frequency limit (banked)
		(2:0) CARRIER_	(2:0) CARRIER_	0x0	Exponent of the carrier recovery

Address	Register Name	Register Write	Register Read	Default	Description
		RECOVERY_EXTRA_ FREQ_LIMIT_EXP	RECOVERY_EXTRA_ FREQ_LIMIT_EXP		frequency limit (banked)
0x40040C64	RF2_REG19	(30) RSSI_BANK_EN_ RSSI_DITHER	(30) RSSI_BANK_EN_ RSSI_DITHER	0x0	Speed on the RSSI triangular dithering signal (banked)
		(29) RSSI_BANK_ FAST_RSSI	(29) RSSI_BANK_ FAST_RSSI	0x0	RSSI filtering speed (banked)
		(28) RSSI_BANK_EN_ FAST_PRE_SYNC	(28) RSSI_BANK_EN_ FAST_PRE_SYNC	0x1	Fast mode switching during the preamble reception (banked)
		(27:24) RSSI_BANK_ TAU_RSSI_FILTERING	(27:24) RSSI_BANK_ TAU_RSSI_FILTERING	0x1	Time constant of the RSSI filtering block (banked)
		(20) DECISION_USE_ VIT_SOFT	(20) DECISION_USE_ VIT_SOFT	0x0	Viterbi soft decoding (banked)
		(19:18) DECISION_ VITERBI_LEN	(19:18) DECISION_ VITERBI_LEN	0x2	Set the Viterbi path length (banked)
		(17) DECISION_ VITERBI_POW_NLIN	(17) DECISION_ VITERBI_POW_NLIN	0x1	Viterbi algorithm uses power instead of amplitude to evaluate the error on the path (banked)
		(16) DECISION_EN_ VITERBI_GFSK	(16) DECISION_EN_ VITERBI_GFSK	0x1	Viterbi algorithm for the GFSK decoding (banked)
		(15:8) FSK_FCR_AMP_ 3_FSK_FCR_AMP3	(15:8) FSK_FCR_ AMP_3_FSK_FCR_AMP3	0x44	FSK amplitude high (banked)
		(7:0) FSK_FCR_AMP_ 2_FSK_FCR_AMP2	(7:0) FSK_FCR_AMP_ 2_FSK_FCR_AMP2	0x30	FSK amplitude mid (banked)
0x40040C68	RF2_REG1A	(28:24) PA_PWR_PA_ PWR	(28:24) PA_PWR_PA_ PWR	0xC	Signed value that sets the PA power
		(22) RSSI_BANK_ALT_ USE_RSSI_ALT	(22) RSSI_BANK_ ALT_USE_RSSI_ALT	0x0	Use alternative RRSI configuration (banked)
		(21) RSSI_BANK_ALT_	(21) RSSI_BANK_	0x0	RSSI filtering speed (banked)

Address	Register Name	Register Write	Register Read	Default	Description
		FAST_RSSI_ALT	ALT_FAST_RSSI_ALT		
		(19:16) RSSI_BANK_ ALT_TAU_RSSI_ FILTERING_ALT	(19:16) RSSI_BANK_ ALT_TAU_RSSI_ FILTERING_ALT	0x3	Time constant of the RSSI filtering block (banked)
		(15:0) CORRECT_ CFREQ_IF_CORRECT_ CFREQ_IF	(15:0) CORRECT_ CFREQ_IF_CORRECT_ CFREQ_IF	0x1555	Unsigned value that specifies the IF for the Rx mode (banked)
0x40040C6C	RF2_REG1B	(31) PLL_BANK_EN_ LOW_CHP_BIAS_TX	(31) PLL_BANK_EN_ LOW_CHP_BIAS_TX	0x0	Set the en_low_chp_bias bit in Tx mode (banked)
		(30) PLL_BANK_EN_ LOW_CHP_BIAS_RX	(30) PLL_BANK_EN_ LOW_CHP_BIAS_RX	0x1	Set the en_low_chp_bias bit in Rx mode (banked)
		(29:28) PLL_BANK_ PLL_FILTER_RES_ TRIM_TX	(29:28) PLL_BANK_ PLL_FILTER_RES_ TRIM_TX	0x3	Modify the value of the loop filter resistor R2 when bit 5 is high in Tx mode (banked)
		(27:24) PLL_BANK_ IQ_PLL_0_TX	(27:24) PLL_BANK_ IQ_PLL_0_TX	0x4	Charge pump bias for Tx case (banked)
		(22) PLL_BANK_LOW_ DR_TX	(22) PLL_BANK_LOW_ DR_TX	0x0	Enable low data-rate mode in Tx mode (banked)
		(21:20) PLL_BANK_ PLL_FILTER_RES_ TRIM_RX	(21:20) PLL_BANK_ PLL_FILTER_RES_ TRIM_RX	0x0	Modify the value of the loop filter resistor R2 when bit 5 is high in Rx mode (banked)
		(19:16) PLL_BANK_ IQ_PLL_0_RX	(19:16) PLL_BANK_ IQ_PLL_0_RX	0xB	Charge pump bias for Rx (banked)
		(15) ANACLK_USE_ NEW_ANACK	(15) ANACLK_USE_ NEW_ANACK	0x0	Use the new analog clock generator (banked)
		(13:12) ANACLK_DIV_ CK_RSSI	(13:12) ANACLK_ DIV_CK_RSSI	0x0	Set the master clock divider for the RSSI clock (banked)
		(11:10) ANACLK_DIV_ CK_FILT	(11:10) ANACLK_ DIV_CK_FILT	0x0	Set the master clock divider for the channel filter clock (banked)

Address	Register Name	Register Write	Register Read	Default	Description
		(9:8) ANACLK_DIV_CK_PHADC	(9:8) ANACLK_DIV_CK_PHADC	0x0	Set the master clock divider for the phase ADC clock (banked)
		(7:4) ANACLK_DIV_RSSI	(7:4) ANACLK_DIV_RSSI	0x1	Unsigned value that specifies the division factor for the clock controlling the RSSI (banked)
		(3:0) ANACLK_DIV_FILT	(3:0) ANACLK_DIV_FILT	0x5	Unsigned value that specifies the division factor for the clock controlling the channel filter (banked)
0x40040C70	RF2_RSSI_CTRL	(31:30) RSSI_CTRL_AGC_DECAY_TAU	(31:30) RSSI_CTRL_AGC_DECAY_TAU	0x3	Time constant of the decay speed
		(29) RSSI_CTRL_AGC_USE_LNA	(29) RSSI_CTRL_AGC_USE_LNA	0x1	AGC algorithm uses LNA bias
		(28) RSSI_CTRL_AGC_MODE	(28) RSSI_CTRL_AGC_MODE	0x1	AGC algorithm selection
		(27:26) RSSI_CTRL_AGC_WAIT	(27:26) RSSI_CTRL_AGC_WAIT	0x3	Set the wait time of the AGC after switching between state
		(25) RSSI_CTRL_PAYLOAD_BLOCKS_AGC	(25) RSSI_CTRL_PAYLOAD_BLOCKS_AGC	0x1	AGC payload blocking
		(24) RSSI_CTRL_BYPASS_AGC	(24) RSSI_CTRL_BYPASS_AGC	0x0	AGC algorithm bypass
		(20:16) PA_PWR_OFFSET_PA_PWR_OFFSET	(20:16) PA_PWR_OFFSET_PA_PWR_OFFSET	0x0	Signed value that sets the PA power (banked)
		(12:8) FILTER_BIAS_IQ_FI_BW	(12:8) FILTER_BIAS_IQ_FI_BW	0x14	Bias for the bandwidth of the channel filter (banked)
		(4:0) FILTER_BIAS_IQ_FI_FC	(4:0) FILTER_BIAS_IQ_FI_FC	0xB	Bias for the central frequency of the channel filter (banked)
0x40040C74	RF2_REG1D	(31:28) AGC_PEAK_DET_PEAK_DET_TAU	(31:28) AGC_PEAK_DET_PEAK_DET_TAU	0x7	Time constant of the peak detector monostable circuit

Address	Register Name	Register Write	Register Read	Default	Description
		(27:26) AGC_PEAK_DET_PEAK_DET_THR_LOW	(27:26) AGC_PEAK_DET_PEAK_DET_THR_LOW	0x0	Threshold for the low level of the peak detector
		(25) AGC_PEAK_DET_PEAK_DET_THR_HIGH	(25) AGC_PEAK_DET_PEAK_DET_THR_HIGH	0x0	Threshold for the high level of the peak detector
		(24) AGC_PEAK_DET_EN_AGC_PEAK	(24) AGC_PEAK_DET_EN_AGC_PEAK	0x1	Enable AGC peak detector
		(23:16) AGC_THR_HIGH_AGC_THR_HIGH	(23:16) AGC_THR_HIGH_AGC_THR_HIGH	0x69	AGC threshold high level (banked)
		(15:8) AGC_THR_LOW_AGC_THR_LOW	(15:8) AGC_THR_LOW_AGC_THR_LOW	0x40	AGC threshold low level (banked)
		(7:4) ATT_CTRL_ATT_CTRL_MAX	(7:4) ATT_CTRL_ATT_CTRL_MAX	0xB	Maximum attenuation level in AGC algorithm
		(3:0) ATT_CTRL_SET_RX_ATT_CTRL	(3:0) ATT_CTRL_SET_RX_ATT_CTRL	0x0	Attenuation level if the AGC is bypassed
0x40040C78	RF2_AGC_LUT1	(31:22) AGC_LUT_1_AGC_LEVEL_2_LO	(31:22) AGC_LUT_1_AGC_LEVEL_2_LO	0x280	AGC values level 2 (LSB)
		(21:11) AGC_LUT_1_AGC_LEVEL_1	(21:11) AGC_LUT_1_AGC_LEVEL_1	0x80	AGC values level 1
		(10:0) AGC_LUT_1_AGC_LEVEL_0	(10:0) AGC_LUT_1_AGC_LEVEL_0	0x0	AGC values level 0
0x40040C7C	RF2_AGC_LUT2	(31:23) AGC_LUT_2_AGC_LEVEL_5_LO	(31:23) AGC_LUT_2_AGC_LEVEL_5_LO	0x84	AGC values level 5 (LSB)
		(22:12) AGC_LUT_2_AGC_LEVEL_4	(22:12) AGC_LUT_2_AGC_LEVEL_4	0x284	AGC values level 4
		(11:1) AGC_LUT_2_AGC_LEVEL_3	(11:1) AGC_LUT_2_AGC_LEVEL_3	0x480	AGC values level 3
		(0) AGC_LUT_2_AGC_	(0) AGC_LUT_2_AGC_	0x0	AGC values level 2 (MSB)

Address	Register Name	Register Write	Register Read	Default	Description
		LEVEL_2_HI	LEVEL_2_HI		
0x40040C80	RF2_AGC_LUT3	(31:24) AGC_LUT_3_ AGC_LEVEL_8_LO	(31:24) AGC_LUT_3_ AGC_LEVEL_8_LO	0x9D	AGC values level 8 (LSB)
		(23:13) AGC_LUT_3_ AGC_LEVEL_7	(23:13) AGC_LUT_3_ AGC_LEVEL_7	0x495	AGC values level 7
		(12:2) AGC_LUT_3_ AGC_LEVEL_6	(12:2) AGC_LUT_3_ AGC_LEVEL_6	0x485	AGC values level 6
		(1:0) AGC_LUT_3_ AGC_LEVEL_5_HI	(1:0) AGC_LUT_3_ AGC_LEVEL_5_HI	0x2	AGC values level 5 (MSB)
0x40040C84	RF2_AGC_LUT4	(31:25) AGC_LUT_4_ AGC_LEVEL_11_LO	(31:25) AGC_LUT_4_ AGC_LEVEL_11_LO	0x7F	AGC values level 11 (LSB)
		(24:14) AGC_LUT_4_ AGC_LEVEL_10	(24:14) AGC_LUT_4_ AGC_LEVEL_10	0x4FF	AGC values level 10
		(13:3) AGC_LUT_4_ AGC_LEVEL_9	(13:3) AGC_LUT_4_ AGC_LEVEL_9	0x49F	AGC values level 9
		(2:0) AGC_LUT_4_ AGC_LEVEL_8_HI	(2:0) AGC_LUT_4_ AGC_LEVEL_8_HI	0x4	AGC values level 8 (MSB)
0x40040C88	RF2_AGC_LUT5	(26:25) IEEE802154_ OPTS_CNT_LIM_802154	(26:25) IEEE802154_OPTS_ CNT_LIM_802154	0x2	Set the number of samples to wait before increasing the threshold
		(24:22) IEEE802154_ OPTS_CNT_OK_INC_ 802154	(24:22) IEEE802154_OPTS_ CNT_OK_INC_802154	0x4	Set the increment to the counter that indicates that the correlators peaks are coherent
		(21) IEEE802154_ OPTS_USE_OS_802154	(21) IEEE802154_ OPTS_USE_OS_802154	0x1	Enable the new algorithm working in the oversampled domain for the demodulation of the IEEE 802.15.4 protocol
		(20) IEEE802154_ OPTS_EN_DW_TEST	(20) IEEE802154_ OPTS_EN_DW_TEST	0x0	Tx data-whitening before the convolutional code block

Address	Register Name	Register Write	Register Read	Default	Description
		(18:16) IEEE802154_OPTS_C2B_THR	(18:16) IEEE802154_OPTS_C2B_THR	0x4	Threshold of the chip2bit correlator of the IEEE 802.15.4 protocol
		(13:12) DATA_STREAMS_BER_CLK_MODE	(13:12) DATA_STREAMS_BER_CLK_MODE	0x0	Set the clock output mode for BER mode or RW mode
		(10) DATA_STREAMS_RX_DATA_NOT_SAMPLED	(10) DATA_STREAMS_RX_DATA_NOT_SAMPLED	0x0	Signal rx_data in test modes sampling
		(9) DATA_STREAMS_PHASE_GREY	(9) DATA_STREAMS_PHASE_GREY	0x0	Phase signal encoding
		(8) DATA_STREAMS_TX_IN_CLK_TOGGLE	(8) DATA_STREAMS_TX_IN_CLK_TOGGLE	0x0	Input clock
		(3:0) AGC_LUT_5_AGC_LEVEL_11_HI	(3:0) AGC_LUT_5_AGC_LEVEL_11_HI	0xE	AGC values level 11 (MSB)
0x40040C8C	RF2_AGC_ATT1	(31:30) AGC_ATT_1_AGC_ATT_AB_LO	(31:30) AGC_ATT_1_AGC_ATT_AB_LO	0x3	AGC attenuation step 10/11 (LSB)
		(29:27) AGC_ATT_1_AGC_ATT_9A	(29:27) AGC_ATT_1_AGC_ATT_9A	0x5	AGC attenuation step 9/10
		(26:24) AGC_ATT_1_AGC_ATT_89	(26:24) AGC_ATT_1_AGC_ATT_89	0x3	AGC attenuation step 8/9
		(23:21) AGC_ATT_1_AGC_ATT_78	(23:21) AGC_ATT_1_AGC_ATT_78	0x4	AGC attenuation step 7/8
		(20:18) AGC_ATT_1_AGC_ATT_67	(20:18) AGC_ATT_1_AGC_ATT_67	0x3	AGC attenuation step 6/7
		(17:15) AGC_ATT_1_AGC_ATT_56	(17:15) AGC_ATT_1_AGC_ATT_56	0x2	AGC attenuation step 5/6
		(14:12) AGC_ATT_1_AGC_ATT_45	(14:12) AGC_ATT_1_AGC_ATT_45	0x2	AGC attenuation step 4/5

Address	Register Name	Register Write	Register Read	Default	Description
		(11:9) AGC_ATT_1_AGC_ATT_34	(11:9) AGC_ATT_1_AGC_ATT_34	0x2	AGC attenuation step 3/4
		(8:6) AGC_ATT_1_AGC_ATT_23	(8:6) AGC_ATT_1_AGC_ATT_23	0x1	AGC attenuation step 2/3
		(5:3) AGC_ATT_1_AGC_ATT_12	(5:3) AGC_ATT_1_AGC_ATT_12	0x1	AGC attenuation step 1/2
		(2:0) AGC_ATT_1_AGC_ATT_01	(2:0) AGC_ATT_1_AGC_ATT_01	0x4	AGC attenuation step 0/1
0x40040C90	RF2_AGC_ATT2	(31:28) TIMINGS_3_T_DLL	(31:28) TIMINGS_3_T_DLL	0x2	Time needed by the DLL blocks to switch on
		(27:24) TIMINGS_3_T_PLL_TX	(27:24) TIMINGS_3_T_PLL_TX	0x2	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	0xC	Time needed by the subband algorithm to calibrate in Tx mode
		(19:16) TIMINGS_2_T_TX_RF	(19:16) TIMINGS_2_T_TX_RF	0x1	Time needed by the RF blocks to switch on in Tx mode
		(14:12) TIMINGS_1_T_GRANULARITY_TX	(14:12) TIMINGS_1_T_GRANULARITY_TX	0x3	Define the granularity of the timer in Tx mode
		(10:8) TIMINGS_1_T_GRANULARITY_RX	(10:8) TIMINGS_1_T_GRANULARITY_RX	0x5	Define the granularity of the timer in Rx mode
		(1) AGC_ATT_2_AGC_ATT_1DB	(1) AGC_ATT_2_AGC_ATT_1DB	0x0	Attenuation steps
		(0) AGC_ATT_2_AGC_ATT_AB_HI	(0) AGC_ATT_2_AGC_ATT_AB_HI	0x1	AGC attenuation step 10/11 (MSB)
0x40040C94	RF2_REG25	(31) TIMEOUT_EN_RX_TIMEOUT	(31) TIMEOUT_EN_RX_TIMEOUT	0x0	Timeout of the Rx when the system is on FSM mode
		(30:28) TIMEOUT_T_TIMEOUT_GR	(30:28) TIMEOUT_T_TIMEOUT_GR	0x0	Granularity of the timer in timeout Rx mode

Address	Register Name	Register Write	Register Read	Default	Description
		(27:24) TIMEOUT_T_RX_TIMEOUT	(27:24) TIMEOUT_T_RX_TIMEOUT	0x0	Time that has to occur before the timeout
		(21) TIMING_FAST_RX_EN_FAST_RX_TXFILT	(21) TIMING_FAST_RX_EN_FAST_RX_TXFILT	0x0	Filter Tx configuration for the fast Rx PLL
		(20) TIMING_FAST_RX_EN_FAST_RX	(20) TIMING_FAST_RX_EN_FAST_RX	0x0	Fast Rx PLL
		(19:16) TIMING_FAST_RX_T_RX_FAST_CHP	(19:16) TIMING_FAST_RX_T_RX_FAST_CHP	0x0	Time to switch off the fast CHP in Rx mode
		(15:12) TIMINGS_5_T_RX_RF	(15:12) TIMINGS_5_T_RX_RF	0x0	Time needed by the RF blocks to switch on in Rx mode
		(11:8) TIMINGS_5_T_RX_BB	(11:8) TIMINGS_5_T_RX_BB	0x1	Time needed by the BB blocks to switch on in Rx mode
		(7:4) TIMINGS_4_T_SUBBAND_RX	(7:4) TIMINGS_4_T_SUBBAND_RX	0x5	Time needed by the subband algorithm to calibrate in Rx mode
		(3:0) TIMINGS_4_T_PLL_RX	(3:0) TIMINGS_4_T_PLL_RX	0x1	Time needed by the PLL blocks to switch on in Rx mode
0x40040C98	RF2_BIAS_0_2	(31:28) BIAS_2_IQ_RXTX_6	(31:28) BIAS_2_IQ_RXTX_6	0x3	VCOM_MX bias
		(27:24) BIAS_2_IQ_RXTX_5	(27:24) BIAS_2_IQ_RXTX_5	0x8	VCOM_LO bias
		(23:20) BIAS_1_IQ_RXTX_3	(23:20) BIAS_1_IQ_RXTX_3	0x6	PrePA Casc bias
		(19:16) BIAS_1_IQ_RXTX_2	(19:16) BIAS_1_IQ_RXTX_2	0x6	PrePA In bias
		(15:12) BIAS_0_IQ_RXTX_1	(15:12) BIAS_0_IQ_RXTX_1	0x7	PA backoff bias

Address	Register Name	Register Write	Register Read	Default	Description
		(11:8) BIAS_0_IQ_ RXTX_0	(11:8) BIAS_0_IQ_ RXTX_0	0x3	PA bias
		(7) INTERFACE_CONF_ EN_SYNC_IFACE	(7) INTERFACE_ CONF_EN_SYNC_IFACE	0x0	Interfaces resynchronization
		(6:4) INTERFACE_ CONF_APB_WAIT_STATE	(6:4) INTERFACE_ CONF_APB_WAIT_ STATE	0x0	Select the number of wait states during the APB transaction
		(1:0) INTERFACE_ CONF_SPI_SELECT	(1:0) INTERFACE_ CONF_SPI_SELECT	0x0	Select the SPI mode
0x40040C9C	RF2_BIAS_3_6	(31:28) BIAS_6_IQ_ BB_0	(31:28) BIAS_6_IQ_ BB_0	0x7	ACD_O bias
		(27:24) BIAS_6_IQ_ PLL_3	(27:24) BIAS_6_IQ_ PLL_3	0x7	DLL bias
		(23:20) BIAS_5_IQ_ PLL_4_RX	(23:20) BIAS_5_IQ_ PLL_4_RX	0x8	VCO bias for Rx mode
		(19:16) BIAS_5_IQ_ PLL_4_TX	(19:16) BIAS_5_IQ_ PLL_4_TX	0xA	VCO bias for Tx mode
		(15:12) BIAS_4_IQ_ PLL_2	(15:12) BIAS_4_IQ_ PLL_2	0x7	Sub-band comparator bias
		(11:8) BIAS_4_IQ_ PLL_1	(11:8) BIAS_4_IQ_ PLL_1	0x4	Dynamic divider bias
		(7:4) BIAS_3_IQ_ RXTX_8	(7:4) BIAS_3_IQ_ RXTX_8	0x7	IFA ctrl_c bias
		(3:0) BIAS_3_IQ_ RXTX_7	(3:0) BIAS_3_IQ_ RXTX_7	0x7	IFA ctrl_r bias
0x40040CA0	RF2_BIAS_7_9	(31:28) BIAS_9_IQ_ BB_6	(31:28) BIAS_9_IQ_ BB_6	0x9	Peak detector threshold bias 0
		(27:24) BIAS_9_IQ_	(27:24) BIAS_9_IQ_	0x5	Peak detector bias

Address	Register Name	Register Write	Register Read	Default	Description
		BB_5	BB_5		
		(23:20) SWCAP_FSM_SB_CAP_RX	(23:20) SWCAP_FSM_SB_CAP_RX	0x0	VCO subband selection (FSM in Rx mode)
		(19:16) SWCAP_FSM_SB_CAP_TX	(19:16) SWCAP_FSM_SB_CAP_TX	0x0	VCO subband selection (FSM in Tx mode)
		(15:12) BIAS_8_IQ_BB_4	(15:12) BIAS_8_IQ_BB_4	0x9	RSSI_D bias
		(11:8) BIAS_8_IQ_BB_3	(11:8) BIAS_8_IQ_BB_3	0xF	RSSI_G bias
		(7:4) BIAS_7_IQ_BB_2	(7:4) BIAS_7_IQ_BB_2	0x6	ACD_L bias
		(3:0) BIAS_7_IQ_BB_1	(3:0) BIAS_7_IQ_BB_1	0x6	ACD_C bias
0x40040CA4	RF2_BIAS_10_12	(30) SD_MASH_MASH_DITHER_TYPE	(30) SD_MASH_MASH_DITHER_TYPE	0x0	Enable the new dithering scheme
		(29) SD_MASH_MASH_ENABLE	(29) SD_MASH_MASH_ENABLE	0x0	Enable the sigma delta mash
		(28) SD_MASH_MASH_DITHER	(28) SD_MASH_MASH_DITHER	0x1	Enable dithering on the sigma delta mash
		(27:25) SD_MASH_MASH_ORDER	(27:25) SD_MASH_MASH_ORDER	0x3	Order of the sigma delta mash
		(24) SD_MASH_MASH_RSTB	(24) SD_MASH_MASH_RSTB	0x1	Reset of the sigma delta mash (active low)
		(23:20) BIAS_12_LNA_AGC_BIAS_3	(23:20) BIAS_12_LNA_AGC_BIAS_3	0x6	LNA bias for AGC level 3
		(19:16) BIAS_12_LNA_AGC_BIAS_2	(19:16) BIAS_12_LNA_AGC_BIAS_2	0x7	LNA bias for AGC level 2
		(15:12) BIAS_11_	(15:12) BIAS_11_	0x8	LNA bias for AGC level 1

Address	Register Name	Register Write	Register Read	Default	Description
		LNA_AGC_BIAS_1	LNA_AGC_BIAS_1		
		(11:8) BIAS_11_LNA_AGC_BIAS_0	(11:8) BIAS_11_LNA_AGC_BIAS_0	0x9	LNA bias for AGC level 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	0x6	Peak detector threshold bias 2
0x40040CA8	RF2_REG2A	(27:24) SD_MASH_MASK_MASH_MASK	(27:24) SD_MASH_MASK_MASH_MASK	0x0	Mask the n LSB of the fractional part of the MASH (debug only)
		(19) BIAS_EN_2_EN_PTAT	(19) BIAS_EN_2_EN_PTAT	0x1	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)
0x40040CAC	RF2_PLL_CTRL	(26) PLL_CTRL_DISABLE_CHP_SBS	(26) PLL_CTRL_DISABLE_CHP_SBS	0x0	Charge-pump disabling during sub-band selection (FLL and frequency ratios)
		(25) PLL_CTRL_PLL_RX_48MEG	(25) PLL_CTRL_PLL_RX_48MEG	0x1	PLL frequency
		(24) PLL_CTRL_SWCAP_TX_SAME_RX	(24) PLL_CTRL_SWCAP_TX_SAME_RX	0x0	Registers for Rx and Tx modes swcap in case of swcap_fsm=1
		(23) PLL_CTRL_SWCAP_FSM	(23) PLL_CTRL_SWCAP_FSM	0x1	Selection of the swcap_fsm register

Address	Register Name	Register Write	Register Read	Default	Description
		(22) PLL_CTRL_DLL_RSTB	(22) PLL_CTRL_DLL_RSTB	0x1	Reset signal of the DLL (active low)
		(21:18) PLL_CTRL_VCO_SUBBAND_TRIM	(21:18) PLL_CTRL_VCO_SUBBAND_TRIM	0x0	VCO sub-band selection bits
		(17) PLL_CTRL_SUB_SEL_OFFS_EN	(17) PLL_CTRL_SUB_SEL_OFFS_EN	0x0	Add offset to sub-band selection comparator
		(16) PLL_CTRL_DIV2_CLKVCO_TEST_EN	(16) PLL_CTRL_DIV2_CLKVCO_TEST_EN	0x0	VCO signal divided by the programmable divider
		(15) PLL_CTRL_VCODIV_CLK_TEST_EN	(15) PLL_CTRL_VCODIV_CLK_TEST_EN	0x0	Output on GPIO the VCO signal divided by the programmable divider
		(13) PLL_CTRL_CHP_DEAD_ZONE_EN	(13) PLL_CTRL_CHP_DEAD_ZONE_EN	0x0	Charge-pump dead zone
		(12:11) PLL_CTRL_CHP_CURR_OFF_TRIM_TX	(12:11) PLL_CTRL_CHP_CURR_OFF_TRIM_TX	0x3	Charge-pump offset current values selection bits in Tx mode
		(10:9) PLL_CTRL_CHP_CURR_OFF_TRIM_RX	(10:9) PLL_CTRL_CHP_CURR_OFF_TRIM_RX	0x3	Charge-pump offset current values selection bits in Rx mode
		(8) PLL_CTRL_HIGH_BW_FILTER_EN_TX	(8) PLL_CTRL_HIGH_BW_FILTER_EN_TX	0x1	PLL filter high bandwidth needed in Tx mode
		(7) PLL_CTRL_HIGH_BW_FILTER_EN_RX	(7) PLL_CTRL_HIGH_BW_FILTER_EN_RX	0x0	PLL filter high bandwidth needed in Rx mode
		(6) PLL_CTRL_FAST_CHP_EN_TX	(6) PLL_CTRL_FAST_CHP_EN_TX	0x1	High current output of the charge-pump for PLL Tx high bandwidth mode
		(5) PLL_CTRL_FAST_CHP_EN_RX	(5) PLL_CTRL_FAST_CHP_EN_RX	0x0	High current output of the charge-pump for PLL Rx high bandwidth mode
		(4:3) PLL_CTRL_CHP_MODE_TRIM	(4:3) PLL_CTRL_CHP_MODE_TRIM	0x0	Select the frequency inside sub-band selection

Address	Register Name	Register Write	Register Read	Default	Description
		(2) PLL_CTRL_CHP_CMC_EN	(2) PLL_CTRL_CHP_CMC_EN	0x1	Common mode control block of the charge-pump
		(1) PLL_CTRL_CHP_CURR_OFF_EN_TX	(1) PLL_CTRL_CHP_CURR_OFF_EN_TX	0x1	Charge-pump offset current in Tx mode
		(0) PLL_CTRL_CHP_CURR_OFF_EN_RX	(0) PLL_CTRL_CHP_CURR_OFF_EN_RX	0x0	Charge-pump offset current in Rx mode
0x40040CB0	RF2_DLL_CTRL	(31:29) RSSI_TUN_1_RSSI_TUN_GAIN	(31:29) RSSI_TUN_1_RSSI_TUN_GAIN	0x1	RSSI tuning for gain
		(28:24) RSSI_TUN_1_RSSI_ODD_OFFSET	(28:24) RSSI_TUN_1_RSSI_ODD_OFFSET	0x4	RSSI tuning for odd stages (offset to the even triangular wave)
		(23:20) RSSI_TUN_1_RSSI_EVEN_MAX	(23:20) RSSI_TUN_1_RSSI_EVEN_MAX	0x1	RSSI tuning for even stages (maximum value of the triangular wave)
		(19:16) RSSI_TUN_1_RSSI_EVEN_MIN	(19:16) RSSI_TUN_1_RSSI_EVEN_MIN	0x1	RSSI tuning for even stages (minimum value of the triangular wave)
		(12) DLL_CTRL_CK_LAST_SEL_DELAY	(12) DLL_CTRL_CK_LAST_SEL_DELAY	0x0	Last SEL delay
		(11) DLL_CTRL_CK_FIRST_SEL_DELAY	(11) DLL_CTRL_CK_FIRST_SEL_DELAY	0x0	First SEL delay
		(10) DLL_CTRL_CK_EXT_SEL	(10) DLL_CTRL_CK_EXT_SEL	0x0	Input clock selection
		(9) DLL_CTRL_CK_DIG_EN	(9) DLL_CTRL_CK_DIG_EN	0x0	Alternate ck_dig pin to output the PLL reference clock signal
		(8) DLL_CTRL_CK_TEST_EN	(8) DLL_CTRL_CK_TEST_EN	0x0	Output on GPIO the PLL reference clock signal via ck_test pin
		(7) DLL_CTRL_TOO_FAST_ENB	(7) DLL_CTRL_TOO_FAST_ENB	0x0	Lock range phase detector
		(6) DLL_CTRL_LOCKED_DET_EN	(6) DLL_CTRL_LOCKED_DET_EN	0x1	Reference frequency multiplier locked detector

Address	Register Name	Register Write	Register Read	Default	Description
		(5) DLL_CTRL_LOCKED_AUTO_CHECK_EN	(5) DLL_CTRL_LOCKED_AUTO_CHECK_EN	0x1	Frequency multiplier is out of lock (usually because some input clocks from ck_xtal or ck_ext are missing)
		(4) DLL_CTRL_FAST_ENB	(4) DLL_CTRL_FAST_ENB	0x0	Disable fast mode locking of the reference frequency multiplier
		(3:2) DLL_CTRL_CK_SEL_TX	(3:2) DLL_CTRL_CK_SEL_TX	0x1	Selection of the clock used as frequency reference of the PLL in Tx mode (also to ck_test and ck_dig outputs)
		(1:0) DLL_CTRL_CK_SEL_RX	(1:0) DLL_CTRL_CK_SEL_RX	0x0	Selection of the clock used as frequency reference of the PLL in Rx mode (also to ck_test and ck_dig outputs)
0x40040CB4	RF2_REG2D	(29:28) PA_CONF_SW_CN	(29:28) PA_CONF_SW_CN	0x0	Harmonic 2 notch tuning
		(27) PA_CONF_TX_SWITCHPA	(27) PA_CONF_TX_SWITCHPA	0x0	Switch PA
		(26) PA_CONF_TX_0DBM	(26) PA_CONF_TX_0DBM	0x1	Select between PPA and PA
		(25) PA_CONF_LIN_RAMP	(25) PA_CONF_LIN_RAMP	0x0	PA ramp-up linearization
		(24) PA_CONF_MIN_PA_PWR	(24) PA_CONF_MIN_PA_PWR	0x1	Set the minimum power during the PA ramp-up
		(23) CTRL_RX_SWITCH_LP	(23) CTRL_RX_SWITCH_LP	0x0	Switch the low-pass filter in the Rx chain
		(22) CTRL_RX_USE_PEAK_DETECTOR	(22) CTRL_RX_USE_PEAK_DETECTOR	0x1	Peak detector powering
		(21) CTRL_RX_START_MIX_ON_CAL	(21) CTRL_RX_START_MIX_ON_CAL	0x0	Mixer enabling

Address	Register Name	Register Write	Register Read	Default	Description
		(20:16) CTRL_RX_ CTRL_RX	(20:16) CTRL_RX_ CTRL_RX	0xF	Rx control
		(15) CTRL_ADC_ PHADC_THERM_OUT_EN	(15) CTRL_ADC_ PHADC_THERM_OUT_EN	0x1	Enable the buffers of phase ADC thermometric code (banked)
		(14:13) CTRL_ADC_ PHADC_DELLATCH	(14:13) CTRL_ADC_ PHADC_DELLATCH	0x1	Phase ADC delay latch trimming (banked)
		(12:8) CTRL_ADC_ CTRL_ADC	(12:8) CTRL_ADC_ CTRL_ADC	0x5	Phase ADC control (banked)
		(6:5) RSSI_TUN_2_ RSSI_TRI_CK_DIV	(6:5) RSSI_TUN_2_ RSSI_TRI_CK_DIV	0x0	Speed on the RSSI triangular dithering signal (cf reg RSSI_TUN)
		(4) RSSI_TUN_2_ RSSI_ONE_CK_RSSI_ PHADC	(4) RSSI_TUN_2_ RSSI_ONE_CK_RSSI_ PHADC	0x0	RSSI and phase ADC clocks sharing
		(3) RSSI_TUN_2_ RSSI_FULL	(3) RSSI_TUN_2_ RSSI_FULL	0x1	RSSI full scale
		(2) RSSI_TUN_2_ RSSI_1DB	(2) RSSI_TUN_2_ RSSI_1DB	0x0	LSB resolution
		(1:0) RSSI_TUN_2_ RSSI_PRE_ATT	(1:0) RSSI_TUN_2_ RSSI_PRE_ATT	0x3	Pre attenuation of the RSSI signal
0x40040CB8	RF2_REG2E	(31:24) XTAL_TRIM_ XTAL_TRIM_INIT	(31:24) XTAL_TRIM_ XTAL_TRIM_INIT	0x60	Initial trimming of the XTAL
		(23:16) XTAL_TRIM_ XTAL_TRIM	(23:16) XTAL_TRIM_ XTAL_TRIM	0x60	Trimming of the XTAL
		(12) ENABLES_ SEPARATE_PPA_CASC	(12) ENABLES_ SEPARATE_PPA_CASC	0x0	PA cascode bit
		(11:6) ENABLES_EN_ RXTX	(11:6) ENABLES_EN_ RXTX	0x0	Enable signals
		(5:0) ENABLES_EN_BB	(5:0) ENABLES_EN_	0x0	Enable signals for the BB

Address	Register Name	Register Write	Register Read	Default	Description
			BB		
0x40040CBC	RF2_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
		(19) XTAL_CTRL_LOW_CLK_READY_TH_EN	(19) XTAL_CTRL_LOW_CLK_READY_TH_EN	0x0	clk_ready threshold
		(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	Clock selection for the digital block
		(16) XTAL_CTRL_XO_EN_B_REG	(16) XTAL_CTRL_XO_EN_B_REG	0x1	XTAL oscillator enable
		(15:14) XTAL_CTRL_XTAL_CKDIV	(15:14) XTAL_CTRL_XTAL_CKDIV	0x0	XTAL trimming speed
		(13) XTAL_CTRL_CLK_OUT_EN_B	(13) XTAL_CTRL_CLK_OUT_EN_B	0x0	Output clock to go to main IP
		(12) XTAL_CTRL_REG_VALUE_SEL	(12) XTAL_CTRL_REG_VALUE_SEL	0x0	Control bits of xtal_reg
		(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

Address	Register Name	Register Write	Register Read	Default	Description
					core oscillator
		(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
		(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	Force output clocks on clk_pll, clk_dig and clk_out
		(4) XTAL_CTRL_CLK_DIG_EN_B	(4) XTAL_CTRL_CLK_DIG_EN_B	0x0	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	XTAL buffer disabling
		(2) XTAL_CTRL_HP_MODE	(2) XTAL_CTRL_HP_MODE	0x0	Bias current increase in the clock buffer
		(1) XTAL_CTRL_LP_MODE	(1) XTAL_CTRL_LP_MODE	0x0	Bias current decrease in the clock buffer
		(0) XTAL_CTRL_EXT_CLK_MODE	(0) XTAL_CTRL_EXT_CLK_MODE	0x0	Use XTAL pads as external clock input
0x40040CC0	RF2_SUBBAND	(31:24) SUBBAND_OFFSET_SB_OFFSET_RX	(31:24) SUBBAND_OFFSET_SB_OFFSET_RX	0xF1	Offset to add in frequency count in order to compensate the offset of the varicap
		(23:16) SUBBAND_OFFSET_SB_OFFSET	(23:16) SUBBAND_OFFSET_SB_OFFSET	0xD0	Offset to add in frequency count in order to compensate the offset of the varicap
		(15:12) SWCAP_LIM_SB_MAX_VAL	(15:12) SWCAP_LIM_SB_MAX_VAL	0xF	Maximum subband value in linear search subband (freq and comp)
		(11:8) SWCAP_LIM_	(11:8) SWCAP_LIM_	0x0	Minimum subband value in linear

Address	Register Name	Register Write	Register Read	Default	Description
		SB_MIN_VAL	SB_MIN_VAL		search subband (freq and comp)
		(7) SUBBAND_CONF_ SB_FLL_MODE	(7) SUBBAND_CONF_ SB_FLL_MODE	0x1	FLL mode for the subband selection
		(6) SUBBAND_CONF_ SB_INV_BAND	(6) SUBBAND_CONF_ SB_INV_BAND	0x0	Invert the meaning of sb_high and sb_ low
		(5:4) SUBBAND_CONF_ SB_FREQ_CNT	(5:4) SUBBAND_ CONF_SB_FREQ_CNT	0x0	The length to count in frequency mode
		(3:2) SUBBAND_CONF_ SB_WAIT_T	(3:2) SUBBAND_ CONF_SB_WAIT_T	0x0	Time to wait to the PLL to settle
		(1:0) SUBBAND_CONF_ SB_MODE	(1:0) SUBBAND_ CONF_SB_MODE	0x0	Sub-band algorithm mode
0x40040CC4	RF2_REG31	(31:30) RSSI_ DETECT_RSSI_DET_CR_ LEN	(31:30) RSSI_ DETECT_RSSI_DET_ CR_LEN	0x0	Number of samples to estimate the carrier offset (banked)
		(29:28) RSSI_ DETECT_RSSI_DET_ WAIT	(29:28) RSSI_ DETECT_RSSI_DET_ WAIT	0x0	Symbols to wait after the RSSI detection (banked)
		(27:26) RSSI_ DETECT_RSSI_DET_ DIFF_LL	(27:26) RSSI_ DETECT_RSSI_DET_ DIFF_LL	0x0	Set the distance between the actual value and the subtracted one (banked)
		(25) RSSI_DETECT_ EN_ABS_RSSI_DETECT	(25) RSSI_DETECT_ EN_ABS_RSSI_DETECT	0x0	Absolute RSSI detection (banked)
		(24) RSSI_DETECT_ EN_DIFF_RSSI_DETECT	(24) RSSI_DETECT_ EN_DIFF_RSSI_ DETECT	0x0	Differential RSSI detection (banked)
		(23) SUBBAND_CORR_ SUBBAND_CORR_EN	(23) SUBBAND_CORR_ SUBBAND_CORR_EN	0x0	Subband correction
		(22:20) SUBBAND_ CORR_SUBBAND_CORR_	(22:20) SUBBAND_ CORR_SUBBAND_CORR_	0x0	Subband correction in Rx

Address	Register Name	Register Write	Register Read	Default	Description
		RX	RX		
		(18:16) SUBBAND_CORR_SUBBAND_CORR_TX	(18:16) SUBBAND_CORR_SUBBAND_CORR_TX	0x0	Subband correction in Tx
		(11) TXRX_CONF_INV_CLK_PLL_TX	(11) TXRX_CONF_INV_CLK_PLL_TX	0x0	Invert PLL clock when the radio is in Tx mode
		(10) TXRX_CONF_INV_CLK_DIG_TX	(10) TXRX_CONF_INV_CLK_DIG_TX	0x0	Invert digital clock when the radio is in Tx mode
		(9:8) TXRX_CONF_SB_WAIT_T_TX	(9:8) TXRX_CONF_SB_WAIT_T_TX	0x0	Xor value to apply to sb_wait_t (register SUBBAND_CONF) when the radio is in Tx mode
		(7) PA_RAMPUP_FULL_PA_RAMPUP	(7) PA_RAMPUP_FULL_PA_RAMPUP	0x1	PA rampup configuration
		(6:4) PA_RAMPUP_DEL_PA_RAMPUP	(6:4) PA_RAMPUP_DEL_PA_RAMPUP	0x4	Time to wait to start the ramp-up after the PA enable is detected
		(3:2) PA_RAMPUP_TAU_PA_RAMPUP	(3:2) PA_RAMPUP_TAU_PA_RAMPUP	0x0	Time constant of the ramp-up/ramp-down
		(1) PA_RAMPUP_EN_PA_RAMPDOWN	(1) PA_RAMPUP_EN_PA_RAMPDOWN	0x1	PA ramp-down
		(0) PA_RAMPUP_EN_PA_RAMPUP	(0) PA_RAMPUP_EN_PA_RAMPUP	0x1	PA ramp-up linearization
0x40040CC8	RF2_DEMOD_CTRL	(31) SYNC_WORD_CORR_EN_SYNC_WORD_CORR	(31) SYNC_WORD_CORR_EN_SYNC_WORD_CORR	0x1	Sync word bias correction with RSSI detection (banked)
		(29:24) SYNC_WORD_CORR_SYNC_WORD_BIAS	(29:24) SYNC_WORD_CORR_SYNC_WORD_BIAS	0x8	Set the sync word bias (banked)
		(23:16) RSSI_DETECT_ABS_THR_	(23:16) RSSI_DETECT_ABS_THR_	0x0	Threshold used for absolute RSSI detection

Address	Register Name	Register Write	Register Read	Default	Description
		RSSI_DET_ABS_THR	RSSI_DET_ABS_THR		
		(15:8) RSSI_DETECT_DIFF_THR_RSSI_DET_DIFF_THR	(15:8) RSSI_DETECT_DIFF_THR_RSSI_DET_DIFF_THR	0x0	Threshold used for differential RSSI detection
		(7) DEMOD_CTRL_DL_SYNC_NO_DATA	(7) DEMOD_CTRL_DL_SYNC_NO_DATA	0x1	No data going through the demodulator, until the delay line detects the sync word (banked)
		(6) DEMOD_CTRL_EN_DELLINE_SYNC_DET	(6) DEMOD_CTRL_EN_DELLINE_SYNC_DET	0x1	Sync word detection in the delay line (banked)
		(5) DEMOD_CTRL_RSSI_DET_FILT	(5) DEMOD_CTRL_RSSI_DET_FILT	0x0	Additional filtering on the RSSI value (banked)
		(4) DEMOD_CTRL_EN_FAST_CLK_RECOV	(4) DEMOD_CTRL_EN_FAST_CLK_RECOV	0x0	Clock recovery during the resto of the preamble (banked)
		(3) DEMOD_CTRL_EN_MIN_MAX_MF	(3) DEMOD_CTRL_EN_MIN_MAX_MF	0x0	Min max algo after the matched filter (banked)
		(2) DEMOD_CTRL_EN_PRE_SYNC	(2) DEMOD_CTRL_EN_PRE_SYNC	0x0	Sync detection on the non-delayed path (banked)
		(1) DEMOD_CTRL_BLOCK_RSSI_DET	(1) DEMOD_CTRL_BLOCK_RSSI_DET	0x0	RSSI detection during the slow-down period (banked)
		(0) DEMOD_CTRL_EARLY_FINE_RECOV	(0) DEMOD_CTRL_EARLY_FINE_RECOV	0x0	Early fine recovery after the packet detection or pre-sync (banked)
0x40040CCC	RF2_REG33	(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
		(23:16) SPARES_SPARES	(23:16) SPARES_SPARES	0x0	Spare bits
		(14) PADS_PE_DS_GPIO_DS	(14) PADS_PE_DS_GPIO_DS	0x0	Increased drive strength of the digital pads
		(13) PADS_PE_DS_	(13) PADS_PE_DS_	0x0	Pull-up of the GPIO pads

Address	Register Name	Register Write	Register Read	Default	Description
		GPIO_PE	GPIO_PE		
		(12) PADS_PE_DS_NRESET_PE	(12) PADS_PE_DS_NRESET_PE	0x0	Pull-up of the NRESET pads
		(11) PADS_PE_DS_SPI_MISO_PE	(11) PADS_PE_DS_SPI_MISO_PE	0x0	Pull-up of the SPI MISO pads
		(10) PADS_PE_DS_SPI_MOSI_PE	(10) PADS_PE_DS_SPI_MOSI_PE	0x0	Pull-up of the SPI MOSI pads
		(9) PADS_PE_DS_SPI_SCLK_PE	(9) PADS_PE_DS_SPI_SCLK_PE	0x0	Pull-up of the SPI CLK pads
		(8) PADS_PE_DS_SPI_CS_N_PE	(8) PADS_PE_DS_SPI_CS_N_PE	0x0	Pull-up of the SPI CSN pads
		(7:6) SUBBAND_FLL_SB_FLL_DITHER	(7:6) SUBBAND_FLL_SB_FLL_DITHER	0x0	Select the dithering
		(5:4) SUBBAND_FLL_SB_FLL_CIC_TAU	(5:4) SUBBAND_FLL_SB_FLL_CIC_TAU	0x3	Set the CIC decimator factor
		(3) SUBBAND_FLL_SB_FLL_PH_4_N8	(3) SUBBAND_FLL_SB_FLL_PH_4_N8	0x0	Phases in the frequency detector
		(2:0) SUBBAND_FLL_SB_FLL_WAIT	(2:0) SUBBAND_FLL_SB_FLL_WAIT	0x3	Set the number of CIC samples before stopping the FLL
0x40040CD0	RF2_REG34	(29:24) CLK_RECOVERY_CLK_RECOV_CORR	(29:24) CLK_RECOVERY_CLK_RECOV_CORR	0x4	Number of samples that covers the clock recovery correlator
		(23:16) CLK_RECOVERY_CLK_AB_LIMIT	(23:16) CLK_RECOVERY_CLK_AB_LIMIT	0x80	Time constant for switch the clock phase if chosen wrong in clk recovery algorithm
		(15) TX_PRE_DIST_EN_PRE_DIST	(15) TX_PRE_DIST_EN_PRE_DIST	0x1	Tx pre-distortion filter (banked)
		(13:8) TX_PRE_DIST_	(13:8) TX_PRE_	0x2E	Coefficient b0 of the Tx pre-distortion

Address	Register Name	Register Write	Register Read	Default	Description
		PRE_DIST_B0	DIST_PRE_DIST_B0		filter (banked)
		(5:0) TX_PRE_DIST_ PRE_DIST_A0	(5:0) TX_PRE_DIST_ PRE_DIST_A0	0x2F	Coefficient a0 of the Tx pre-distortion filter (banked)
0x40040CD4	RF2_BLE_LR	(30:24) BLR_SYNC_ THRESHOLD_BLE_SYNC_ THR	(30:24) BLR_SYNC_ THRESHOLD_BLE_ SYNC_THR	0x38	Threshold for the BLR sync word detector
		(19:16) BLR_ PREAMBLE_BLE_PRE_ THR	(19:16) BLR_ PREAMBLE_BLE_PRE_ THR	0x1	Threshold for the BLR preamble detector
		(15) BLE_LONG_ RANGE_BLR_PUT_RI_ FIFO	(15) BLE_LONG_ RANGE_BLR_PUT_RI_ FIFO	0x1	During the reception the RI (rate indicator) is put into the Rx FIFO (banked)
		(14) BLE_LONG_ RANGE_BLR500_NO_ ROUGH	(14) BLE_LONG_ RANGE_BLR500_NO_ ROUGH	0x1	Rough recovery is stopped during the 500kbps payloads of BLR packets (banked)
		(13) BLE_LONG_ RANGE_BLR_LIN_ FILTER	(13) BLE_LONG_ RANGE_BLR_LIN_ FILTER	0x1	Matched filter (banked)
		(12) BLE_LONG_ RANGE_EN_BLR_FLUSH	(12) BLE_LONG_ RANGE_EN_BLR_FLUSH	0x1	Viterbi path 0 flushing at the end of the packet (banked)
		(11) BLE_LONG_ RANGE_BLR_USE_EXT_ LEN	(11) BLE_LONG_ RANGE_BLR_USE_EXT_ LEN	0x0	BLR_PKT_LEN for flushing out the Viterbi (banked)
		(10) BLE_LONG_ RANGE_DISABLE_BLR_ TX	(10) BLE_LONG_ RANGE_DISABLE_BLR_ TX	0x0	Long Range feature in Tx mode (banked)
		(9) BLE_LONG_RANGE_ BLR_500_N125	(9) BLE_LONG_ RANGE_BLR_500_N125	0x0	Data rate selection (banked)
		(8) BLE_LONG_RANGE_	(8) BLE_LONG_	0x0	BLE long range mode (banked)

Address	Register Name	Register Write	Register Read	Default	Description
		EN_BLR	RANGE_EN_BLR		
		(4) HW_TRIGGER_HW_TRIG_GPIO	(4) HW_TRIGGER_HW_TRIG_GPIO	0x0	HW trigger is mapped on the GPIO instead of the Tx_on signal 0x0
		(3) HW_TRIGGER_HW_TRIG_SUBBAND	(3) HW_TRIGGER_HW_TRIG_SUBBAND	0x0	Activate the sub-band selection during the Tx activation
		(2) HW_TRIGGER_HW_TRIG_TX_NRX	(2) HW_TRIGGER_HW_TRIG_TX_NRX	0x0	Activate the Tx mode
		(1) HW_TRIGGER_HW_TRIG_LOW	(1) HW_TRIGGER_HW_TRIG_LOW	0x0	Set the trigger polarity
		(0) HW_TRIGGER_HW_TRIG_ACTIVE	(0) HW_TRIGGER_HW_TRIG_ACTIVE	0x0	Enable HW trigger
0x40040CD8	RF2_REG36	(30:28) IQ_SPARES_EN_BIAS_SPARE	(30:28) IQ_SPARES_EN_BIAS_SPARE	0x0	Enable for IQ spares
		(27:24) IQ_SPARES_IQ_SPARE_2	(27:24) IQ_SPARES_IQ_SPARE_2	0x0	Spare bias 2
		(23:20) IQ_SPARES_IQ_SPARE_1	(23:20) IQ_SPARES_IQ_SPARE_1	0x0	Spare bias 1
		(19:16) IQ_SPARES_IQ_SPARE_0	(19:16) IQ_SPARES_IQ_SPARE_0	0x0	Spare bias 0
		(8) MISC_ISO_VDDA	(8) MISC_ISO_VDDA	0x0	Isolate VDDA signals
		(5) BLR_DEMAPPER_BLR_SEND_DECODED_RI	(5) BLR_DEMAPPER_BLR_SEND_DECODED_RI	0x0	Fully decode the rate indicator
		(4) BLR_DEMAPPER_BLR_USE_EXT_VIT_GFSK	(4) BLR_DEMAPPER_BLR_USE_EXT_VIT_GFSK	0x1	500kbps BLR uses the Viterbi GFSK decision
		(3:2) BLR_DEMAPPER_BLR_500_DPHASE	(3:2) BLR_DEMAPPER_BLR_500_	0x3	Set the distance between samples for the phase to frequency conversion in

Address	Register Name	Register Write	Register Read	Default	Description
			DPHASE		S2 mode
		(1) BLR_DEMAPPER_BLR_500_LOW_GAIN	(1) BLR_DEMAPPER_BLR_500_LOW_GAIN	0x0	Set the low gain in S2 mode
		(0) BLR_DEMAPPER_BLR_125_LOW_GAIN	(0) BLR_DEMAPPER_BLR_125_LOW_GAIN	0x0	Set the low gain in S8 mode
0x40040CDC	RF2_PROT_TIMER	(31) PROT_TIMER_CONF_EN_PROT_TIMER	(31) PROT_TIMER_CONF_EN_PROT_TIMER	0x0	Enable the protocol timer
		(29:27) PROT_TIMER_CONF_PT_T_STP_1	(29:27) PROT_TIMER_CONF_PT_T_STP_1	0x0	Configure the time stamp 1
		(26:24) PROT_TIMER_CONF_PT_T_STP_0	(26:24) PROT_TIMER_CONF_PT_T_STP_0	0x0	Configure the time stamp 0
		(22) STAGING_PS_NZ_START_BIT	(22) STAGING_PS_NZ_START_BIT	0x0	Select the frequency offset
		(21) STAGING_PS_NZ_START	(21) STAGING_PS_NZ_START	0x0	Start the pulse shaper with a +/- 250 kHz frequency offset
		(20) STAGING_DEL_PA_RAMPDW	(20) STAGING_DEL_PA_RAMPDW	0x0	Delay the PA ramp-down by 4.5 us
		(19) STAGING_PEAK_DET_TH_SHIFT	(19) STAGING_PEAK_DET_TH_SHIFT	0x0	Peak detector threshold shift
		(18:17) STAGING_AGC_DERIV_LVL	(18:17) STAGING_AGC_DERIV_LVL	0x2	Select the AGC derivative level
		(16) STAGING_AGC_USE_DERIV	(16) STAGING_AGC_USE_DERIV	0x0	AGC algorithm uses the derivative information to accelerate the AGC settling
		(15:8) BLE_DTM_BLE_DTM_LEN	(15:8) BLE_DTM_BLE_DTM_LEN	0x25	Set the BLE DTM packet length

Address	Register Name	Register Write	Register Read	Default	Description
		(7) BLE_DTM_EN_BLE_DTM	(7) BLE_DTM_EN_BLE_DTM	0x0	Enable the BLE DTM automatic packets
		(3:0) BLE_DTM_BLE_DTM_PKT_TYPE	(3:0) BLE_DTM_BLE_DTM_PKT_TYPE	0x0	Set the BLE DTM packet type (see Bluetooth specification)
0x40040CE0	RF2_CTE_OPTS	(29) CTE_OPTS_RECT_PS_CTE	(29) CTE_OPTS_RECT_PS_CTE	0x0	Use rectangular pulse shape during the CTE
		(28) CTE_OPTS_USE_CTE_WO_CP	(28) CTE_OPTS_USE_CTE_WO_CP	0x0	Enable the CTE without reading or inserting the CP
		(27) CTE_OPTS_CTE_AMPL	(27) CTE_OPTS_CTE_AMPL	0x0	Enable the usage of the RSSI values to adapt the amplitude of the IQ signal based to the RSSI value
		(26) CTE_OPTS_DF_AOA_SLOT_TIME	(26) CTE_OPTS_DF_AOA_SLOT_TIME	0x0	Indicate the switching/sampling slot period for AoA
		(25) CTE_OPTS_CP_INSERT	(25) CTE_OPTS_CP_INSERT	0x0	Force the CP bit in the packet header to 1
		(24) CTE_OPTS_EN_READ_CP	(24) CTE_OPTS_EN_READ_CP	0x0	CP bit is read in the packet header (BLE standard)
		(23:16) CTE_OPTS_CTE_INFO	(23:16) CTE_OPTS_CTE_INFO	0x0	Set the CTEInfo field in the packet header while cp_insert is set to 1
		(14:10) ASK_MOD_ASK_MAX	(14:10) ASK_MOD_ASK_MAX	0xC	Set the maximum value for the ASK modulation
		(9:5) ASK_MOD_ASK_MIN	(9:5) ASK_MOD_ASK_MIN	0x0	Set the minimum value for the ASK modulation
		(4:1) ASK_MOD_ASK_CNT	(4:1) ASK_MOD_ASK_CNT	0x7	Set the how long to count for the ASK modulation
		(0) ASK_MOD_EN_RSSI_ASK	(0) ASK_MOD_EN_RSSI_ASK	0x0	PA will perform an ASK modulation
0x40040CE4	RF2_PT_DELTA_0	(31:30) PT_DELTA_	(31:30) PT_DELTA_	0x0	Multiplier for the delta t0

Address	Register Name	Register Write	Register Read	Default	Description
		TS_0_PT_DELTA_T0_MULT	TS_0_PT_DELTA_T0_MULT		
		(19:0) PT_DELTA_TS_0_PT_DELTA_T0	(19:0) PT_DELTA_TS_0_PT_DELTA_T0	0x0	Delta t0 for the protocol timer
0x40040CE8	RF2_PT_DELTA_1	(31:30) PT_DELTA_TS_1_PT_DELTA_T1_MULT	(31:30) PT_DELTA_TS_1_PT_DELTA_T1_MULT	0x0	Multiplier for the delta t1
		(19:0) PT_DELTA_TS_1_PT_DELTA_T1	(19:0) PT_DELTA_TS_1_PT_DELTA_T1	0x0	Delta t1 for the protocol timer
0x40040CEC	RF2_CTE_IF	(25:16) CTE_CTRL_DELAY_TX_DF_DELAY_TX	(25:16) CTE_CTRL_DELAY_TX_DF_DELAY_TX	0x0	Delay (in 62.5ns) form the serializer up to the antenna in direction finding (banked)
		(15) ANTENNA_CONF_DF_IND_PATTERN	(15) ANTENNA_CONF_DF_IND_PATTERN	0x0	Separate the antenna switching pattern from the reference one
		(14) ANTENNA_CONF_DF_IND_ANTENNA	(14) ANTENNA_CONF_DF_IND_ANTENNA	0x0	Make the antenna for DF independent from the rest of the packet
		(13:8) ANTENNA_CONF_ANT_LUT_M	(13:8) ANTENNA_CONF_ANT_LUT_M	0x0	Number of states used (-1) in the antenna LUT
		(5) CTE_AUTO_PULL_EXT_IQ_SMP_TYPE	(5) CTE_AUTO_PULL_EXT_IQ_SMP_TYPE	0x0	Select the external IQ sample signal qualifier type
		(4) CTE_AUTO_PULL_IQ_MSB	(4) CTE_AUTO_PULL_IQ_MSB	0x0	Select which signal is sent over the MSB in case of a 16bits buffers
		(3:2) CTE_AUTO_PULL_IQ_DATA_BUS_SIZE	(3:2) CTE_AUTO_PULL_IQ_DATA_BUS_SIZE	0x0	Select the bus data size of IQ signals
		(1) CTE_AUTO_PULL_CTE_QUAL	(1) CTE_AUTO_PULL_CTE_QUAL	0x0	Select the CTE data qualifier
		(0) CTE_AUTO_PULL_EN_CTE_AUTO_PULL	(0) CTE_AUTO_PULL_EN_CTE_AUTO_PULL	0x0	Enable the automatic push of CTE data to an external IP

Address	Register Name	Register Write	Register Read	Default	Description
0x40040CF0	RF2_CTE_CTRL	(25:16) CTE_CTRL_ DELAY_RX_DF_DELAY_ SWITCH_RX	(25:16) CTE_CTRL_ DELAY_RX_DF_DELAY_ SWITCH_RX	0x0	Delay (in 62.5ns) from the antenna up to the deserializer in direction finding (banked)
		(9:0) CTE_CTRL_ DELAY_RX_DF_DELAY_ SAMPLE_RX	(9:0) CTE_CTRL_ DELAY_RX_DF_DELAY_ SAMPLE_RX	0x0	Delay (in 62.5ns) from the matched filter up to the deserializer in direction finding (banked)
0x40040CF4	RF2_AGC_ADVANCED	(26:16) AGC_ SWITCHES_AGC_ SHORTS_LUT	(26:16) AGC_ SWITCHES_AGC_ SHORTS_LUT	0x0	Array of values that indicates if the highpass shorts must be set for the AGC state passage from n -> n+1
		(8) DEBUG_FAKE_IQ_ SAMPLES	(8) DEBUG_FAKE_IQ_ SAMPLES	0x0	Generate fake IQ samples
		(7:4) AGC_ADVANCED_ AGC_TAU_SHORTS	(7:4) AGC_ ADVANCED_AGC_TAU_ SHORTS	0x0	Time constant that indicates the time that shorts must be on
		(3) AGC_ADVANCED_ AGC_EN_SHORT_PHADC	(3) AGC_ADVANCED_ AGC_EN_SHORT_PHADC	0x0	Enable the short on the phase ADC highpass filter
		(2) AGC_ADVANCED_ AGC_EN_SHORT_IFA	(2) AGC_ADVANCED_ AGC_EN_SHORT_IFA	0x0	Enable the short on the IFA highpass filter
		(1) AGC_ADVANCED_ AGC_USE_SHORTS	(1) AGC_ADVANCED_ AGC_USE_SHORTS	0x0	Enable the usage of the shorts located in the BB path
		(0) AGC_ADVANCED_ AGC_FULL_SPEED	(0) AGC_ADVANCED_ AGC_FULL_SPEED	0x0	Enable the maximum speed in AGC
0x40040CF8	RF2_DATA_STREAMING	(9) DATA_STREAMING_ DMA_PHASE_TYPE	(9) DATA_ STREAMING_DMA_ PHASE_TYPE	0x0	Use the phase after the rescaler instead of the raw phase from phase ADC (banked)
		(8) DATA_STREAMING_ DMA_EN_BUS	(8) DATA_ STREAMING_DMA_EN_ BUS	0x0	Enable the DMA bus on the IP interface (banked)
		(6) DATA_STREAMING_	(6) DATA_	0x1	Restart sampling after the CTE period

Address	Register Name	Register Write	Register Read	Default	Description
		PERIODIC_SAMPLE_AFTER_CTE	STREAMING_PERIODIC_SAMPLE_AFTER_CTE		(banked)
		(5) DATA_STREAMING_PERIODIC_SAMPLE_AT_SYNC	(5) DATA_STREAMING_PERIODIC_SAMPLE_AT_SYNC	0x0	Start sampling at the sync detection signal from the delay line (banked)
		(4) DATA_STREAMING_PERIODIC_SAMPLE_OSR_CLK	(4) DATA_STREAMING_PERIODIC_SAMPLE_OSR_CLK	0x0	Oversample (8x) the reference clock of the periodic sample (banked)
		(3:1) DATA_STREAMING_PERIODIC_SAMPLE_OSR	(3:1) DATA_STREAMING_PERIODIC_SAMPLE_OSR	0x3	Division factor (-1) of for the sampling period of the periodic IQ sampling (banked)
		(0) DATA_STREAMING_PERIODIC_SAMPLE_EN_IQ	(0) DATA_STREAMING_PERIODIC_SAMPLE_EN_IQ	0x0	Sample periodically I and Q channels after the matched filter and put into the IQ FIFO (banked)
0x40040CFC	RF2_REVISION	–	(31:24) CHIP_ID	0x30	Remapped register of CHIP_ID
0x40040D00	RF2_FSM_CTRL	–	(31:30) RXFIFO_STATUS_RX_BIST_ERRORS	0x0	Rx FIFO BIST result
		(31:25) RXFIFO_STATUS_RX_BIST	–	N/A	Start the bist test on the Rx FIFO (code 0x5d)
		–	(29) RXFIFO_STATUS_RX_NEAR_UNDERFLOW	0x0	Rx FIFO near underflow
		–	(28) RXFIFO_STATUS_RX_NEAR_	0x0	Rx FIFO near overflow

Address	Register Name	Register Write	Register Read	Default	Description
			OVERFLOW		
		–	(27) RXFIFO_ STATUS_RX_ UNDERFLOW	0x0	Rx FIFO underflow
		–	(26) RXFIFO_ STATUS_RX_OVERFLOW	0x0	Rx FIFO overflow
		–	(25) RXFIFO_ STATUS_RX_FULL	0x0	Rx FIFO full
		–	(24) RXFIFO_ STATUS_RX_EMPTY	0x0	Rx FIFO empty
		(24) RXFIFO_STATUS_ RX_FLUSH	–	N/A	Rx FIFO flush
		–	(23:22) TXFIFO_ STATUS_TX_BIST_ ERRORS	0x0	Tx FIFO BIST result
		(23:17) TXFIFO_ STATUS_TX_BIST	–	N/A	Start the bist test on the Tx FIFO (code 0x5d)
		–	(21) TXFIFO_ STATUS_TX_NEAR_ UNDERFLOW	0x0	Tx FIFO near underflow
		–	(20) TXFIFO_ STATUS_TX_NEAR_ OVERFLOW	0x0	Tx FIFO near overflow
		–	(19) TXFIFO_ STATUS_TX_ UNDERFLOW	0x0	Tx FIFO underflow
		–	(18) TXFIFO_ STATUS_TX_OVERFLOW	0x0	Tx FIFO overflow
		–	(17) TXFIFO_	0x0	Tx FIFO full

Address	Register Name	Register Write	Register Read	Default	Description
			STATUS_TX_FULL		
		–	(16) TXFIFO_ STATUS_TX_EMPTY	0x0	Tx FIFO empty
		(16) TXFIFO_STATUS_ TX_FLUSH	–	N/A	Tx FIFO flush
		–	(10) FSM_STATUS_ TX_NRX	0x0	Select Rx or Tx mode
		–	(9:8) FSM_STATUS_ STATUS	0x0	Status of the FSM
		(3) FSM_MODE_RESET	–	N/A	FSM reset
		–	(2) FSM_MODE_RX_ MODE	0x0	Rx status
		(2) FSM_MODE_TX_NRX	–	N/A	Set the radio in Tx or Rx mode
		–	(1) FSM_MODE_TX_ MODE	0x0	Tx status
		(1:0) FSM_MODE_MODE	–	N/A	Set the FSM mode
		–	(0) FSM_MODE_N_ IDLE	0x0	FSM status
0x40040D04	RF2_IQFIFO_STATUS	–	(24:16) TXFIFO_ COUNT_TX_COUNT	0x0	Number of bytes in the Tx FIFO
		–	(15:8) IQFIFO_ COUNT_IQ_COUNT	0x0	Number of bytes in the IQ FIFO
		–	(7:6) IQFIFO_ STATUS_IQ_BIST_ ERRORS	0x0	IQ FIFO BIST result
		(7:1) IQFIFO_ STATUS_IQ_BIST	–	N/A	Start the BIST test on the IQ FIFO (code 0x5d)

Address	Register Name	Register Write	Register Read	Default	Description
		–	(5) IQFIFO_STATUS_IQ_NEAR_UNDERFLOW	0x0	IQ FIFO near underflow
		–	(4) IQFIFO_STATUS_IQ_NEAR_OVERFLOW	0x0	IQ FIFO near overflow
		–	(3) IQFIFO_STATUS_IQ_UNDERFLOW	0x0	IQ FIFO underflow
		–	(2) IQFIFO_STATUS_IQ_OVERFLOW	0x0	IQ FIFO overflow
		–	(1) IQFIFO_STATUS_IQ_FULL	0x0	IQ FIFO full
		–	(0) IQFIFO_STATUS_IQ_EMPTY	0x0	IQ FIFO empty
		(0) IQFIFO_STATUS_FLUSH	–	N/A	IQ FIFO flush
0x40040D08	RF2_TXFIFO	(7:0) TXFIFO_TX_DATA	–	N/A	Data to be sent
0x40040D0C	RF2_RXFIFO	–	(7:0) RXFIFO_RX_DATA	0x0	Received data
0x40040D10	RF2_IQFIFO	–	(7:0) IQFIFO_IQ_DATA	0x0	IQ data for AoA or AoD
0x40040D14	RF2_REG45	–	(25:16) RSSI_AVG_RSSI_AVG	0x0	Filtered RSSI value
		–	(8:0) RXFIFO_COUNT_RX_COUNT	0x0	Number of bytes in the Rx FIFO
0x40040D18	RF2_DESER_STATUS	–	(7) DESER_STATUS_SIGNAL_RECEIVING	0x0	Deserializer enabling
		–	(6) DESER_STATUS_SYNC_DETECTED	0x0	Sync word detection

Address	Register Name	Register Write	Register Read	Default	Description
		–	(5) DESER_STATUS_WAIT_SYNC	0x0	Deserializer waiting for the sync word
		–	(4) DESER_STATUS_IS_ADDRESS_BR	0x0	Received address
		–	(3) DESER_STATUS_PKT_LEN_ERR	0x0	Packet length
		–	(2) DESER_STATUS_ADDRESS_ERR	0x0	Address error
		–	(1) DESER_STATUS_CRC_ERR	0x0	CRC error
		–	(0) DESER_STATUS_DESER_FINISH	0x0	Deserializer status
0x40040D1C	RF2_BLE_AEC_CCM	–	(2) BLE_AES_CCM_BLE_AES_MIC_OK	0x0	AES CCM MIC error
		–	(1) BLE_AES_CCM_BLE_AES_DONE_RX	0x0	AES CCM packet decoding
		–	(0) BLE_AES_CCM_BLE_AES_DONE_TX	0x0	AES CCM packet encoding
0x40040D20	RF2_IRQ_STATUS	–	(5) IRQ_STATUS_FLAG_RXFIFO	0x0	IRQ RXFIFO status
		–	(4) IRQ_STATUS_FLAG_TXFIFO	0x0	IRQ TXFIFO status
		–	(3) IRQ_STATUS_FLAG_SYNC	0x0	IRQ SYNC status
		–	(2) IRQ_STATUS_FLAG_RECEIVED	0x0	IRQ RECEIVED status
		–	(1) IRQ_STATUS_FLAG_RXSTOP	0x0	IRQ RXSTOP status

Address	Register Name	Register Write	Register Read	Default	Description
		–	(0) IRQ_STATUS_FLAG_TX	0x0	IRQ Tx status
0x40040D24	RF2_RSSI_MIN_MAX	–	(25:16) RSSI_MAX_RSSI_MAX	0x0	Maximum RSSI value over a filtering period
		–	(9:0) RSSI_MIN_RSSI_MIN	0x0	Minimum RSSI value over a filtering period
0x40040D28	RF2_REG4A	–	(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) DR_ERR_IND_DR_ERR_IND	0x0	Data-rate error indicator
		–	(9:0) RSSI_PKT_RSSI_PKT	0x0	Filtered RSSI value sampled during the packet reception
0x40040D2C	RF2_FEI	–	(31:16) FEI_PKT_FEI_PKT	0x0	Frequency error indicator sampled during the packet reception
		–	(15:0) FEI_FEI_OUT	0x0	Frequency error indicator
0x40040D30	RF2_REG4C	–	(31:24) LINK_QUAL_PKT_LINK_QUALITY_PKT	0x0	Link quality indicator sampled during the packet reception
		–	(23:16) LINK_QUAL_LINK_QUALITY	0x0	Instantaneous link quality indicator
		–	(15:0) FEI_AFC_FEI_AFC	0x0	Frequency error indicator sampled during the AFC
0x40040D34	RF2_ANALOG_INFO	(25:24) BLR_READOUT_BLR_RATE	–	N/A	Bluetooth LE long range rate indicator
		–	(22:20) PEAK_DET_	0x0	Distance from the subband center

Address	Register Name	Register Write	Register Read	Default	Description
			VAL_PEAK_DET_FILT		(only available with the FLL method)
		–	(18:16) PEAK_DET_ VAL_PEAK_DET_RAW	0x0	Distance from the subband center (only available with the FLL method)
		–	(15) ANALOG_INFO_ POR_VDDA	0x0	VDDA LDO disable status
		–	(14) ANALOG_INFO_ PLL_UNLOCK	0x0	PLL unlock status
		–	(13) ANALOG_INFO_ XTAL_FINISH	0x0	XTAL algorithm status
		–	(12) ANALOG_INFO_ DLL_LOCKED	0x0	DLL lock status
		–	(11) ANALOG_INFO_ CLK_DIG_READY	0x0	Ready signal of the digital clock
		–	(10) ANALOG_INFO_ CLK_PLL_READY	0x0	PLL clock status
		–	(9:8) ANALOG_INFO_ SUBBAND	0x0	Status of the subband comparator Hi
		–	(7:0) SUBBAND_ERR_ SB_FLL_ERR	0x0	Distance from the subband center (only available with the FLL method)
0x40040D38	RF2_SAMPLE_RSSI	(0) SAMPLE_RSSI	–	N/A	Sample the thermometric RSSI
0x40040D3C	RF2_RSSI_THERM	–	(29:0) RSSI_THERM	0x0	Thermometric value of the RSSI
0x40040D80	RF2_LUT_ANTENNA_ARRAY_ 1	(31:28) LUT_ ANTENNA_ARRAY_1_ ANTENNA_7	(31:28) LUT_ ANTENNA_ARRAY_1_ ANTENNA_7	0x0	Antenna 7 specification
		(27:24) LUT_ ANTENNA_ARRAY_1_ ANTENNA_6	(27:24) LUT_ ANTENNA_ARRAY_1_ ANTENNA_6	0x0	Antenna 6 specification
		(23:20) LUT_	(23:20) LUT_	0x0	Antenna 5 specification

Address	Register Name	Register Write	Register Read	Default	Description
		ANTENNA_ARRAY_1_ ANTENNA_5	ANTENNA_ARRAY_1_ ANTENNA_5		
		(19:16) LUT_ ANTENNA_ARRAY_1_ ANTENNA_4	(19:16) LUT_ ANTENNA_ARRAY_1_ ANTENNA_4	0x0	Antenna 4 specification
		(15:12) LUT_ ANTENNA_ARRAY_1_ ANTENNA_3	(15:12) LUT_ ANTENNA_ARRAY_1_ ANTENNA_3	0x3	Antenna 3 specification
		(11:8) LUT_ANTENNA_ ARRAY_1_ANTENNA_2	(11:8) LUT_ ANTENNA_ARRAY_1_ ANTENNA_2	0x2	Antenna 2 specification
		(7:4) LUT_ANTENNA_ ARRAY_1_ANTENNA_1	(7:4) LUT_ANTENNA_ ARRAY_1_ANTENNA_1	0x1	Antenna 1 specification
		(3:0) LUT_ANTENNA_ ARRAY_1_ANTENNA_0	(3:0) LUT_ANTENNA_ ARRAY_1_ANTENNA_0	0x0	Antenna 0 specification
0x40040D84	RF2_LUT_ANTENNA_ARRAY_ 2	(31:28) LUT_ ANTENNA_ARRAY_2_ ANTENNA_15	(31:28) LUT_ ANTENNA_ARRAY_2_ ANTENNA_15	0x0	Antenna 15 specification
		(27:24) LUT_ ANTENNA_ARRAY_2_ ANTENNA_14	(27:24) LUT_ ANTENNA_ARRAY_2_ ANTENNA_14	0x0	Antenna 14 specification
		(23:20) LUT_ ANTENNA_ARRAY_2_ ANTENNA_13	(23:20) LUT_ ANTENNA_ARRAY_2_ ANTENNA_13	0x0	Antenna 13 specification
		(19:16) LUT_ ANTENNA_ARRAY_2_ ANTENNA_12	(19:16) LUT_ ANTENNA_ARRAY_2_ ANTENNA_12	0x0	Antenna 12 specification
		(15:12) LUT_ ANTENNA_ARRAY_2_ ANTENNA_11	(15:12) LUT_ ANTENNA_ARRAY_2_ ANTENNA_11	0x0	Antenna 11 specification

Address	Register Name	Register Write	Register Read	Default	Description
		(11:8) LUT_ANTENNA_ARRAY_2_ANTENNA_10	(11:8) LUT_ANTENNA_ARRAY_2_ANTENNA_10	0x0	Antenna 10 specification
		(7:4) LUT_ANTENNA_ARRAY_2_ANTENNA_9	(7:4) LUT_ANTENNA_ARRAY_2_ANTENNA_9	0x0	Antenna 9 specification
		(3:0) LUT_ANTENNA_ARRAY_2_ANTENNA_8	(3:0) LUT_ANTENNA_ARRAY_2_ANTENNA_8	0x0	Antenna 8 specification
0x40040D88	RF2_LUT_ANTENNA_ARRAY_3	(31:28) LUT_ANTENNA_ARRAY_3_ANTENNA_23	(31:28) LUT_ANTENNA_ARRAY_3_ANTENNA_23	0x0	Antenna 23 specification
		(27:24) LUT_ANTENNA_ARRAY_3_ANTENNA_22	(27:24) LUT_ANTENNA_ARRAY_3_ANTENNA_22	0x0	Antenna 22 specification
		(23:20) LUT_ANTENNA_ARRAY_3_ANTENNA_21	(23:20) LUT_ANTENNA_ARRAY_3_ANTENNA_21	0x0	Antenna 21 specification
		(19:16) LUT_ANTENNA_ARRAY_3_ANTENNA_20	(19:16) LUT_ANTENNA_ARRAY_3_ANTENNA_20	0x0	Antenna 20 specification
		(15:12) LUT_ANTENNA_ARRAY_3_ANTENNA_19	(15:12) LUT_ANTENNA_ARRAY_3_ANTENNA_19	0x0	Antenna 19 specification
		(11:8) LUT_ANTENNA_ARRAY_3_ANTENNA_18	(11:8) LUT_ANTENNA_ARRAY_3_ANTENNA_18	0x0	Antenna 18 specification
		(7:4) LUT_ANTENNA_ARRAY_3_ANTENNA_17	(7:4) LUT_ANTENNA_ARRAY_3_ANTENNA_17	0x0	Antenna 17 specification
		(3:0) LUT_ANTENNA_ARRAY_3_ANTENNA_16	(3:0) LUT_ANTENNA_ARRAY_3_ANTENNA_16	0x0	Antenna 16 specification

Address	Register Name	Register Write	Register Read	Default	Description
0x40040D8C	RF2_LUT_ANTENNA_ARRAY_4	(31:28) LUT_ANTENNA_ARRAY_4_ANTENNA_31	(31:28) LUT_ANTENNA_ARRAY_4_ANTENNA_31	0x0	Antenna 31 specification
		(27:24) LUT_ANTENNA_ARRAY_4_ANTENNA_30	(27:24) LUT_ANTENNA_ARRAY_4_ANTENNA_30	0x0	Antenna 30 specification
		(23:20) LUT_ANTENNA_ARRAY_4_ANTENNA_29	(23:20) LUT_ANTENNA_ARRAY_4_ANTENNA_29	0x0	Antenna 29 specification
		(19:16) LUT_ANTENNA_ARRAY_4_ANTENNA_28	(19:16) LUT_ANTENNA_ARRAY_4_ANTENNA_28	0x0	Antenna 28 specification
		(15:12) LUT_ANTENNA_ARRAY_4_ANTENNA_27	(15:12) LUT_ANTENNA_ARRAY_4_ANTENNA_27	0x0	Antenna 27 specification
		(11:8) LUT_ANTENNA_ARRAY_4_ANTENNA_26	(11:8) LUT_ANTENNA_ARRAY_4_ANTENNA_26	0x0	Antenna 26 specification
		(7:4) LUT_ANTENNA_ARRAY_4_ANTENNA_25	(7:4) LUT_ANTENNA_ARRAY_4_ANTENNA_25	0x0	Antenna 25 specification
		(3:0) LUT_ANTENNA_ARRAY_4_ANTENNA_24	(3:0) LUT_ANTENNA_ARRAY_4_ANTENNA_24	0x0	Antenna 24 specification
0x40040DC0	RF2_REG50	–	(27) FEATURES_HAS_BLE_AES	0x0	Bluetooth AES block availability
		–	(26) FEATURES_HAS_BLE_DF_AOA_AOD	0x1	Bluetooth Direction Finding AoA/AoD feature availability
		–	(25) FEATURES_HAS_BLE_LONG_RANGE	0x1	Bluetooth long range feature availability
		–	(24) FEATURES_	0x1	Features availability

Address	Register Name	Register Write	Register Read	Default	Description
			FEATURES_AVAILABLE		
		(23:16) BLR_PKT_LEN_BLR_PKT_LEN	-	N/A	Packet length of the BLR packet
		(15:8) PROT_TIMER_PT_CMD	-	N/A	Protocol timer command
		(0) COMMANDS_START_SUBBAND	-	N/A	Subband selection algorithm
0x40040DE0	RF2_REG51	(31:24) FSM_MODE_RM_TX	(31:24) FSM_MODE_RM_TX	0x0	Remapped register of FSM_MODE
		(23:16) PA_PWR_RM	(23:16) PA_PWR_RM	0x0	Remapped register of PA_PWR
		(15:8) CHANNEL_RM_TX	(15:8) CHANNEL_RM_TX	0x0	Remapped register of CHANNEL
		(7:0) RATE_TX	(7:0) RATE_TX	0x0	Remapped register of BANK
0x40040DE8	RF2_REG52	(31:24) ACCESS_ADDRESS	(31:24) ACCESS_ADDRESS	0x0	Remapped register of PATTERN
		(23:16) FSM_MODE_RM_RX	(23:16) FSM_MODE_RM_RX	0x0	Remapped register of FSM_MODE
		(15:8) CHANNEL_RM_RX	(15:8) CHANNEL_RM_RX	0x0	Remapped register of CHANNEL
		(7:0) RATE_RX	(7:0) RATE_RX	0x0	Remapped register of BANK
0x40040DF0	RF2_REG53	-	(23:16) RSSI_MAX_RM	0x0	Remapped register of RSSI_MAX
		-	(15:8) RSSI_MIN_RM	0x0	Remapped register of RSSI_MIN
		-	(7:0) RSSI_AVG_RM	0x0	Remapped register of RSSI_AVG
0x40040DF4	RF2_REG54	(7:0) BLR_PACKET_LEN	-	N/A	Remapped register of BLR_PACKET_LEN
0x40040DF8	RF2_REG55	-	(7:0) ITRX_	0x0	Remapped register of ITRX_

Address	Register Name	Register Write	Register Read	Default	Description
			FEATURES		FEATURES
0x40040DFC	RF2_REG56	–	(31:24) CHIP_ID_CHIP_ID	0x30	Version of the chip
		–	(23:16) MD5_REGS_MD5_REGS	0x0	MD5 calculated on the register map file
		(15:8) SCAN_2_SCAN_2_PASSWORD	–	N/A	SCAN 2 key
		(7:0) SCAN_1_SCAN_1_PASSWORD	–	N/A	SCAN 1 key

A.25 IMPLEMENTATION CONTROL BLOCK

Address	Register Name	Register Write	Register Read	Default	Description
0xE000E004	ICB_ICTR	–	(3:0) INTLINESNUM	0x1	Number of interrupt inputs in steps of 32

A.26 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.27 TIME OF FLIGHT TIMER

Address	Register Name	Register Write	Register Read	Default	Description
0x40001E00	TOF_CFG	(21) ERROR_INT_ENABLE	(21) ERROR_INT_ENABLE	0x0	Enable the interrupt generation when an error is detected
		(20) OVERRUN_INT_ENABLE	(20) OVERRUN_INT_ENABLE	0x0	Enable the interrupt generation when a data or average data overrun is detected
		(19) AVG_DATA_INT_ENABLE	(19) AVG_DATA_INT_ENABLE	0x0	Enable the interrupt generation when a new average data is available
		(18) DATA_INT_ENABLE	(18) DATA_INT_ENABLE	0x0	Enable the interrupt generation when a new data is available
		(17) AVG_DATA_DMA_ENABLE	(17) AVG_DATA_DMA_ENABLE	0x0	Enable the DMA request when a new average data is available
		(16) DATA_DMA_ENABLE	(16) DATA_DMA_ENABLE	0x0	Enable the DMA request when a new data is available
		(14:12) AVG_CFG	(14:12) AVG_CFG	0x0	Select the amount of data for averaging
		(10:8) STOP_SRC	(10:8) STOP_SRC	0x0	Select the source of the external stop trigger
		(6:4) START_SRC	(6:4) START_SRC	0x0	Select the source of the external start trigger
		(1:0) CLK_PRESCALE	(1:0) CLK_PRESCALE	0x0	Select the time of flight timer clock prescale
0x40001E04	TOF_CTRL	–	(8) ENABLE_STATUS	0x0	Status of the time of flight timer
		(4) STOP	–	N/A	Stop the time of flight timer
		(3) START	–	N/A	Start the time of flight timer
		(2) RESET	–	N/A	Synchronously reset the time of flight timer

Address	Register Name	Register Write	Register Read	Default	Description
		(1) DISABLE	–	N/A	Disable the time of flight timer
		(0) ENABLE	–	N/A	Enable the time of flight timer
0x40001E08	TOF_STATUS	–	(23:16) AVG_DATA_STATUS	0x0	Average data timer status
		–	(13) AVG_DATA_REQ	0x0	Indicate that a new average data can be read
		–	(12) DATA_REQ	0x0	Indicate that a new data can be read
		–	(11) BUSY	0x0	Indicate if the time of flight timer is idle or busy
		–	(10) ERROR	0x0	Detect two consecutive start triggers (sticky bit)
		–	(9) AVG_DATA_OVERRUN	0x0	Indicate that an average data overrun has occurred (sticky bit)
		–	(8) DATA_OVERRUN	0x0	Indicate that a data overrun has occurred (sticky bit)
		(5) AVG_DATA_CLEAR	–	N/A	Clear the average data register and restart the average computation
		(4) MAX_DATA_CLEAR	–	N/A	Clear the max data register
		(3) MIN_DATA_CLEAR	–	N/A	Clear the min data register
		(2) ERROR_CLEAR	–	N/A	Clear the error flag
		(1) AVG_DATA_OVERRUN_CLEAR	–	N/A	Clear the average data overrun flag
		(0) DATA_OVERRUN_CLEAR	–	N/A	Clear the data overrun flag
0x40001E0C	TOF_LINK_CFG	(16:12) LINK_FORMAT	(16:12) LINK_FORMAT	0x0	Configure the link format for BLE link filtering
		(8:4) LINK_LABEL	(8:4) LINK_LABEL	0x0	Configure the link label for BLE link

Address	Register Name	Register Write	Register Read	Default	Description
					filtering
		(0) LINK_FILTER_EN	(0) LINK_FILTER_EN	0x0	Enable the BLE link filtering based on link label and format
0x40001E10	TOF_DATA	–	(19:0) DATA	0x0	Time of flight timer data (unsigned)
0x40001E14	TOF_MIN_DATA	–	(19:0) MIN_DATA	0xFFFFF	Time of flight minimum data (unsigned)
0x40001E18	TOF_MAX_DATA	–	(19:0) MAX_DATA	0x0	Time of flight maximum data (unsigned)
0x40001E1C	TOF_AVG_DATA	–	(27:8) AVG_DATA_INT	0x0	Time of flight average data (unsigned integer part)
		–	(7:0) AVG_DATA_DEC	0x0	Time of flight average data (unsigned decimal part)
0x40001EFC	TOF_ID_NUM	–	(15:8) TOF_MAJOR_ REVISION	0x1	Time of flight timer major revision number
		–	(7:0) TOF_MINOR_ REVISION	0x0	Time of flight timer minor revision number

A.28 System Control and ID register not in the SCB

Address	Register Name	Register Write	Register Read	Default	Description
0xE000E004 - 0xE000E000	SCnSCB_ICTR	–	(4:0) INTLINESNUM	0x1	Number of interrupt inputs in steps of 32

A.29 NESTED VECTOR INTERRUPT CONTROLLER

Address	Register Name	Register Write	Register Read	Default	Description
0xE000E100	NVIC_ISER0	(31) ASCC_PHASE	(31) ASCC_PHASE	0x0	ASCC_PHASE interrupt set enable
		(30) ASCC_PERIOD	(30) ASCC_PERIOD	0x0	ASCC_PERIOD interrupt set enable
		(29) CC312	(29) CC312	0x0	CC312 interrupt set enable
		(28) FPU	(28) FPU	0x0	FPU interrupt set enable
		(27) LIN0	(27) LIN0	0x0	LIN0 interrupt set enable
		(26) PCM0_ERROR	(26) PCM0_ERROR	0x0	PCM0_ERROR interrupt set enable
		(25) PCM0_RX_TX	(25) PCM0_RX_TX	0x0	PCM0_RX_TX interrupt set enable
		(24) UART0_ERROR	(24) UART0_ERROR	0x0	UART0_ERROR interrupt set enable
		(23) UART0_TX	(23) UART0_TX	0x0	UART0_TX interrupt set enable
		(22) UART0_RX	(22) UART0_RX	0x0	UART0_RX interrupt set enable
		(21) I2C1	(21) I2C1	0x0	I2C1 interrupt set enable
		(20) I2C0	(20) I2C0	0x0	I2C0 interrupt set enable
		(19) SPI1_COM	(19) SPI1_COM	0x0	SPI1_COM interrupt set enable
		(18) SPI1_TX	(18) SPI1_TX	0x0	SPI1_TX interrupt set enable
		(17) SPI1_RX	(17) SPI1_RX	0x0	SPI1_RX interrupt set enable
		(16) SPI0_COM	(16) SPI0_COM	0x0	SPI0_COM interrupt set enable
		(15) SPI0_TX	(15) SPI0_TX	0x0	SPI0_TX interrupt set enable
		(14) SPI0_RX	(14) SPI0_RX	0x0	SPI0_RX interrupt set enable
		(13) WATCHDOG	(13) WATCHDOG	0x0	WATCHDOG interrupt set enable
		(12) GPIO3	(12) GPIO3	0x0	GPIO3 interrupt set enable
		(11) GPIO2	(11) GPIO2	0x0	GPIO2 interrupt set enable

Address	Register Name	Register Write	Register Read	Default	Description
		(10) GPIO1	(10) GPIO1	0x0	GPIO1 interrupt set enable
		(9) GPIO0	(9) GPIO0	0x0	GPIO0 interrupt set enable
		(8) FIFO	(8) FIFO	0x0	FIFO 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) LSAD_MONITOR	(3) LSAD_MONITOR	0x0	LSAD_MONITOR 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
0xE000E104	NVIC_ISER1	(24) DMA3	(24) DMA3	0x0	DMA3 interrupt set enable
		(23) DMA2	(23) DMA2	0x0	DMA2 interrupt set enable
		(22) DMA1	(22) DMA1	0x0	DMA1 interrupt set enable
		(21) DMA0	(21) DMA0	0x0	DMA0 interrupt set enable
		(20) ACCESS_ERROR	(20) ACCESS_ERROR	0x0	ACCESS_ERROR interrupt set enable
		(19) FLASH0_ECC	(19) FLASH0_ECC	0x0	FLASH0_ECC interrupt set enable
		(18) FLASH0_COPY	(18) FLASH0_COPY	0x0	FLASH0_COPY interrupt set enable
		(17) RF_RXFIFO	(17) RF_RXFIFO	0x0	RF_RXFIFO interrupt set enable
		(16) RF_TXFIFO	(16) RF_TXFIFO	0x0	RF_TXFIFO interrupt set enable
		(15) RF_SYNC	(15) RF_SYNC	0x0	RF_SYNC interrupt set enable
		(14) RF_IRQ_RECEIVED	(14) RF_IRQ_ RECEIVED	0x0	RF_IRQ_RECEIVED interrupt set enable

Address	Register Name	Register Write	Register Read	Default	Description
		(13) RF_RXSTOP	(13) RF_RXSTOP	0x0	RF_RXSTOP interrupt set enable
		(12) RF_TX	(12) RF_TX	0x0	RF_TX interrupt set enable
		(11) TOF	(11) TOF	0x0	TOF interrupt set enable
		(10) BLE_COEX_RX_TX	(10) BLE_COEX_RX_TX	0x0	BLE_COEX_RX_TX interrupt set enable
		(9) BLE_COEX_IN_PROCESS	(9) BLE_COEX_IN_PROCESS	0x0	BLE_COEX_IN_PROCESS interrupt set enable
		(8) BLE_ERROR	(8) BLE_ERROR	0x0	BLE_ERROR interrupt set enable
		(7) BLE_FIFO	(7) BLE_FIFO	0x0	BLE_FIFO interrupt set enable
		(6) BLE_HSLOT	(6) BLE_HSLOT	0x0	BLE_HSLOT interrupt set enable
		(5) BLE_SLP	(5) BLE_SLP	0x0	BLE_SLP interrupt set enable
		(4) BLE_CRYPT	(4) BLE_CRYPT	0x0	BLE_CRYPT interrupt set enable
		(3) BLE_TIMESTAMP_TGT2	(3) BLE_TIMESTAMP_TGT2	0x0	BLE_TIMESTAMP_TGT2 interrupt set enable
		(2) BLE_TIMESTAMP_TGT1	(2) BLE_TIMESTAMP_TGT1	0x0	BLE_TIMESTAMP_TGT1 interrupt set enable
		(1) BLE_FINETGT	(1) BLE_FINETGT	0x0	BLE_FINETGT interrupt set enable
		(0) BLE_SW	(0) BLE_SW	0x0	BLE_SW interrupt set enable
0xE000E180	NVIC_ICER0	(31) ASCC_PHASE	(31) ASCC_PHASE	0x0	ASCC_PHASE interrupt clear enable
		(30) ASCC_PERIOD	(30) ASCC_PERIOD	0x0	ASCC_PERIOD interrupt clear enable
		(29) CC312	(29) CC312	0x0	CC312 interrupt clear enable
		(28) FPU	(28) FPU	0x0	FPU interrupt clear enable
		(27) LIN0	(27) LIN0	0x0	LIN0 interrupt clear enable
		(26) PCM0_ERROR	(26) PCM0_ERROR	0x0	PCM0_ERROR interrupt clear enable
		(25) PCM0_RX_TX	(25) PCM0_RX_TX	0x0	PCM0_RX_TX interrupt clear enable

Address	Register Name	Register Write	Register Read	Default	Description
		(24) UART0_ERROR	(24) UART0_ERROR	0x0	UART0_ERROR interrupt clear enable
		(23) UART0_TX	(23) UART0_TX	0x0	UART0_TX interrupt clear enable
		(22) UART0_RX	(22) UART0_RX	0x0	UART0_RX interrupt clear enable
		(21) I2C1	(21) I2C1	0x0	I2C1 interrupt clear enable
		(20) I2C0	(20) I2C0	0x0	I2C0 interrupt clear enable
		(19) SPI1_COM	(19) SPI1_COM	0x0	SPI1_COM interrupt clear enable
		(18) SPI1_TX	(18) SPI1_TX	0x0	SPI1_TX interrupt clear enable
		(17) SPI1_RX	(17) SPI1_RX	0x0	SPI1_RX interrupt clear enable
		(16) SPI0_COM	(16) SPI0_COM	0x0	SPI0_COM interrupt clear enable
		(15) SPI0_TX	(15) SPI0_TX	0x0	SPI0_TX interrupt clear enable
		(14) SPI0_RX	(14) SPI0_RX	0x0	SPI0_RX interrupt clear enable
		(13) WATCHDOG	(13) WATCHDOG	0x0	WATCHDOG interrupt clear enable
		(12) GPIO3	(12) GPIO3	0x0	GPIO3 interrupt clear enable
		(11) GPIO2	(11) GPIO2	0x0	GPIO2 interrupt clear enable
		(10) GPIO1	(10) GPIO1	0x0	GPIO1 interrupt clear enable
		(9) GPIO0	(9) GPIO0	0x0	GPIO0 interrupt clear enable
		(8) FIFO	(8) FIFO	0x0	FIFO 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) LSAD_MONITOR	(3) LSAD_MONITOR	0x0	LSAD_MONITOR interrupt clear enable
		(2) RTC_CLOCK	(2) RTC_CLOCK	0x0	RTC_CLOCK interrupt clear enable

Address	Register Name	Register Write	Register Read	Default	Description
		(1) RTC_ALARM	(1) RTC_ALARM	0x0	RTC_ALARM interrupt clear enable
		(0) WAKEUP	(0) WAKEUP	0x0	WAKEUP interrupt clear enable
0xE000E184	NVIC_ICER1	(24) DMA3	(24) DMA3	0x-1	DMA3 interrupt clear enable
		(23) DMA2	(23) DMA2	0x0	DMA2 interrupt clear enable
		(22) DMA1	(22) DMA1	0x0	DMA1 interrupt clear enable
		(21) DMA0	(21) DMA0	0x0	DMA0 interrupt clear enable
		(20) ACCESS_ERROR	(20) ACCESS_ERROR	0x0	ACCESS_ERROR interrupt clear enable
		(19) FLASH0_ECC	(19) FLASH0_ECC	0x0	FLASH0_ECC interrupt clear enable
		(18) FLASH0_COPY	(18) FLASH0_COPY	0x0	FLASH0_COPY interrupt clear enable
		(17) RF_RXFIFO	(17) RF_RXFIFO	0x0	RF_RXFIFO interrupt clear enable
		(16) RF_TXFIFO	(16) RF_TXFIFO	0x0	RF_TXFIFO interrupt clear enable
		(15) RF_SYNC	(15) RF_SYNC	0x0	RF_SYNC interrupt clear enable
		(14) RF_IRQ_RECEIVED	(14) RF_IRQ_ RECEIVED	0x0	RF_IRQ_RECEIVED interrupt clear enable
		(13) RF_RXSTOP	(13) RF_RXSTOP	0x0	RF_RXSTOP interrupt clear enable
		(12) RF_TX	(12) RF_TX	0x0	RF_TX interrupt clear enable
		(11) TOF	(11) TOF	0x0	TOF interrupt clear enable
		(10) BLE_COEX_RX_TX	(10) BLE_COEX_RX_TX	0x0	BLE_COEX_RX_TX interrupt clear enable
		(9) BLE_COEX_IN_ PROCESS	(9) BLE_COEX_IN_ PROCESS	0x0	BLE_COEX_IN_PROCESS interrupt clear enable
		(8) BLE_ERROR	(8) BLE_ERROR	0x0	BLE_ERROR interrupt clear enable
		(7) BLE_FIFO	(7) BLE_FIFO	0x0	BLE_FIFO interrupt clear enable
		(6) BLE_HSLOT	(6) BLE_HSLOT	0x0	BLE_HSLOT interrupt clear enable

Address	Register Name	Register Write	Register Read	Default	Description
		(5) BLE_SLP	(5) BLE_SLP	0x0	BLE_SLP interrupt clear enable
		(4) BLE_CRYPT	(4) BLE_CRYPT	0x0	BLE_CRYPT interrupt clear enable
		(3) BLE_TIMESTAMP_TGT2	(3) BLE_TIMESTAMP_TGT2	0x0	BLE_TIMESTAMP_TGT2 interrupt clear enable
		(2) BLE_TIMESTAMP_TGT1	(2) BLE_TIMESTAMP_TGT1	0x0	BLE_TIMESTAMP_TGT1 interrupt clear enable
		(1) BLE_FINETGT	(1) BLE_FINETGT	0x0	BLE_FINETGT interrupt clear enable
		(0) BLE_SW	(0) BLE_SW	0x0	BLE_SW interrupt clear enable
0xE000E200	NVIC_ISPR0	(31) ASCC_PHASE	(31) ASCC_PHASE	0x0	ASCC_PHASE interrupt set pending
		(30) ASCC_PERIOD	(30) ASCC_PERIOD	0x0	ASCC_PERIOD interrupt set pending
		(29) CC312	(29) CC312	0x0	CC312 interrupt set pending
		(28) FPU	(28) FPU	0x0	FPU interrupt set pending
		(27) LIN0	(27) LIN0	0x0	LIN0 interrupt set pending
		(26) PCM0_ERROR	(26) PCM0_ERROR	0x0	PCM0_ERROR interrupt set pending
		(25) PCM0_RX_TX	(25) PCM0_RX_TX	0x0	PCM0_RX_TX interrupt set pending
		(24) UART0_ERROR	(24) UART0_ERROR	0x0	UART0_ERROR interrupt set pending
		(23) UART0_TX	(23) UART0_TX	0x0	UART0_TX interrupt set pending
		(22) UART0_RX	(22) UART0_RX	0x0	UART0_RX interrupt set pending
		(21) I2C1	(21) I2C1	0x0	I2C1 interrupt set pending
		(20) I2C0	(20) I2C0	0x0	I2C0 interrupt set pending
		(19) SPI1_COM	(19) SPI1_COM	0x0	SPI1_COM interrupt set pending
		(18) SPI1_TX	(18) SPI1_TX	0x0	SPI1_TX interrupt set pending
		(17) SPI1_RX	(17) SPI1_RX	0x0	SPI1_RX interrupt set pending
		(16) SPI0_COM	(16) SPI0_COM	0x0	SPI0_COM interrupt set pending

Address	Register Name	Register Write	Register Read	Default	Description
		(15) SPI0_TX	(15) SPI0_TX	0x0	SPI0_TX interrupt set pending
		(14) SPI0_RX	(14) SPI0_RX	0x0	SPI0_RX interrupt set pending
		(13) WATCHDOG	(13) WATCHDOG	0x0	WATCHDOG interrupt set pending
		(12) GPIO3	(12) GPIO3	0x0	GPIO3 interrupt set pending
		(11) GPIO2	(11) GPIO2	0x0	GPIO2 interrupt set pending
		(10) GPIO1	(10) GPIO1	0x0	GPIO1 interrupt set pending
		(9) GPIO0	(9) GPIO0	0x0	GPIO0 interrupt set pending
		(8) FIFO	(8) FIFO	0x0	FIFO 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) LSAD_MONITOR	(3) LSAD_MONITOR	0x0	LSAD_MONITOR 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
0xE000E204	NVIC_ISPR1	(24) DMA3	(24) DMA3	0x0	DMA3 interrupt set pending
		(23) DMA2	(23) DMA2	0x0	DMA2 interrupt set pending
		(22) DMA1	(22) DMA1	0x0	DMA1 interrupt set pending
		(21) DMA0	(21) DMA0	0x0	DMA0 interrupt set pending
		(20) ACCESS_ERROR	(20) ACCESS_ERROR	0x0	ACCESS_ERROR interrupt set pending
		(19) FLASH0_ECC	(19) FLASH0_ECC	0x0	FLASH0_ECC interrupt set pending
		(18) FLASH0_COPY	(18) FLASH0_COPY	0x0	FLASH0_COPY interrupt set pending

Address	Register Name	Register Write	Register Read	Default	Description
		(17) RF_RXFIFO	(17) RF_RXFIFO	0x0	RF_RXFIFO interrupt set pending
		(16) RF_TXFIFO	(16) RF_TXFIFO	0x0	RF_TXFIFO interrupt set pending
		(15) RF_SYNC	(15) RF_SYNC	0x0	RF_SYNC interrupt set pending
		(14) RF_IRQ_RECEIVED	(14) RF_IRQ_ RECEIVED	0x0	RF_IRQ_RECEIVED interrupt set pending
		(13) RF_RXSTOP	(13) RF_RXSTOP	0x0	RF_RXSTOP interrupt set pending
		(12) RF_TX	(12) RF_TX	0x0	RF_TX interrupt set pending
		(11) TOF	(11) TOF	0x0	TOF interrupt set pending
		(10) BLE_COEX_RX_TX	(10) BLE_COEX_RX_TX	0x0	BLE_COEX_RX_TX interrupt set pending
		(9) BLE_COEX_IN_ PROCESS	(9) BLE_COEX_IN_ PROCESS	0x0	BLE_COEX_IN_PROCESS interrupt set pending
		(8) BLE_ERROR	(8) BLE_ERROR	0x0	BLE_ERROR interrupt set pending
		(7) BLE_FIFO	(7) BLE_FIFO	0x0	BLE_FIFO interrupt set pending
		(6) BLE_HSLOT	(6) BLE_HSLOT	0x0	BLE_HSLOT interrupt set pending
		(5) BLE_SLP	(5) BLE_SLP	0x0	BLE_SLP interrupt set pending
		(4) BLE_CRYPT	(4) BLE_CRYPT	0x0	BLE_CRYPT interrupt set pending
		(3) BLE_TIMESTAMP_ TGT2	(3) BLE_TIMESTAMP_ TGT2	0x0	BLE_TIMESTAMP_TGT2 interrupt set pending
		(2) BLE_TIMESTAMP_ TGT1	(2) BLE_TIMESTAMP_ TGT1	0x0	BLE_TIMESTAMP_TGT1 interrupt set pending
		(1) BLE_FINETGT	(1) BLE_FINETGT	0x0	BLE_FINETGT interrupt set pending
		(0) BLE_SW	(0) BLE_SW	0x0	BLE_SW interrupt set pending
0xE000E280	NVIC_ICPR0	(31) ASCC_PHASE	(31) ASCC_PHASE	0x0	ASCC_PHASE interrupt clear pending
		(30) ASCC_PERIOD	(30) ASCC_PERIOD	0x0	ASCC_PERIOD interrupt clear pending

Address	Register Name	Register Write	Register Read	Default	Description
		(29) CC312	(29) CC312	0x0	CC312 interrupt clear pending
		(28) FPU	(28) FPU	0x0	FPU interrupt clear pending
		(27) LIN0	(27) LIN0	0x0	LIN0 interrupt clear pending
		(26) PCM0_ERROR	(26) PCM0_ERROR	0x0	PCM0_ERROR interrupt clear pending
		(25) PCM0_RX_TX	(25) PCM0_RX_TX	0x0	PCM0_RX_TX interrupt clear pending
		(24) UART0_ERROR	(24) UART0_ERROR	0x0	UART0_ERROR interrupt clear pending
		(23) UART0_TX	(23) UART0_TX	0x0	UART0_TX interrupt clear pending
		(22) UART0_RX	(22) UART0_RX	0x0	UART0_RX interrupt clear pending
		(21) I2C1	(21) I2C1	0x0	I2C1 interrupt clear pending
		(20) I2C0	(20) I2C0	0x0	I2C0 interrupt clear pending
		(19) SPI1_COM	(19) SPI1_COM	0x0	SPI1_COM interrupt clear pending
		(18) SPI1_TX	(18) SPI1_TX	0x0	SPI1_TX interrupt clear pending
		(17) SPI1_RX	(17) SPI1_RX	0x0	SPI1_RX interrupt clear pending
		(16) SPI0_COM	(16) SPI0_COM	0x0	SPI0_COM interrupt clear pending
		(15) SPI0_TX	(15) SPI0_TX	0x0	SPI0_TX interrupt clear pending
		(14) SPI0_RX	(14) SPI0_RX	0x0	SPI0_RX interrupt clear pending
		(13) WATCHDOG	(13) WATCHDOG	0x0	WATCHDOG interrupt clear pending
		(12) GPIO3	(12) GPIO3	0x0	GPIO3 interrupt clear pending
		(11) GPIO2	(11) GPIO2	0x0	GPIO2 interrupt clear pending
		(10) GPIO1	(10) GPIO1	0x0	GPIO1 interrupt clear pending
		(9) GPIO0	(9) GPIO0	0x0	GPIO0 interrupt clear pending
		(8) FIFO	(8) FIFO	0x0	FIFO interrupt clear pending
		(7) TIMER3	(7) TIMER3	0x0	TIMER3 interrupt clear pending

Address	Register Name	Register Write	Register Read	Default	Description
		(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) LSAD_MONITOR	(3) LSAD_MONITOR	0x0	LSAD_MONITOR 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
0xE000E284	NVIC_ICPR1	(24) DMA3	(24) DMA3	0x0	DMA3 interrupt clear pending
		(23) DMA2	(23) DMA2	0x0	DMA2 interrupt clear pending
		(22) DMA1	(22) DMA1	0x0	DMA1 interrupt clear pending
		(21) DMA0	(21) DMA0	0x0	DMA0 interrupt clear pending
		(20) ACCESS_ERROR	(20) ACCESS_ERROR	0x0	ACCESS_ERROR interrupt clear pending
		(19) FLASH0_ECC	(19) FLASH0_ECC	0x0	FLASH0_ECC interrupt clear pending
		(18) FLASH0_COPY	(18) FLASH0_COPY	0x0	FLASH0_COPY interrupt clear pending
		(17) RF_RXFIFO	(17) RF_RXFIFO	0x0	RF_RXFIFO interrupt clear pending
		(16) RF_TXFIFO	(16) RF_TXFIFO	0x0	RF_TXFIFO interrupt clear pending
		(15) RF_SYNC	(15) RF_SYNC	0x0	RF_SYNC interrupt clear pending
		(14) RF_IRQ_RECEIVED	(14) RF_IRQ_ RECEIVED	0x0	RF_IRQ_RECEIVED interrupt clear pending
		(13) RF_RXSTOP	(13) RF_RXSTOP	0x0	RF_RXSTOP interrupt clear pending
		(12) RF_TX	(12) RF_TX	0x0	RF_TX interrupt clear pending
		(11) TOF	(11) TOF	0x0	TOF interrupt clear pending
		(10) BLE_COEX_RX_TX	(10) BLE_COEX_RX_TX	0x0	BLE_COEX_RX_TX interrupt clear pending

Address	Register Name	Register Write	Register Read	Default	Description
		(9) BLE_COEX_IN_PROCESS	(9) BLE_COEX_IN_PROCESS	0x0	BLE_COEX_IN_PROCESS interrupt clear pending
		(8) BLE_ERROR	(8) BLE_ERROR	0x0	BLE_ERROR interrupt clear pending
		(7) BLE_FIFO	(7) BLE_FIFO	0x0	BLE_FIFO interrupt clear pending
		(6) BLE_HSLOT	(6) BLE_HSLOT	0x0	BLE_HSLOT interrupt clear pending
		(5) BLE_SLP	(5) BLE_SLP	0x0	BLE_SLP interrupt clear pending
		(4) BLE_CRYPT	(4) BLE_CRYPT	0x0	BLE_CRYPT interrupt clear pending
		(3) BLE_TIMESTAMP_TGT2	(3) BLE_TIMESTAMP_TGT2	0x0	BLE_TIMESTAMP_TGT2 interrupt clear pending
		(2) BLE_TIMESTAMP_TGT1	(2) BLE_TIMESTAMP_TGT1	0x0	BLE_TIMESTAMP_TGT1 interrupt clear pending
		(1) BLE_FINETGT	(1) BLE_FINETGT	0x0	BLE_FINETGT interrupt clear pending
		(0) BLE_SW	(0) BLE_SW	0x0	BLE_SW interrupt clear pending
0xE000E300	NVIC_IABR0	–	(31) ASCC_PHASE	0x0	Set the ASCC_PHASE interrupt as active
		–	(30) ASCC_PERIOD	0x0	Set the ASCC_PERIOD interrupt as active
		–	(29) CC312	0x0	Set the CC312 interrupt as active
		–	(28) FPU	0x0	Set the FPU interrupt as active
		–	(27) LIN0	0x0	Set the LIN0 interrupt as active
		–	(26) PCM0_ERROR	0x0	Set the PCM0_ERROR interrupt as active
		–	(25) PCM0_RX_TX	0x0	Set the PCM0_RX_TX interrupt as active
		–	(24) UART0_ERROR	0x0	Set the UART0_ERROR interrupt as active
		–	(23) UART0_TX	0x0	Set the UART0_TX interrupt as active
		–	(22) UART0_RX	0x0	Set the UART0_RX interrupt as active

Address	Register Name	Register Write	Register Read	Default	Description
		–	(21) I2C1	0x0	Set the I2C1 interrupt as active
		–	(20) I2C0	0x0	Set the I2C0 interrupt as active
		–	(19) SPI1_COM	0x0	Set the SPI1_COM interrupt as active
		–	(18) SPI1_TX	0x0	Set the SPI1_TX interrupt as active
		–	(17) SPI1_RX	0x0	Set the SPI1_RX interrupt as active
		–	(16) SPI0_COM	0x0	Set the SPI0_COM interrupt as active
		–	(15) SPI0_TX	0x0	Set the SPI0_TX interrupt as active
		–	(14) SPI0_RX	0x0	Set the SPI0_RX interrupt as active
		–	(13) WATCHDOG	0x0	Set the WATCHDOG interrupt as active
		–	(12) GPIO3	0x0	Set the GPIO3 interrupt as active
		–	(11) GPIO2	0x0	Set the GPIO2 interrupt as active
		–	(10) GPIO1	0x0	Set the GPIO1 interrupt as active
		–	(9) GPIO0	0x0	Set the GPIO0 interrupt as active
		–	(8) FIFO	0x0	Set the FIFO 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) LSAD_MONITOR	0x0	Set the LSAD_MONITOR 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	–	(24) DMA3	0x0	Set the DMA3 interrupt as active
		–	(23) DMA2	0x0	Set the DMA2 interrupt as active
		–	(22) DMA1	0x0	Set the DMA1 interrupt as active
		–	(21) DMA0	0x0	Set the DMA0 interrupt as active
		–	(20) ACCESS_ERROR	0x0	Set the ACCESS_ERROR interrupt as active
		–	(19) FLASH0_ECC	0x0	Set the FLASH0_ECC interrupt as active
		–	(18) FLASH0_COPY	0x0	Set the FLASH0_COPY interrupt as active
		–	(17) RF_RXFIFO	0x0	Set the RF_RXFIFO interrupt as active
		–	(16) RF_TXFIFO	0x0	Set the RF_TXFIFO interrupt as active
		–	(15) RF_SYNC	0x0	Set the RF_SYNC interrupt as active
		–	(14) RF_IRQ_ RECEIVED	0x0	Set the RF_IRQ_RECEIVED interrupt as active
		–	(13) RF_RXSTOP	0x0	Set the RF_RXSTOP interrupt as active
		–	(12) RF_TX	0x0	Set the RF_TX interrupt as active
		–	(11) TOF	0x0	Set the TOF interrupt as active
		–	(10) BLE_COEX_RX_TX	0x0	Set the BLE_COEX_RX_TX interrupt as active
		–	(9) BLE_COEX_IN_ PROCESS	0x0	Set the BLE_COEX_IN_PROCESS interrupt as active
		–	(8) BLE_ERROR	0x0	Set the BLE_ERROR interrupt as active
		–	(7) BLE_FIFO	0x0	Set the BLE_FIFO interrupt as active
		–	(6) BLE_HSLOT	0x0	Set the BLE_HSLOT interrupt as active
		–	(5) BLE_SLP	0x0	Set the BLE_SLP interrupt as active

Address	Register Name	Register Write	Register Read	Default	Description
		–	(4) BLE_CRYPT	0x0	Set the BLE_CRYPT interrupt as active
		–	(3) BLE_TIMESTAMP_TGT2	0x0	Set the BLE_TIMESTAMP_TGT2 interrupt as active
		–	(2) BLE_TIMESTAMP_TGT1	0x0	Set the BLE_TIMESTAMP_TGT1 interrupt as active
		–	(1) BLE_FINETGT	0x0	Set the BLE_FINETGT interrupt as active
		–	(0) BLE_SW	0x0	Set the BLE_SW interrupt as active
0xE000E380	NVIC_ITNS0	(31) ASCC_PHASE	(31) ASCC_PHASE	0x0	Set the ASCC_PHASE interrupt as non-secure
		(30) ASCC_PERIOD	(30) ASCC_PERIOD	0x0	Set the ASCC_PERIOD interrupt as non-secure
		(29) CC312	(29) CC312	0x0	Set the CC312 interrupt as non-secure
		(28) FPU	(28) FPU	0x0	Set the FPU interrupt as non-secure
		(27) LIN0	(27) LIN0	0x0	Set the LIN0 interrupt as non-secure
		(26) PCM0_ERROR	(26) PCM0_ERROR	0x0	Set the PCM0_ERROR interrupt as non-secure
		(25) PCM0_RX_TX	(25) PCM0_RX_TX	0x0	Set the PCM0_RX_TX interrupt as non-secure
		(24) UART0_ERROR	(24) UART0_ERROR	0x0	Set the UART0_ERROR interrupt as non-secure
		(23) UART0_TX	(23) UART0_TX	0x0	Set the UART0_TX interrupt as non-secure
		(22) UART0_RX	(22) UART0_RX	0x0	Set the UART0_RX interrupt as non-secure
		(21) I2C1	(21) I2C1	0x0	Set the I2C1 interrupt as non-secure
		(20) I2C0	(20) I2C0	0x0	Set the I2C0 interrupt as non-secure

Address	Register Name	Register Write	Register Read	Default	Description
		(19) SPI1_COM	(19) SPI1_COM	0x0	Set the SPI1_COM interrupt as non-secure
		(18) SPI1_TX	(18) SPI1_TX	0x0	Set the SPI1_TX interrupt as non-secure
		(17) SPI1_RX	(17) SPI1_RX	0x0	Set the SPI1_RX interrupt as non-secure
		(16) SPI0_COM	(16) SPI0_COM	0x0	Set the SPI0_COM interrupt as non-secure
		(15) SPI0_TX	(15) SPI0_TX	0x0	Set the SPI0_TX interrupt as non-secure
		(14) SPI0_RX	(14) SPI0_RX	0x0	Set the SPI0_RX interrupt as non-secure
		(13) WATCHDOG	(13) WATCHDOG	0x0	Set the WATCHDOG interrupt as non-secure
		(12) GPIO3	(12) GPIO3	0x0	Set the GPIO3 interrupt as non-secure
		(11) GPIO2	(11) GPIO2	0x0	Set the GPIO2 interrupt as non-secure
		(10) GPIO1	(10) GPIO1	0x0	Set the GPIO1 interrupt as non-secure
		(9) GPIO0	(9) GPIO0	0x0	Set the GPIO0 interrupt as non-secure
		(8) FIFO	(8) FIFO	0x0	Set the FIFO interrupt as non-secure
		(7) TIMER3	(7) TIMER3	0x0	Set the TIMER3 interrupt as non-secure
		(6) TIMER2	(6) TIMER2	0x0	Set the TIMER2 interrupt as non-secure
		(5) TIMER1	(5) TIMER1	0x0	Set the TIMER1 interrupt as non-secure
		(4) TIMER0	(4) TIMER0	0x0	Set the TIMER0 interrupt as non-secure
		(3) LSAD_MONITOR	(3) LSAD_MONITOR	0x0	Set the LSAD_MONITOR interrupt as non-secure
		(2) RTC_CLOCK	(2) RTC_CLOCK	0x0	Set the RTC_CLOCK interrupt as non-secure
		(1) RTC_ALARM	(1) RTC_ALARM	0x0	Set the RTC_ALARM interrupt as non-secure

Address	Register Name	Register Write	Register Read	Default	Description
		(0) WAKEUP	(0) WAKEUP	0x0	Set the WAKEUP interrupt as non-secure
0xE000E384	NVIC_ITNS1	(24) DMA3	(24) DMA3	0x0	Set the DMA3 interrupt as non-secure
		(23) DMA2	(23) DMA2	0x0	Set the DMA2 interrupt as non-secure
		(22) DMA1	(22) DMA1	0x0	Set the DMA1 interrupt as non-secure
		(21) DMA0	(21) DMA0	0x0	Set the DMA0 interrupt as non-secure
		(20) ACCESS_ERROR	(20) ACCESS_ERROR	0x0	Set the ACCESS_ERROR interrupt as non-secure
		(19) FLASH0_ECC	(19) FLASH0_ECC	0x0	Set the FLASH0_ECC interrupt as non-secure
		(18) FLASH0_COPY	(18) FLASH0_COPY	0x0	Set the FLASH0_COPY interrupt as non-secure
		(17) RF_RXFIFO	(17) RF_RXFIFO	0x0	Set the RF_RXFIFO interrupt as non-secure
		(16) RF_TXFIFO	(16) RF_TXFIFO	0x0	Set the RF_TXFIFO interrupt as non-secure
		(15) RF_SYNC	(15) RF_SYNC	0x0	Set the RF_SYNC interrupt as non-secure
		(14) RF_IRQ_RECEIVED	(14) RF_IRQ_ RECEIVED	0x0	Set the RF_IRQ_RECEIVED interrupt as non-secure
		(13) RF_RXSTOP	(13) RF_RXSTOP	0x0	Set the RF_RXSTOP interrupt as non-secure
		(12) RF_TX	(12) RF_TX	0x0	Set the RF_TX interrupt as non-secure
		(11) TOF	(11) TOF	0x0	Set the TOF interrupt as non-secure
		(10) BLE_COEX_RX_TX	(10) BLE_COEX_RX_TX	0x0	Set the BLE_COEX_RX_TX interrupt as non-secure
		(9) BLE_COEX_IN_ PROCESS	(9) BLE_COEX_IN_ PROCESS	0x0	Set the BLE_COEX_IN_PROCESS interrupt as non-secure

Address	Register Name	Register Write	Register Read	Default	Description
		(8) BLE_ERROR	(8) BLE_ERROR	0x0	Set the BLE_ERROR interrupt as non-secure
		(7) BLE_FIFO	(7) BLE_FIFO	0x0	Set the BLE_FIFO interrupt as non-secure
		(6) BLE_HSLOT	(6) BLE_HSLOT	0x0	Set the BLE_HSLOT interrupt as non-secure
		(5) BLE_SLP	(5) BLE_SLP	0x0	Set the BLE_SLP interrupt as non-secure
		(4) BLE_CRYPT	(4) BLE_CRYPT	0x0	Set the BLE_CRYPT interrupt as non-secure
		(3) BLE_TIMESTAMP_TGT2	(3) BLE_TIMESTAMP_TGT2	0x0	Set the BLE_TIMESTAMP_TGT2 interrupt as non-secure
		(2) BLE_TIMESTAMP_TGT1	(2) BLE_TIMESTAMP_TGT1	0x0	Set the BLE_TIMESTAMP_TGT1 interrupt as non-secure
		(1) BLE_FINETGT	(1) BLE_FINETGT	0x0	Set the BLE_FINETGT interrupt as non-secure
		(0) BLE_SW	(0) BLE_SW	0x0	Set the BLE_SW interrupt as non-secure
0xE000E400	NVIC_IPR0	(31:29) LSAD_MONITOR	(31:29) LSAD_MONITOR	0x0	Configure the LSAD_MONITOR 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_IPR1	(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

Address	Register Name	Register Write	Register Read	Default	Description
		(7:5) TIMER0	(7:5) TIMER0	0x0	Configure the TIMER0 interrupt priority
0xE000E408	NVIC_IPR2	(31:29) GPIO2	(31:29) GPIO2	0x0	Configure the GPIO2 interrupt priority
		(23:21) GPIO1	(23:21) GPIO1	0x0	Configure the GPIO1 interrupt priority
		(15:13) GPIO0	(15:13) GPIO0	0x0	Configure the GPIO0 interrupt priority
		(7:5) FIFO	(7:5) FIFO	0x0	Configure the FIFO interrupt priority
0xE000E40C	NVIC_IPR3	(31:29) SPI0_TX	(31:29) SPI0_TX	0x0	Configure the SPI0_TX interrupt priority
		(23:21) SPI0_RX	(23:21) SPI0_RX	0x0	Configure the SPI0_RX interrupt priority
		(15:13) WATCHDOG	(15:13) WATCHDOG	0x0	Configure the WATCHDOG interrupt priority
		(7:5) GPIO3	(7:5) GPIO3	0x0	Configure the GPIO3 interrupt priority
0xE000E410	NVIC_IPR4	(31:29) SPI1_COM	(31:29) SPI1_COM	0x0	Configure the SPI1_COM interrupt priority
		(23:21) SPI1_TX	(23:21) SPI1_TX	0x0	Configure the SPI1_TX interrupt priority
		(15:13) SPI1_RX	(15:13) SPI1_RX	0x0	Configure the SPI1_RX interrupt priority
		(7:5) SPI0_COM	(7:5) SPI0_COM	0x0	Configure the SPI0_COM interrupt priority
0xE000E414	NVIC_IPR5	(31:29) UART0_TX	(31:29) UART0_TX	0x0	Configure the UART0_TX interrupt priority
		(23:21) UART0_RX	(23:21) UART0_RX	0x0	Configure the UART0_RX interrupt priority
		(15:13) I2C1	(15:13) I2C1	0x0	Configure the I2C1 interrupt priority
		(7:5) I2C0	(7:5) I2C0	0x0	Configure the I2C0 interrupt priority
0xE000E418	NVIC_IPR6	(31:29) LIN0	(31:29) LIN0	0x0	Configure the LIN0 interrupt priority
		(23:21) PCM0_ERROR	(23:21) PCM0_ERROR	0x0	Configure the PCM0_ERROR interrupt priority
		(15:13) PCM0_RX_TX	(15:13) PCM0_RX_TX	0x0	Configure the PCM0_RX_TX interrupt priority
		(7:5) UART0_ERROR	(7:5) UART0_ERROR	0x0	Configure the UART0_ERROR interrupt

Address	Register Name	Register Write	Register Read	Default	Description
					priority
0xE000E41C	NVIC_IPR7	(31:29) ASCC_PHASE	(31:29) ASCC_PHASE	0x0	Configure the ASCC_PHASE interrupt priority
		(23:21) ASCC_PERIOD	(23:21) ASCC_PERIOD	0x0	Configure the ASCC_PERIOD interrupt priority
		(15:13) CC312	(15:13) CC312	0x0	Configure the CC312 interrupt priority
		(7:5) FPU	(7:5) FPU	0x0	Configure the FPU interrupt priority
0xE000E420	NVIC_IPR8	(31:29) BLE_ TIMESTAMP_TGT2	(31:29) BLE_ TIMESTAMP_TGT2	0x0	Configure the BLE_TIMESTAMP_TGT2 interrupt priority
		(23:21) BLE_ TIMESTAMP_TGT1	(23:21) BLE_ TIMESTAMP_TGT1	0x0	Configure the BLE_TIMESTAMP_TGT1 interrupt priority
		(15:13) BLE_FINETGT	(15:13) BLE_FINETGT	0x0	Configure the BLE_FINETGT interrupt priority
		(7:5) BLE_SW	(7:5) BLE_SW	0x0	Configure the BLE_SW interrupt priority
0xE000E424	NVIC_IPR9	(31:29) BLE_FIFO	(31:29) BLE_FIFO	0x0	Configure the BLE_FIFO interrupt priority
		(23:21) BLE_HSLOT	(23:21) BLE_HSLOT	0x0	Configure the BLE_HSLOT interrupt priority
		(15:13) BLE_SLP	(15:13) BLE_SLP	0x0	Configure the BLE_SLP interrupt priority
		(7:5) BLE_CRYPT	(7:5) BLE_CRYPT	0x0	Configure the BLE_CRYPT interrupt priority
0xE000E428	NVIC_IPR10	(31:29) TOF	(31:29) TOF	0x0	Configure the TOF interrupt priority
		(23:21) BLE_COEX_RX_ TX	(23:21) BLE_COEX_ RX_TX	0x0	Configure the BLE_COEX_RX_TX interrupt priority
		(15:13) BLE_COEX_IN_ PROCESS	(15:13) BLE_COEX_ IN_PROCESS	0x0	Configure the BLE_COEX_IN_ PROCESS interrupt priority
		(7:5) BLE_ERROR	(7:5) BLE_ERROR	0x0	Configure the BLE_ERROR interrupt priority

Address	Register Name	Register Write	Register Read	Default	Description
0xE000E42C	NVIC_IPR11	(31:29) RF_SYNC	(31:29) RF_SYNC	0x0	Configure the RF_SYNC interrupt priority
		(23:21) RF_IRQ_ RECEIVED	(23:21) RF_IRQ_ RECEIVED	0x0	Configure the RF_IRQ_RECEIVED interrupt priority
		(15:13) RF_RXSTOP	(15:13) RF_RXSTOP	0x0	Configure the RF_RXSTOP interrupt priority
		(7:5) RF_TX	(7:5) RF_TX	0x0	Configure the RF_TX interrupt priority
0xE000E430	NVIC_IPR12	(31:29) FLASH0_ECC	(31:29) FLASH0_ECC	0x0	Configure the FLASH0_ECC interrupt priority
		(23:21) FLASH0_COPY	(23:21) FLASH0_COPY	0x0	Configure the FLASH0_COPY interrupt priority
		(15:13) RF_RXFIFO	(15:13) RF_RXFIFO	0x0	Configure the RF_RXFIFO interrupt priority
		(7:5) RF_TXFIFO	(7:5) RF_TXFIFO	0x0	Configure the RF_TXFIFO interrupt priority
0xE000E434	NVIC_IPR13	(31:29) ACCESS_ERROR	(31:29) ACCESS_ ERROR	0x0	Configure the ACCESS_ERROR interrupt priority
		(23:21) DMA0	(23:21) DMA0	0x0	Configure the DMA0 interrupt priority
		(15:13) DMA1	(15:13) DMA1	0x0	Configure the DMA1 interrupt priority
		(7:5) DMA2	(7:5) DMA2	0x0	Configure the DMA2 interrupt priority
0xE000E438	NVIC_IPR14	(7:5) DMA3	(7:5) DMA3	0x0	Configure the DMA3 interrupt priority
0xE002E100	NVIC_ISER0_NS				
0xE002E104	NVIC_ISER1_NS				
0xE002E180	NVIC_ICER0_NS				
0xE002E184	NVIC_ICER1_NS				
0xE002E200	NVIC_ISPR0_NS				

Address	Register Name	Register Write	Register Read	Default	Description
0xE002E204	NVIC_ISPR1_NS				
0xE002E280	NVIC_ICPR0_NS				
0xE002E284	NVIC_ICPR1_NS				
0xE002E300	NVIC_IABR0_NS				
0xE002E304	NVIC_IABR1_NS				
0xE002E400	NVIC_IPR0_NS				
0xE002E404	NVIC_IPR1_NS				
0xE002E408	NVIC_IPR2_NS				
0xE002E40C	NVIC_IPR3_NS				
0xE002E410	NVIC_IPR4_NS				
0xE002E414	NVIC_IPR5_NS				
0xE002E418	NVIC_IPR6_NS				
0xE002E41C	NVIC_IPR7_NS				
0xE002E420	NVIC_IPR8_NS				
0xE002E424	NVIC_IPR9_NS				
0xE002E428	NVIC_IPR10_NS				
0xE002E42C	NVIC_IPR11_NS				
0xE002E430	NVIC_IPR12_NS				
0xE002E434	NVIC_IPR13_NS				
0xE002E538	NVIC_IPR14_NS				

A.30 System Control Block

Address	Register Name	Register Write	Register Read	Default	Description
0xE000ED00	SCB_CPUID	–	(31:24) IMPLEMENTER	0x41	Implementer code
		–	(23:20) VARIANT	0x0	Implementation variant
		–	(19:16) ARCHITECTURE	0xF	Architecture number
		–	(15:4) PARTNO	0xD21	Part number
		–	(3:0) REVISION	0x4	Revision code
0xE000ED04	SCB_ICSR	(31) PENDNMISET	(31) PENDNMISET	0x0	Sets the NMI exception pending
		(30) PENDNMICLR	–	N/A	Pend NMI clear
		(28) PENDSVSET	(28) PENDSVSET	0x1	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	0x1	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
		(24) STTNS	(24) STTNS	0x0	SysTick target non-secure (RAZ/WI from non-secure state)
		–	(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
		–	(20:12) VECTPENDING	0x0	Pending external interrupt number

Address	Register Name	Register Write	Register Read	Default	Description
		–	(11) RETTOBASE	0x0	Set to 1 when the an exception handler is being run.
		–	(8:0) 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.
		(14) PRIS	(14) PRIS	0x0	Prioritize Secure exceptions
		(13) BFHFNMINS	(13) BFHFNMINS	0x0	BusFault, HardFault and NMI non-secure enable.
		(10:8) PRIGROUP	(10:8) PRIGROUP	0x0	Priority group setting. Controls how many bits in the priority register are used for pre-empty priority vs. sub-priority.
		(3) SYSRESETREQ_S	(3) SYSRESETREQ_S	0x0	System reset request secure or non secure config
		(2) SYSRESETREQ	–	N/A	Requests a chip-level system reset
		(1) VECTCLRACTIVE	–	N/A	Clears all active exception state information
0xE000ED10	SCB_SCR	(4) SEVONPEND	(4) SEVONPEND	0x0	Set to 1 to cause the WFE to wake up if a new interrupt is pended
		(3) SLEEPDEEPS	(3) SLEEPDEEPS	0x0	Sleep deep secure state access
		(2) SLEEPDEEP	(2) SLEEPDEEP	0x0	Enable SLEEPDEEP output signal when entering sleep mode
		(1) SLEEPONEXIT	(1) SLEEPONEXIT	0x0	Enable sleep on exit feature when

Address	Register Name	Register Write	Register Read	Default	Description
					returning from handler to thread mode
0xE000ED14	SCB_CCR	(18) BP	(18) BP	0x0	Branch prediction enable
		(10) STKOFHFNMIGN	(10) STKOFHFNMIGN	0x0	Stack overflow in HardFauld and NMI ignore
		(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) RESERVED_AT_1	0x1	reserved
0xE000ED18	SCB_SHPR1	(31:24) NVIC_SECURE_FAULT_PRIORITY	(31:24) NVIC_SECURE_FAULT_PRIORITY	0x0	Configure the secure fault priority
		(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_SHPR2	(31:24) NVIC_SVC_PRIORITY	(31:24) NVIC_SVC_PRIORITY	0x0	Configure the SVCall priority
0xE000ED20	SCB_SHPR3	(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_	(7:0) NVIC_	0x0	Configure the Debug Monitor priority

Address	Register Name	Register Write	Register Read	Default	Description
		DBGMONITOR_PRIORITY	DBGMONITOR_PRIORITY		
0xE000ED24	SCB_SHCSR	–	(21) HARDFAULTPENDED	0x0	Hard fault exception pended state
		–	(20) SECUREFAULTPENDED	0x0	Secure fault exception pended state
		(19) SECUREFAULTENA	(19) SECUREFAULTENA	0x0	Secure fault handler enable
		(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
		(5) NMIACT	(5) NMIACT	0x0	NMI exception hangler is active

Address	Register Name	Register Write	Register Read	Default	Description
		(4) SFAULTACT	(4) SFAULTACT	0x0	Secure fault 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
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
		(20) STKOF	(20) STKOF	0x0	Stack overflow flag
		(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
		–	(13) LSPERR	0x0	Lazy state preservation error
		(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

Address	Register Name	Register Write	Register Read	Default	Description
		–	(7) MMARVALID	0x0	Indicates if memory management fault address register is valid
		–	(5) MLSPERR	0x0	Inidicates if memory management lazy state preservation error flag
		(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	Memory Management address
0xE000ED38	SCB_BFAR	–	(31:0) NVIC_BFAR	0x0	Bus Fault address
0xE000ED8C	SCB_NSACR	–	(11) CP11	0x0	Enable non-secure access to the Floating-point extension
		(10) CP10	(10) CP10	0x0	Enables Non-secure access to the Floating-point Extension

A.31 MEMORY PROTECTION UNIT

Address	Register Name	Register Write	Register Read	Default	Description
0xE000ED90	MPU_TYPE	–	(15:8) NBR_OF_REGION	0x4	Number of regions
0xE000ED94	MPU_CTRL	(2) PRIVDEFENA	(2) PRIVDEFENA	0x0	Privileged default enable
		(1) HFNMIENA	(1) HFNMIENA	0x0	Hard fault and NMI enable
		(0) ENABLE	(0) ENABLE	0x0	Enable the MPU
0xE000ED98	MPU_RNR	–	(7:0) REGION	0x0	Region number
0xE000ED9C	MPU_RBAR	(31:5) BASE	(31:5) BASE	0x0	Base address
		(4:3) SH	(4:3) SH	0x0	Shareability
		(2:1) AP	(2:1) AP	0x0	Access permissions
		(0) XN	(0) XN	0x0	Execute never
0xE000EDA0	MPU_RLAR	(31:5) LIMIT	(31:5) LIMIT	0x0	Limit address
		(3:1) ATTRINDX	(3:1) ATTRINDX	0x0	Attribute index
		(0) EN	(0) EN	0x0	Region enable
0xE000EDA4	MPU_RBAR1	(31:5) BASE	(31:5) BASE	0x0	Base address
		(4:3) SH	(4:3) SH	0x0	Shareability
		(2:1) AP	(2:1) AP	0x0	Access permissions
		(0) XN	(0) XN	0x0	Execute never
0xE000EDA8	MPU_RLAR1	(31:5) LIMIT	(31:5) LIMIT	0x0	Limit address
		(3:1) ATTRINDX	(3:1) ATTRINDX	0x0	Attribute index
		(0) EN	(0) EN	0x0	Region enable
0xE000EDAC	MPU_RBAR2	(31:5) BASE	(31:5) BASE	0x0	Base address
		(4:3) SH	(4:3) SH	0x0	Shareability

Address	Register Name	Register Write	Register Read	Default	Description
		(2:1) AP	(2:1) AP	0x0	Access permissions
		(0) XN	(0) XN	0x0	Execute never
0xE000EDB0	MPU_RLAR2	(31:5) LIMIT	(31:5) LIMIT	0x0	Limit address
		(3:1) ATTRINDX	(3:1) ATTRINDX	0x0	Attribute index
		(0) EN	(0) EN	0x0	Region enable
0xE000EDB4	MPU_RBAR3	(31:5) BASE	(31:5) BASE	0x0	Base address
		(4:3) SH	(4:3) SH	0x0	Shareability
		(2:1) AP	(2:1) AP	0x0	Access permissions
		(0) XN	(0) XN	0x0	Execute never
0xE000EDB8	MPU_RLAR3	(31:5) LIMIT	(31:5) LIMIT	0x0	Limit address
		(3:1) ATTRINDX	(3:1) ATTRINDX	0x0	Attribute index
		(0) EN	(0) EN	0x0	Region enable
0xE000EDC0	MPU_MAIR0	(31:24) REGION3	(31:24) REGION3	0x0	Memory attribute encoding for MPU regions 3
		(23:16) REGION2	(23:16) REGION2	0x0	Memory attribute encoding for MPU regions 2
		(15:8) REGION1	(15:8) REGION1	0x0	Memory attribute encoding for MPU regions 1
		(7:0) REGION0	(7:0) REGION0	0x0	Memory attribute encoding for MPU regions 0

A.32 SECURITY ATTRIBUTION UNIT

Address	Register Name	Register Write	Register Read	Default	Description
0xE000EDD0	SAU_CTRL	(1) ALLNS	(1) ALLNS	0x0	All non-secure
		(0) ENABLE	(0) ENABLE	0x0	Enable the SAU
0xE000EDD4	SAU_TYPE	–	(7:0) NBR_OF_REGION	0x4	Number of region implemented by the SAU
0xE000EDD8	SAU_RNR	(7:0) SEL_REGION	(7:0) SEL_REGION	0x0	Selects the region currently accessed by SAU_RBAR and SAU_RLAR
0xE000EDDC	SAU_RBAR	(31:5) BASE_ADDRESS	(31:5) BASE_ADDRESS	0x0	Base address
0xE000EDE0	SAU_RLAR	(31:5) LIMIT_ADDRESS	(31:5) LIMIT_ ADDRESS	0x0	Limit address
0xE000EDE4	SAU_SFSR	–	(7) LSERR	0x0	Lazy state error flag
		–	(6) SFARVALID	0x0	Secure fault address valid
		–	(5) LSPERR	0x0	Lazy state preservation error flag
		–	(4) INVTRAN	0x0	Invalid transition flag
		–	(3) AUVIOL	0x0	Attribution unit violation flag
		–	(2) INVER	0x0	Invalid exception return flag
		–	(1) INVIS	0x0	Invalid integrity signature flag
		–	(0) INVEP	0x0	Invalid entry point
0xE000EDE8	SAU_SFAR	–	(31:0) ADDRESS	0x0	Address

A.33 DEBUG CONTROL BLOCK

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
		–	(26) S_RESART_S	0x0	Restart sticky status
		–	(25) S_RESET_ST	0x0	Core has been reset or is being rest. Bit is cleared on read.
		–	(24) S_RETIRE_ST	0x0	Retire sticky status
		–	(20) S_SDE	0x0	Secure debug enable. Indicates whether Secure invasive debug is allowed
		–	(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
		(6:0) 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				
0xE000EE04	DEBUG_DAUTHCTRL	–	(3) INTSPNIDEN	0x0	Internal Secure non-invasive debug

Address	Register Name	Register Write	Register Read	Default	Description
					enable
		–	(2) SPNIDENSEL	0x0	Secure non-invasive debug enable select
		–	(1) INTSPIDEN	0x0	Internal Secure invasive debug enable
		–	(0) SPIDENSEL	0x0	Secure invasive debug enable select
0xE000EE08	DEBUG_DSCSR	(17) CDSKEY	–	N/A	CDS write-enable key
		(16) CDS	(16) CDS	0x0	Current domain secure
		(1) SBRSEL	(1) SBRSEL	0x0	Secure banked register select
		(0) SBRSELEN	(0) SBRSELEN	0x0	Secure banked register select enable

A.34 Software Interrupt Generation

Address	Register Name	Register Write	Register Read	Default	Description
0xE000EF00	SIG_STIR	(8:0) INTID	–	N/A	Indicates the interrupt to be pended

A.35 Floating-Point Extension

Address	Register Name	Register Write	Register Read	Default	Description
0xE000EF34	FPE_FPCCR	(31) ASPEN	(31) ASPEN	0x1	When this bit is set to 1, execution of a floating-point instruction sets the CONTROL.FPCA bit to 1
		(30) LSPEN	(30) LSPEN	0x1	Enable lazy contect save of FP state
		(29) LSPENS	(29) LSPENS	0x0	This bit controls whether the LSPEN bit is writable from the Non-secure state
		(28) CLRONRET	(28) CLRONRET	0x0	Clear floating point caller saved register on exception return

Address	Register Name	Register Write	Register Read	Default	Description
		(27) CLRONRETS	(27) CLRONRETS	0x0	CLRONRET secure only
		(26) TS	(26) TS	0x0	Threat FP registers as Secure enable
		(10) UFRDY	(10) UFRDY	0x0	Indicates whether the software executing when the processor allocated the floating-point stack frame was able to set the Usage Fault exeption to pending.
		(9) SPLIMVIOL	(9) SPLIMVIOL	0x0	This bit is banked between the Security states and indicates whether the floating-point context violates the stack pointer limit that was active when lazy state preservation was activated.
		(8) MONRDY	(8) MONRDY	0x0	Indicates whether the software executing when the processor allocated the floating-point stack frame was able to set the debug monitor execption to pending.
		(7) SFRDY	(7) SFRDY	0x0	Indicates whether the software executing when the processor allocated the floating-point stack frame was able to set the secure fault exception to pending.
		(6) BRDY	(6) BRDY	0x0	Indicates whether the software executing when the processor allocated the floating-point stack frame was able to set the bus fault exception to pending.
		(5) MMRDY	(5) MMRDY	0x0	Indicates whether the software executing when the processor

Address	Register Name	Register Write	Register Read	Default	Description
					allocated the floating-point stack frame was able to set the Mem Manage exception to pending.
		(4) HFRDY	(4) HFRDY	0x0	Indicates whether the software executing when the processor allocated the floating-point stack frame was able to set the Hard fault exception to pending.
		(3) THREAD	(3) THREAD	0x0	Indicates the processor mode when it allocated the floating-point stack frame.
		(2) S	(2) S	0x1	Security status of the floating-point context
		(1) USER	(1) USER	0x0	Indicates the privilege level of the software executing when the processor allocated the floating-point stack frame
		(0) LSPACT	(0) LSPACT	0x0	Indicates whether lazy preservation of the floating-point state is active.
0xE000EF38	FPE_FPCAR	(31:3) ADDRESS	(31:3) ADDRESS	0x0	The location of the unpopulated floating-point register space allocated on an exception stack frame.
0xE000EF3C	FPE_FPDSCR	(26) AHP	(26) AHP	0x0	Alternative half-precision.
		(25) DN	(25) DN	0x0	Default NaN
		(24) FZ	(24) FZ	0x0	Flush-to-zero
		(23:22) RMODE	(23:22) RMODE	0x0	Rounding mode
0xE000EF40	FPE_MVFR0	(31:28) FPROUND	(31:28) FPROUND	0x1	All rounding modes supported

Address	Register Name	Register Write	Register Read	Default	Description
		(23:20) FPSQRT	(23:20) FPSQRT	0x1	Floating-point square root
		(19:16) FPDIVIDE	(19:16) FPDIVIDE	0x1	Floating-point divide
		(11:8) FPDP	(11:8) FPDP	0x2	Floating-point double-precision
		(7:4) FPSP	(7:4) FPSP	0x2	Floating-point single-precision
		(3:0) SIMDREG	(3:0) SIMDREG	0x1	SIMD registers. Indicates size of floating-point extension register file. (15*64 bit registers)
0xE000EF44	FPE_MVFR1	(31:28) FMAC	(31:28) FMAC	0x1	Fused multiply accumulate
		(27:24) FPHP	(27:24) FPHP	0x1	Floating-point half-precision
		(7:4) FPDNAN	(7:4) FPDNAN	0x1	Floating-point default NaN
		(3:0) FPFTZ	(3:0) FPFTZ	0x1	Floating-point flush-to-zero
0xE000EF48	FPE_MVFR2	(7:4) FPMISC	(7:4) FPMISC	0x4	Floating-point miscellaneous

Sections of this manual relating to the Arm Cortex-M33 processor have been republished from the Cortex-M33 Technical Reference (version r0p2) with permission.





Arm, the Arm logo and Cortex are trademarks or registered trademarks of Arm Ltd. Bluetooth is a registered trademark of Bluetooth SIG, Inc. All other brand names and product names appearing in this document are trademarks of their respective holders.

Copyright 2023 Semiconductor Components Industries, LLC (“onsemi”). All rights reserved. Unless agreed to differently in a separate onsemi license agreement, onsemi is providing this “Technology” (e.g. reference design kit, development product, prototype, sample, any other non-production product, software, design-IP, evaluation board, etc.) “AS IS” and does not assume any liability arising from its use; nor does onsemi convey any license to its or any third party’s intellectual property rights. This Technology is provided only to assist users in evaluation of the Technology and the recipient assumes all liability and risk associated with its use, including, but not limited to, compliance with all regulatory standards. onsemi reserves the right to make changes without further notice to any of the Technology.

The Technology is not a finished product and is as such not available for sale to consumers. Unless agreed otherwise in a separate agreement, the Technology is only intended for research, development, demonstration and evaluation purposes and should only be used in laboratory or development areas by persons with technical training and familiarity with the risks associated with handling electrical/mechanical components, systems and subsystems. The user assumes full responsibility/liability for proper and safe handling. Any other use, resale or redistribution for any other purpose is strictly prohibited.

The Technology is not designed, intended, or authorized for use in life support systems, or any FDA Class 3 medical devices or medical devices with a similar or equivalent classification in a foreign jurisdiction, or any devices intended for implantation in the human body. Should you purchase or use the Technology for any such unintended or unauthorized application, you shall indemnify and hold onsemi and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that onsemi was negligent regarding the design or manufacture of the board.

The Technology does not fall within the scope of the European Union directives regarding electromagnetic compatibility, restricted substances (RoHS), recycling (WEEE), FCC, CE or UL, and may not meet the technical requirements of these or other related directives.

THE TECHNOLOGY IS NOT WARRANTED AND PROVIDED ON AN “AS IS” BASIS ONLY. ANY WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT, ARE HEREBY EXPRESSLY DISCLAIMED.

TO THE FULLEST EXTENT PERMITTED BY APPLICABLE LAW, IN NO EVENT SHALL ONSEMI BE LIABLE TO CUSTOMER OR ANY THIRD PARTY. IN NO EVENT SHALL ONSEMI BE LIABLE FOR SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES OF ANY NATURE WHATSOEVER (INCLUDING, BUT NOT LIMITED TO, LOSS OR DISGORGEMENT OF PROFITS, LOSS OF USE AND LOSS OF GOODWILL), REGARDLESS OF WHETHER ONSEMI HAS BEEN GIVEN NOTICE OF ANY SUCH ALLEGED DAMAGES, AND REGARDLESS OF WHETHER SUCH ALLEGED DAMAGES ARE SOUGHT UNDER CONTRACT, TORT OR OTHER THEORIES OF LAW.

Do not use this Technology unless you have carefully read and agree to these limited terms and conditions. By using this Technology, you expressly agree to the limited terms and conditions. All source code is onsemi proprietary and confidential information.

PUBLICATION ORDERING INFORMATION

LITERATURE FULFILLMENT:

Literature Distribution Center for onsemi

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

onsemi Website: www.onsemi.com

Order Literature: http://www.onsemi.com/orderlit

For additional information, please contact your local Sales Representative

M-20872-006