

# **CC253x System-on-Chip Solution for 2.4-GHz IEEE 802.15.4 and ZigBee® Applications**

# **CC2540/41 System-on-Chip Solution for 2.4- GHz *Bluetooth*® low energy Applications**

## **User's Guide**



Literature Number: SWRU191F  
April 2009–Revised April 2014

<b>Preface</b> .....	<b>14</b>
<b>1 Introduction</b> .....	<b>17</b>
1.1 Overview.....	18
1.1.1 CPU and Memory .....	21
1.1.2 Clocks and Power Management .....	21
1.1.3 Peripherals .....	21
1.1.4 Radio.....	23
1.2 Applications .....	23
<b>2 8051 CPU</b> .....	<b>24</b>
2.1 8051 CPU Introduction .....	25
2.2 Memory .....	25
2.2.1 Memory Map .....	25
2.2.2 CPU Memory Space .....	27
2.2.3 Physical Memory .....	28
2.2.4 XDATA Memory Access.....	33
2.2.5 Memory Arbiter .....	33
2.3 CPU Registers .....	34
2.3.1 Data Pointers .....	34
2.3.2 Registers R0–R7 .....	35
2.3.3 Program Status Word.....	35
2.3.4 Accumulator .....	36
2.3.5 B Register .....	36
2.3.6 Stack Pointer.....	36
2.4 Instruction Set Summary .....	36
2.5 Interrupts .....	40
2.5.1 Interrupt Masking .....	41
2.5.2 Interrupt Processing.....	45
2.5.3 Interrupt Priority.....	47
<b>3 Debug Interface</b> .....	<b>50</b>
3.1 Debug Mode .....	51
3.2 Debug Communication .....	51
3.3 Debug Commands .....	53
3.3.1 Debug Configuration .....	55
3.3.2 Debug Status .....	55
3.3.3 Hardware Breakpoints .....	56
3.4 Flash Programming .....	57
3.4.1 Lock Bits.....	57
3.5 Debug Interface and Power Modes .....	57
3.6 Registers .....	59
<b>4 Power Management and Clocks</b> .....	<b>60</b>
4.1 Power Management Introduction.....	61
4.1.1 Active and Idle Modes .....	62
4.1.2 PM1 .....	62
4.1.3 PM2 .....	62

4.1.4	PM3 .....	62
4.2	Power-Management Control .....	62
4.3	Power-Management Registers .....	63
4.4	Oscillators and Clocks .....	66
4.4.1	Oscillators .....	66
4.4.2	System Clock .....	66
4.4.3	32-kHz Oscillators .....	67
4.4.4	Oscillator and Clock Registers .....	67
4.5	Timer Tick Generation .....	69
4.6	Data Retention .....	69
<b>5</b>	<b>Reset .....</b>	<b>70</b>
5.1	Power-On Reset and Brownout Detector .....	71
5.2	Clock-Loss Detector .....	71
<b>6</b>	<b>Flash Controller .....</b>	<b>72</b>
6.1	Flash Memory Organization.....	73
6.2	Flash Write .....	73
6.2.1	Flash-Write Procedure.....	73
6.2.2	Writing Multiple Times to a Word .....	74
6.2.3	DMA Flash Write .....	74
6.2.4	CPU Flash Write.....	75
6.3	Flash Page Erase .....	75
6.3.1	Performing Flash Erase From Flash Memory .....	76
6.3.2	Different Flash Page Size on CC2533 .....	76
6.4	Flash DMA Trigger .....	76
6.5	Flash Controller Registers .....	76
<b>7</b>	<b>I/O Ports .....</b>	<b>78</b>
7.1	Unused I/O Pins .....	79
7.2	Low I/O Supply Voltage .....	79
7.3	General-Purpose I/O .....	79
7.4	General-Purpose I/O Interrupts.....	79
7.5	General-Purpose I/O DMA .....	80
7.6	Peripheral I/O .....	80
7.6.1	Timer 1.....	81
7.6.2	Timer 3.....	81
7.6.3	Timer 4.....	82
7.6.4	USART 0 .....	82
7.6.5	USART 1 .....	82
7.6.6	ADC .....	83
7.6.7	Operational Amplifier and Analog Comparator .....	83
7.7	Debug Interface.....	83
7.8	32-kHz XOSC Input .....	83
7.9	Radio Test Output Signals .....	84
7.10	Power-Down Signal MUX (PMUX).....	84
7.11	I/O Registers .....	84
<b>8</b>	<b>DMA Controller .....</b>	<b>92</b>
8.1	DMA Operation .....	93
8.2	DMA Configuration Parameters .....	95
8.2.1	Source Address.....	95
8.2.2	Destination Address.....	95
8.2.3	Transfer Count.....	95
8.2.4	VLEN Setting.....	96
8.2.5	Trigger Event.....	96

8.2.6	Source and Destination Increment .....	96
8.2.7	DMA Transfer Mode .....	97
8.2.8	DMA Priority .....	97
8.2.9	Byte or Word Transfers .....	97
8.2.10	Interrupt Mask .....	97
8.2.11	Mode 8 Setting .....	97
8.3	DMA Configuration Setup .....	97
8.4	Stopping DMA Transfers .....	98
8.5	DMA Interrupts .....	98
8.6	DMA Configuration-Data Structure .....	98
8.7	DMA Memory Access .....	98
8.8	DMA Registers .....	101
<b>9</b>	<b>Timer 1 (16-Bit Timer) .....</b>	<b>103</b>
9.1	16-Bit Counter .....	104
9.2	Timer 1 Operation .....	104
9.3	Free-Running Mode .....	104
9.4	Modulo Mode .....	105
9.5	Up-and-Down Mode .....	105
9.6	Channel-Mode Control .....	105
9.7	Input Capture Mode .....	106
9.8	Output Compare Mode .....	106
9.9	IR Signal Generation and Learning .....	111
9.9.1	Introduction .....	111
9.9.2	Modulated Codes .....	111
9.9.3	Non-Modulated Codes .....	112
9.9.4	Learning .....	113
9.9.5	Other Considerations .....	113
9.10	Timer 1 Interrupts .....	113
9.11	Timer 1 DMA Triggers .....	113
9.12	Timer 1 Registers .....	114
9.13	Accessing Timer 1 Registers as Array .....	119
<b>10</b>	<b>Timer 3 and Timer 4 (8-Bit Timers) .....</b>	<b>120</b>
10.1	8-Bit Timer Counter .....	121
10.2	Timer 3 and Timer 4 Mode Control .....	121
10.2.1	Free-Running Mode .....	121
10.2.2	Down Mode .....	121
10.2.3	Modulo Mode .....	121
10.2.4	Up-and-Down Mode .....	121
10.3	Channel Mode Control .....	121
10.4	Input Capture Mode .....	122
10.5	Output Compare Mode .....	122
10.6	Timer 3 and Timer 4 Interrupts .....	122
10.7	Timer 3 and Timer 4 DMA Triggers .....	123
10.8	Timer 3 and Timer 4 Registers .....	123
<b>11</b>	<b>Sleep Timer .....</b>	<b>128</b>
11.1	General .....	129
11.2	Timer Compare .....	129
11.3	Timer Capture .....	129
11.4	Sleep Timer Registers .....	130
<b>12</b>	<b>ADC .....</b>	<b>132</b>
12.1	ADC Introduction .....	133
12.2	ADC Operation .....	133

12.2.1	ADC Inputs .....	133
12.2.2	ADC Conversion Sequences .....	134
12.2.3	Single ADC Conversion .....	134
12.2.4	ADC Operating Modes .....	134
12.2.5	ADC Conversion Results .....	135
12.2.6	ADC Reference Voltage .....	135
12.2.7	ADC Conversion Timing .....	135
12.2.8	ADC Interrupts .....	135
12.2.9	ADC DMA Triggers .....	135
12.2.10	ADC Registers .....	136
<b>13</b>	<b>Battery Monitor .....</b>	<b>139</b>
13.1	Functionality and Usage of the Battery Monitor .....	140
13.2	Using the Battery Monitor for Temperature Monitoring .....	140
13.3	Battery Monitor Registers .....	141
<b>14</b>	<b>Random-Number Generator .....</b>	<b>143</b>
14.1	Introduction .....	144
14.2	Random-Number-Generator Operation .....	144
14.2.1	Pseudorandom Sequence Generation .....	144
14.2.2	Seeding .....	144
14.2.3	CRC16 .....	144
14.3	Random-Number-Generator Registers .....	145
<b>15</b>	<b>AES Coprocessor .....</b>	<b>146</b>
15.1	AES Operation .....	147
15.2	Key and IV .....	147
15.3	Padding of Input Data .....	147
15.4	Interface to CPU .....	147
15.5	Modes of Operation .....	147
15.6	CBC-MAC .....	147
15.7	CCM Mode .....	148
15.8	AES Interrupts .....	150
15.9	AES DMA Triggers .....	150
15.10	AES Registers .....	150
<b>16</b>	<b>Watchdog Timer .....</b>	<b>152</b>
16.1	Watchdog Mode .....	153
16.2	Timer Mode .....	153
16.3	Watchdog Timer Register .....	153
<b>17</b>	<b>USART .....</b>	<b>155</b>
17.1	UART Mode .....	156
17.1.1	UART Transmit .....	156
17.1.2	UART Receive .....	156
17.1.3	UART Hardware Flow Control .....	156
17.1.4	UART Character Format .....	157
17.2	SPI Mode .....	157
17.2.1	SPI Master Operation .....	157
17.2.2	SPI Slave Operation .....	158
17.3	SSN Slave-Select Pin .....	158
17.4	Baud-Rate Generation .....	158
17.5	USART Flushing .....	159
17.6	USART Interrupts .....	159
17.7	USART DMA Triggers .....	159
17.8	USART Registers .....	159
<b>18</b>	<b>Operational Amplifier .....</b>	<b>164</b>

18.1	Description .....	165
18.2	Calibration .....	165
18.3	Clock Source .....	165
18.4	Registers .....	165
<b>19</b>	<b>Analog Comparator .....</b>	<b>166</b>
19.1	Description .....	167
19.2	Register .....	167
<b>20</b>	<b>I<sup>2</sup>C .....</b>	<b>168</b>
20.1	Operation .....	169
20.1.1	I <sup>2</sup> C Initialization and Reset .....	170
20.1.2	I <sup>2</sup> C Serial Data .....	170
20.1.3	I <sup>2</sup> C Addressing Modes .....	171
20.1.4	I <sup>2</sup> C Module Operating Modes .....	171
20.1.5	I <sup>2</sup> C Clock Generation and Synchronization .....	177
20.1.6	Bus Error .....	178
20.1.7	I <sup>2</sup> C Interrupt .....	178
20.1.8	I <sup>2</sup> C Pins .....	178
20.2	I <sup>2</sup> C Registers .....	178
<b>21</b>	<b>USB Controller .....</b>	<b>181</b>
21.1	USB Introduction .....	182
21.2	USB Enable .....	182
21.3	48-MHz USB PLL .....	182
21.4	USB Interrupts .....	183
21.5	Endpoint 0 .....	183
21.6	Endpoint-0 Interrupts .....	183
21.6.1	Error Conditions .....	184
21.6.2	SETUP Transactions (IDLE State) .....	184
21.6.3	IN Transactions (TX State) .....	184
21.6.4	OUT Transactions (RX State) .....	185
21.7	Endpoints 1–5 .....	185
21.7.1	FIFO Management .....	185
21.7.2	Double Buffering .....	186
21.7.3	FIFO Access .....	187
21.7.4	Endpoint 1–5 Interrupts .....	187
21.7.5	Bulk or Interrupt IN Endpoint .....	188
21.7.6	Isochronous IN Endpoint .....	188
21.7.7	Bulk or Interrupt OUT Endpoint .....	188
21.7.8	Isochronous OUT Endpoint .....	188
21.8	DMA .....	189
21.9	USB Reset .....	189
21.10	Suspend and Resume .....	189
21.11	Remote Wake-Up .....	189
21.12	USB Registers .....	190
<b>22</b>	<b>Timer 2 (MAC Timer) .....</b>	<b>197</b>
22.1	Timer Operation .....	198
22.1.1	General .....	198
22.1.2	Up Counter .....	198
22.1.3	Timer Overflow .....	198
22.1.4	Timer Delta Increment .....	198
22.1.5	Timer Compare .....	198
22.1.6	Overflow Count .....	198
22.1.7	Overflow-Count Update .....	199

22.1.8	Overflow-Count Overflow .....	199
22.1.9	Overflow-Count Compare.....	199
22.1.10	Capture Input .....	199
22.1.11	Long Compare (CC2541 Only) .....	199
22.2	Interrupts .....	199
22.3	Event Outputs (DMA Trigger and Radio Events) .....	200
22.4	Timer Start-and-Stop Synchronization .....	200
22.4.1	General.....	200
22.4.2	Timer Synchronous Stop .....	200
22.4.3	Timer Synchronous Start .....	201
22.5	Timer 2 Registers .....	202
<b>23</b>	<b>CC253x Radio.....</b>	<b>208</b>
23.1	RF Core .....	209
23.1.1	Interrupts.....	209
23.1.2	Interrupt Registers .....	209
23.2	FIFO Access.....	213
23.3	DMA .....	213
23.4	Memory Map .....	213
23.4.1	RXFIFO .....	214
23.4.2	TXFIFO.....	214
23.4.3	Frame-Filtering and Source-Matching Memory Map .....	214
23.5	Frequency and Channel Programming .....	215
23.6	IEEE 802.15.4-2006 Modulation Format.....	215
23.7	IEEE 802.15.4-2006 Frame Format .....	217
23.7.1	PHY Layer .....	217
23.7.2	MAC Layer.....	217
23.8	Transmit Mode .....	218
23.8.1	TX Control .....	218
23.8.2	TX State Timing.....	218
23.8.3	TXFIFO Access .....	218
23.8.4	Retransmission.....	219
23.8.5	Error Conditions.....	219
23.8.6	TX Flow Diagram .....	219
23.8.7	Transmitted Frame Processing .....	221
23.8.8	Synchronization Header .....	221
23.8.9	Frame-Length Field.....	221
23.8.10	Frame Check Sequence .....	221
23.8.11	Interrupts .....	222
23.8.12	Clear-Channel Assessment.....	222
23.8.13	Output Power Programming .....	222
23.8.14	Tips and Tricks .....	222
23.9	Receive Mode .....	222
23.9.1	RX Control .....	222
23.9.2	RX State Timing .....	223
23.9.3	Received-Frame Processing .....	223
23.9.4	Synchronization Header and Frame-Length Fields .....	224
23.9.5	Frame Filtering .....	224
23.9.6	Source Address Matching .....	227
23.9.7	Frame-Check Sequence .....	230
23.9.8	Acknowledgement Transmission .....	230
23.10	RXFIFO Access.....	232
23.10.1	Using the FIFO and FIFOP .....	232
23.10.2	Error Conditions .....	233

23.10.3	RSSI .....	233
23.10.4	Link Quality Indication .....	234
23.11	Radio-Control State Machine .....	234
23.12	Random-Number Generation .....	236
23.13	Packet Sniffing and Radio Test Output Signals .....	237
23.14	Command Strobe Processor .....	238
23.14.1	Instruction Memory .....	238
23.14.2	Data Registers .....	239
23.14.3	Program Execution .....	239
23.14.4	Interrupt Requests .....	239
23.14.5	Random Number Instruction .....	239
23.14.6	Running CSP Programs .....	239
23.14.7	Registers .....	240
23.14.8	Instruction Set Summary .....	241
23.14.9	Instruction Set Definition .....	243
23.15	Registers .....	255
23.15.1	Register Settings Update .....	256
23.15.2	Register Access Modes .....	256
23.15.3	Register Descriptions .....	257
<b>24</b>	<b>CC2540 and CC2541 Bluetooth low energy Radio .....</b>	<b>275</b>
24.1	Registers .....	276
<b>25</b>	<b>CC2541 Proprietary Mode Radio .....</b>	<b>278</b>
25.1	RF Core .....	279
25.2	Interrupts .....	279
25.2.1	Interrupt Registers .....	279
25.3	RF Core Data Memory .....	280
25.3.1	FIFOs .....	281
25.3.2	DMA .....	284
25.3.3	RAM-Based Registers .....	285
25.3.4	Variables in RAM Page 5 .....	291
25.4	Bit-Stream Processor .....	291
25.4.1	Whitening .....	291
25.4.2	CC2500-Compatible PN9 Whitening .....	292
25.4.3	CRC .....	293
25.4.4	Coprocessor Mode .....	295
25.5	Frequency and Channel Programming .....	296
25.6	Modulation Formats .....	296
25.7	Receiver .....	296
25.8	Packet Format .....	297
25.8.1	RX FIFO Packet Organization .....	299
25.8.2	TX FIFO Packet Organization .....	300
25.8.3	TX Buffers for ACK Payload .....	300
25.9	Link Layer Engine .....	301
25.9.1	Command Register .....	302
25.9.2	Radio Tasks .....	302
25.9.3	RF Test Commands .....	317
25.10	Random Number Generation .....	318
25.11	Packet Sniffing .....	319
25.12	Registers .....	320
25.12.1	Register Overview .....	320
25.12.2	Register Settings Update .....	321
25.12.3	SFR Register Descriptions .....	322
<b>26</b>	<b>Voltage Regulator .....</b>	<b>342</b>



<b>27</b>	<b>Available Software</b> .....	<b>343</b>
27.1	SmartRF™ Software for Evaluation ( <a href="http://www.ti.com/smartrfstudio">www.ti.com/smartrfstudio</a> ) .....	344
27.2	RemoTI™ Network Protocol ( <a href="http://www.ti.com/remoti">www.ti.com/remoti</a> ) .....	344
27.3	SimpliciTI™ Network Protocol ( <a href="http://www.ti.com/simpliciti">www.ti.com/simpliciti</a> ) .....	345
27.4	TIMAC Software ( <a href="http://www.ti.com/timac">www.ti.com/timac</a> ) .....	345
27.5	Z-Stack™ Software ( <a href="http://www.ti.com/z-stack">www.ti.com/z-stack</a> ) .....	346
27.6	BLE Stack Software .....	346
<b>A</b>	<b>Abbreviations</b> .....	<b>347</b>
<b>B</b>	<b>Additional Information</b> .....	<b>350</b>
B.1	Texas Instruments Low-Power RF Web Site .....	351
B.2	Low-Power RF Online Community .....	351
B.3	Texas Instruments Low-Power RF Developer Network.....	351
B.4	Low-Power RF eNewsletter .....	351
<b>C</b>	<b>References</b> .....	<b>352</b>
	<b>Revision History</b> .....	<b>353</b>

## List of Figures

1-1.	CC253x Block Diagram.....	18
1-2.	CC2540 Block Diagram .....	19
1-3.	CC2541 Block Diagram .....	20
2-1.	XDATA Memory Space (Showing SFR and DATA Mapping) .....	26
2-2.	CODE Memory Space .....	26
2-3.	CODE Memory Space for Running Code From SRAM .....	26
2-4.	Interrupt Overview.....	43
3-1.	External Debug Interface Timing .....	51
3-2.	Transmission of One Byte.....	51
3-3.	Typical Command Sequence—No Extra Wait for Response.....	52
3-4.	Typical Command Sequence. Wait for Response .....	53
3-5.	Burst Write Command (First 2 Bytes) .....	55
4-1.	Clock System Overview .....	65
6-1.	Flash Write Using DMA.....	75
8-1.	DMA Operation .....	94
8-2.	Variable Length (VLEN) Transfer Options .....	96
9-1.	Free-Running Mode .....	104
9-2.	Modulo Mode .....	105
9-3.	Up-and-Down Mode .....	105
9-4.	Output Compare Modes, Timer Free-Running Mode .....	108
9-5.	Output Compare Modes, Timer Modulo Mode.....	109
9-6.	Output Compare Modes, Timer Up-and-Down Mode .....	110
9-7.	Block Diagram of Timers in IR-Generation Mode .....	112
9-8.	Modulated Waveform Example .....	112
9-9.	IR Learning Board Diagram .....	113
11-1.	Sleep Timer Capture (Example Using Rising Edge on P0_0) .....	130
12-1.	ADC Block Diagram .....	133
14-1.	Basic Structure of the Random-Number Generator .....	144
15-1.	Message Authentication Phase Block B0 .....	148
15-2.	Authentication Flag Byte .....	148
15-3.	Message Encryption Phase Block .....	149
15-4.	Encryption Flag Byte .....	149
19-1.	Analog Comparator .....	167
20-1.	Block Diagram of the I <sup>2</sup> C Module .....	169
20-2.	I <sup>2</sup> C Bus Connection Diagram .....	170
20-3.	I <sup>2</sup> C Module Data Transfer.....	170
20-4.	Bit Transfer on I <sup>2</sup> C Bus.....	171
20-5.	I <sup>2</sup> C Module 7-Bit Addressing Format .....	171
20-6.	I <sup>2</sup> C Module Addressing Format With Repeated START Condition .....	171
20-7.	Arbitration Procedure Between Two Master Transmitters.....	177
20-8.	Synchronization of Two I <sup>2</sup> C Clock Generators During Arbitration .....	177
21-1.	USB Controller Block Diagram .....	182
21-2.	IN and OUT FIFOs .....	186
23-1.	Modulation .....	216
23-2.	I/Q Phases When Transmitting a Zero-Symbol Chip Sequence, $t_c = 0.5 \mu s$ .....	216
23-3.	Schematic View of the IEEE 802.15.4 Frame Format [1] .....	217
23-4.	Format of the Frame Control Field (FCF) .....	217

23-5. Frame Data Written to the TXFIFO.....	219
23-6. TX Flow .....	220
23-7. Single Transmitted Frame .....	221
23-8. Transmitted Synchronization Header .....	221
23-9. FCS Hardware Implementation .....	222
23-10. Single Received Frame and Transmitted Acknowledgment Frame .....	223
23-11. SFD Signal Timing.....	224
23-12. Filtering Scenarios (Exceptions Generated During Reception).....	226
23-13. Matching Algorithm for Short and Extended Addresses .....	228
23-14. Interrupts Generated by Source Address Matching .....	229
23-15. Data in RXFIFO for Different Settings .....	230
23-16. Acknowledge Frame Format .....	230
23-17. Acknowledgment Timing.....	231
23-18. Command Strobe Timing .....	231
23-19. Behavior of FIFO and FIFOP Signals .....	233
23-20. Main FSM .....	235
23-21. FFT of the Random Bytes .....	236
23-22. Histogram of 20 Million Bytes Generated With the RANDOM Instruction.....	236
23-23. Running a CSP Program .....	240
23-24. Example Hardware Structure for the R* Register Access Mode .....	256
25-1. Mapping of Radio Memory to MCU XDATA Memory Space.....	281
25-2. FIFO Pointers .....	281
25-3. PN7 Whitening .....	292
25-4. CC2500-Compatible Whitening .....	293
25-5. CRC Module.....	294
25-6. Air Interface Packet Format for Basic Mode .....	297
25-7. Air Interface Packet Format for Auto Mode .....	298
25-8. Bits of 9-Bit Header.....	298
25-9. Bits of 10-Bit Header .....	298
25-10. Structure of Packets in the RX FIFO .....	299
25-11. Structure of Packets in the TX FIFO .....	300
25-12. Timing of Packets in RX Tasks .....	316
25-13. Timing of Packets in TX Tasks.....	317
25-14. Complete Appended Packet.....	319

## List of Tables

0-1.	CC253x Family Overview .....	15
0-2.	Register Bit Conventions .....	16
2-1.	SFR Overview .....	29
2-2.	Overview of XREG Registers .....	32
2-3.	Instruction Set Summary .....	37
2-4.	Instructions That Affect Flag Settings .....	40
2-5.	Interrupts Overview .....	41
2-6.	Priority Level Setting .....	48
2-7.	Interrupt Priority Groups .....	48
2-8.	Interrupt Polling Sequence .....	49
3-1.	Debug Commands .....	53
3-2.	Debug Configuration .....	55
3-3.	Debug Status .....	55
3-4.	Relation Between <code>PCON_IDLE</code> and <code>PM_ACTIVE</code> .....	56
3-5.	Flash Lock-Protection Bit Structure Definition .....	57
4-1.	Power Modes .....	61
6-1.	Example Write Sequence .....	74
7-1.	Peripheral I/O Pin Mapping .....	81
8-1.	DMA Trigger Sources .....	98
8-2.	DMA Configuration-Data Structure .....	99
9-1.	Initial Compare Output Values (Compare Mode) .....	107
9-2.	Frequency Error Calculation for 38-kHz Carrier .....	111
10-1.	Initial Compare Output Values (Compare Mode) .....	122
13-1.	Values Showing How Different Temperatures Relate to <code>BATTMON_VOLTAGE</code> for a Typical Device .....	140
13-2.	Values for A and B (for a Typical Device) When Using the Battery monitor for Temperature Monitoring ....	141
17-1.	Commonly Used Baud-Rate Settings for 32 MHz System Clock .....	158
20-1.	Slave Transmitter Mode .....	172
20-2.	Slave Receiver Mode .....	173
20-3.	Master Transmitter Mode .....	175
20-4.	Master Receiver Mode .....	176
20-5.	Miscellaneous States .....	178
20-6.	Clock Rates Defined at 32 MHz .....	179
21-1.	USB Interrupt Flags Interrupt-Enable Mask Registers .....	183
21-2.	FIFO Sizes for EP 1–5 .....	186
22-1.	Internal Registers .....	203
23-1.	Frame Filtering and Source Matching Memory Map .....	214
23-2.	IEEE 802.15.4-2006 Symbol-to-Chip Mapping .....	216
23-3.	FSM State Mapping .....	236
23-4.	Instruction Set Summary .....	242
23-5.	Register Overview .....	255
23-6.	Registers That Require Update From Their Default Value .....	256
23-7.	Register-Bit Access Modes .....	256
25-1.	Radio RAM Pages .....	280
25-2.	Commands to FIFO via <code>RFST</code> Register .....	283
25-3.	Access to FIFO Registers .....	283
25-4.	RAM-Based Registers .....	285
25-5.	Address Structure for Auto Mode .....	289

---

25-6. Address Structure for Basic Mode.....	290
25-7. RAM-Based Registers in RAM Page 5 .....	291
25-8. Register Settings for Different CRCs.....	294
25-9. Register Settings for Some Commonly Used CRCs, Assuming Initialization With All 1s .....	295
25-10. Supported Modulation Formats, Data Rates, and Deviations.....	296
25-11. Segments for Holding ACK Payload for Each Address Entry.....	300
25-12. Commands From MCU to LL Engine via RFST Register .....	302
25-13. Timer 2 Capture Settings .....	304
25-14. End-of-Task Causes.....	304
25-15. Recommended RAM Register Settings for Start Tone .....	306
25-16. Interrupt and Counter Operation for Received Messages.....	307
25-17. Interrupt and Counter Operation for Received Messages.....	308
25-18. End-of-Receive Tasks.....	310
25-19. Interrupt and Counter Operation for Received ACK Packets .....	312
25-20. End-of-Transmit Tasks .....	313
25-21. Additional Reasons for End-of-Transmit on Clear-Channel Tasks.....	315
25-22. Packet-Sniffer Modes of Operation.....	319
25-23. XREG Register Overview.....	320
25-24. Registers That Should Be Updated From Their Default Value, Bit Rates 1 Mbps and Lower .....	321
25-25. Registers That Should Be Updated From Their Default Value, Bit Rate 2 Mbps .....	321

## Read This First

---

---

---

### About This Manual

The CC2540 and CC2541 are cost-effective, low-power, and true system-on-chip (SoC) solutions for *Bluetooth* low energy applications. They enable robust BLE master or slave nodes to be built with very low total bill-of-material costs. The CC2540 and CC2541 combine the excellent performance of a leading RF transceiver with an industry-standard enhanced 8051 MCU, in-system programmable flash memory, 8-KB RAM, and many other powerful supporting features and peripherals. The CC2540 and CC2541 are suited for systems where very low power consumption is required. Very low-power sleep modes are available. Short transition times between operating modes further enable low power consumption.

The CC2540 comes in two different versions: CC2540F128 and CC2540F256, with 128 KB and 256 KB of flash memory, respectively.

The CC2541 comes in two different versions: CC2541F128 and CC2541F256, with 128 KB and 256 KB of flash memory, respectively.

The CC2541F128/F256 comes in two different versions: CC2541F128/F256, with 128 and 256 KB of flash memory, respectively.

Combined with the *Bluetooth* low-energy protocol stack from Texas Instruments, the CC2540F128/CC2540F256 and CC2541F128/CC2541F256 constitute the market's most comprehensive single-mode Bluetooth low energy solution.

The CC253x System-on-Chip solution for 2.4 GHz is suitable for a wide range of applications. These can easily be built on top of the IEEE 802.15.4 based standard protocols (RemoTI™ network protocol, TIMAC software, and Z-Stack™ software for ZigBee® compliant solutions) or on top of the proprietary SimpliciTI™ network protocol. The usage is, however, not limited to these protocols alone. The CC253x family is, for example, also suitable for 6LoWPAN and Wireless HART implementations.

Each chapter of this manual describes details of a module or peripheral; however, not all features are present on all devices. To see the differences regarding features, see [Table 0-1](#) in the [Devices](#) section.

For detailed technical numbers, such as power consumption and RF performance, see the device-specific data sheet ([Appendix C](#)).

### Related Documentation and Software From Texas Instruments

Related documentation (for example, the CC2530 data sheet <http://www-s.ti.com/sc/techlit/swrs081> and CC2540 data sheet <http://www-s.ti.com/sc/techlit/swrs084>) can be found in [Appendix C](#).

For more information regarding software that can be used with the CC253x, CC2540, or CC2541 System-on-Chip solution (for example, SmartRF™ software for radio performance and functionality evaluation), see [Chapter 27](#), which also contains more information regarding the RemoTI network protocol, the SimpliciTI network protocol, the TIMAC software, the Z-Stack software, and the BLE stack software.

## FCC Warning

This equipment generates, uses, and can radiate radio frequency energy and has not been tested for compliance with the limits of computing devices pursuant to subpart J of part 15 of FCC rules, which are designed to provide reasonable protection against radio frequency interference. Operation of this equipment in other environments may cause interference with radio communications, in which case the user at his own expense will be required to take whatever measures may be required to correct this interference.

## If You Need Assistance

All technical support is channeled through the TI Product Information Centers (PIC) - [www.ti.com/support](http://www.ti.com/support). To send an E-mail request, please enter your contact information, along with your request at the following link – [PIC request form](#).

Also visit the Low Power RF, ZigBee, and *Bluetooth* low energy sections of the TI E2E Community ([www.ti.com/lprf-forum](http://www.ti.com/lprf-forum)), where you can easily get in touch with other CC253x, CC2540, and CC2541 users and find FAQs, Design Notes, Application Notes, Videos, and so forth.

## Glossary

Abbreviations used in this user guide can be found in [Appendix A](#).

## Devices

The CC253x System-on-Chip solution family consists of several devices. The following table provides a device overview and points out the differences regarding memory sizes and peripherals. For a complete feature list of any of the devices, see the corresponding data sheet ([Appendix C](#)).

**Table 0-1. CC253x Family Overview**

Feature	CC2530F32, -F64, -F128/, -F256	CC2531F128, CC2531F256	CC2533F32, -F64, -F96	CC2540F128, -F256	CC2541F128, -F256
FLASH_SIZE	32 KB, 64 KB, 128 KB, 256 KB	128 KB, 256 KB	32 KB, 64 KB, 96 KB	128 KB, 256 KB	128 KB, 256 KB
SRAM_SIZE	8 KB, 8 KB, 8 KB, 8 KB	8 KB, 8 KB	4 KB, 4 KB, 6 KB	8 KB	8 KB
USB	Not included	Included	Not included	Included	Not included
ADC	Included	Included	Not included	Included	Included
Battery monitor	Not included	Not included	Included	Not included	Not included
I <sup>2</sup> C	Not included	Not included	Included	Not included	Included
Operational amplifier	Included	Included	Not included	Included	Not included
Analog comparator	Included	Included	Not included	Included	Included

Legend:

FLASH\_SIZE – The size of the flash

SRAM\_SIZE – The size of the SRAM

## Register Conventions

Each SFR and XREG register is described in a separate table, where each table title contains the following information in the format indicated:

For SFR registers: REGISTER NAME (SFR address) – register description

For XREG registers: REGISTER NAME (XDATA address) – register description

Each table has five columns to describe the different register fields as described in the following:

Column 1 – Bit: Denotes which bits of the register are described and addressed in the specific row

Column 2 – Name: Specific name of the register field

Column 3 – Reset: Reset or initial value of the register field

Column 4 – R/W: Key indicating the accessibility of the bits in the field (see [Table 0-2](#) for more details)

Column 5 – Description: More details about the register field, and often a description of the functions of the different values

In the register descriptions, each register field is shown with a symbol (R/W) indicating the access mode of the register field. The register values are always given in binary notation unless prefixed by 0x, which indicates hexadecimal notation.

**Table 0-2. Register Bit Conventions**

SYMBOL	ACCESS MODE
R/W	Read and write
R	Read-only
R0	Read as 0
R1	Read as 1
W	Write-only
W0	Write as 0
W1	Write as 1
H0	Hardware clear
H1	Hardware set



## Introduction

---

---

---

As mentioned in the preface, the CC253x, CC2540, and CC2541 device family provides solutions for a wide range of applications. In order to help the user to develop these applications, this user's guide focuses on the usage of the different building blocks of the CC253x, CC2540, and CC2541 device family. For detailed device descriptions, complete feature lists, and performance numbers, see the device-specific data sheet ([Appendix C](#)).

In order to provide easy access to relevant information, the following subsections guide the reader to the different chapters in this guide.

Topic	Page
1.1 Overview .....	18
1.2 Applications .....	23

### 1.1 Overview

The block diagrams in [Figure 1-1](#), [Figure 1-2](#), and [Figure 1-3](#) show the different building blocks of the CC253x and, CC2540, and CC2541 devices. Not all features and functions of all modules or peripherals are present on all devices of the CC253x, CC2540, and CC2541; hence, see the device-specific data sheet for a device-specific block diagram.

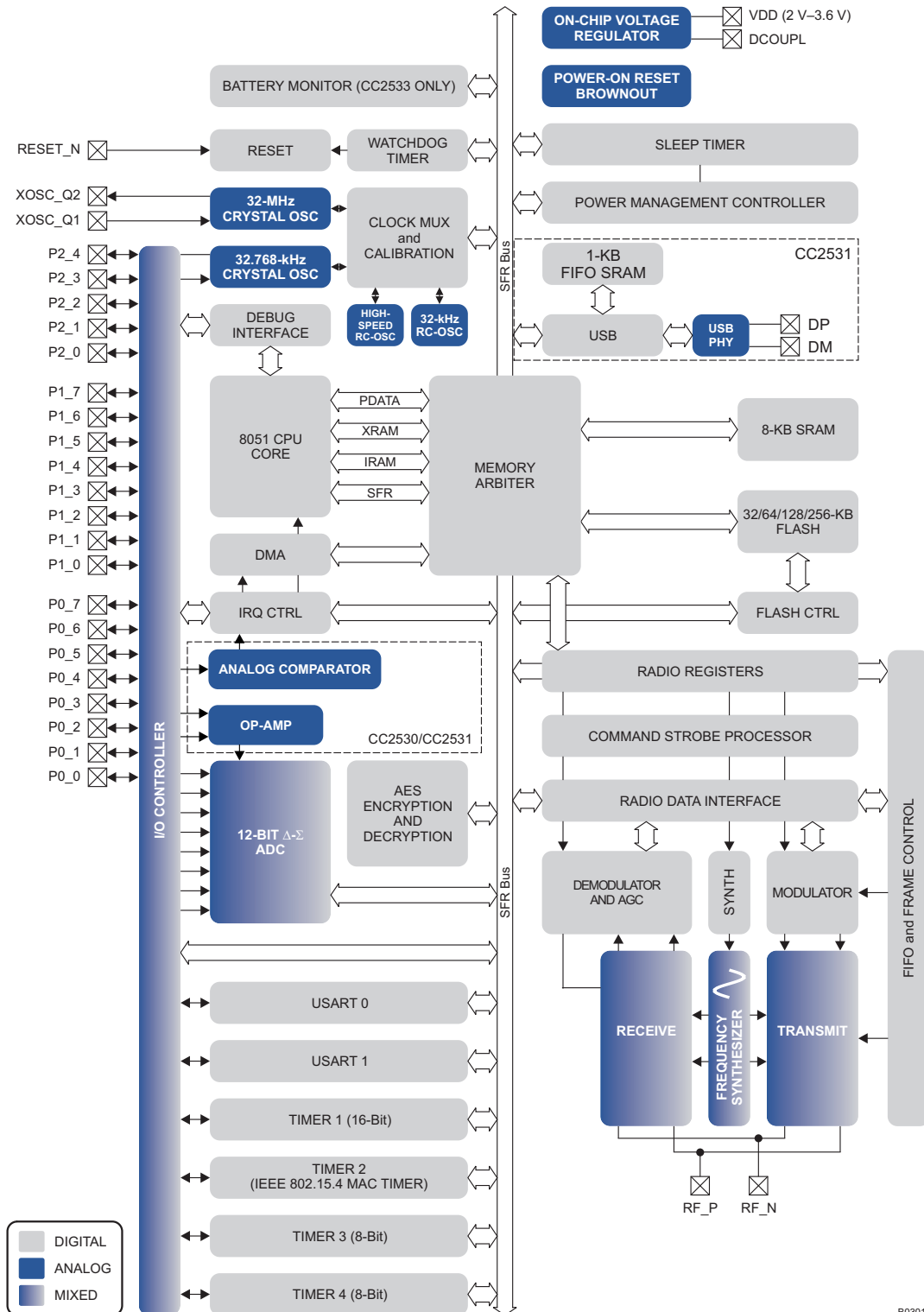
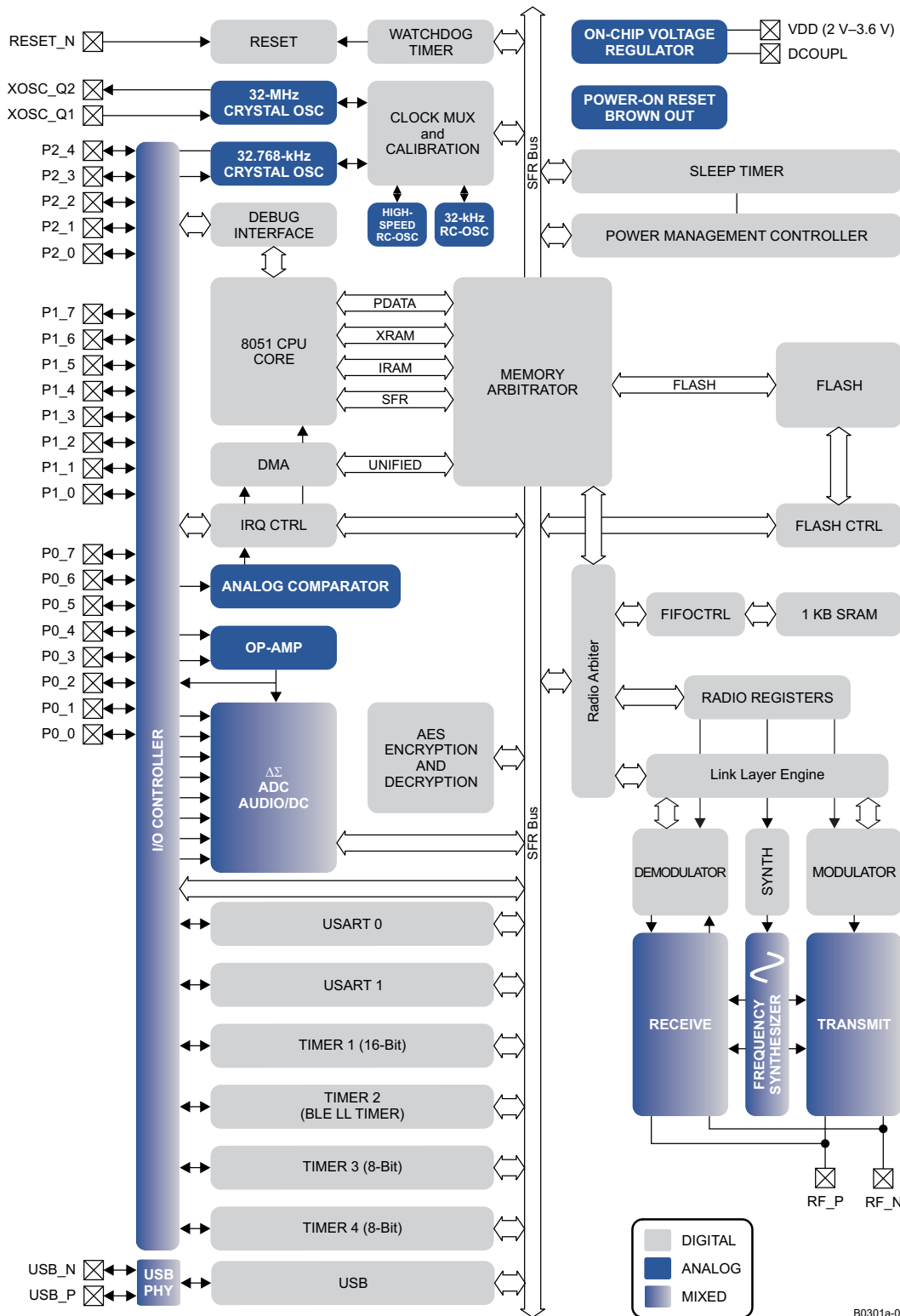


Figure 1-1. CC253x Block Diagram

B0301-03



B0301a-055

Figure 1-2. CC2540 Block Diagram

The modules can be roughly divided into one of three categories: CPU and memory related modules; modules related to peripherals, clocks, and power management; and radio-related modules.

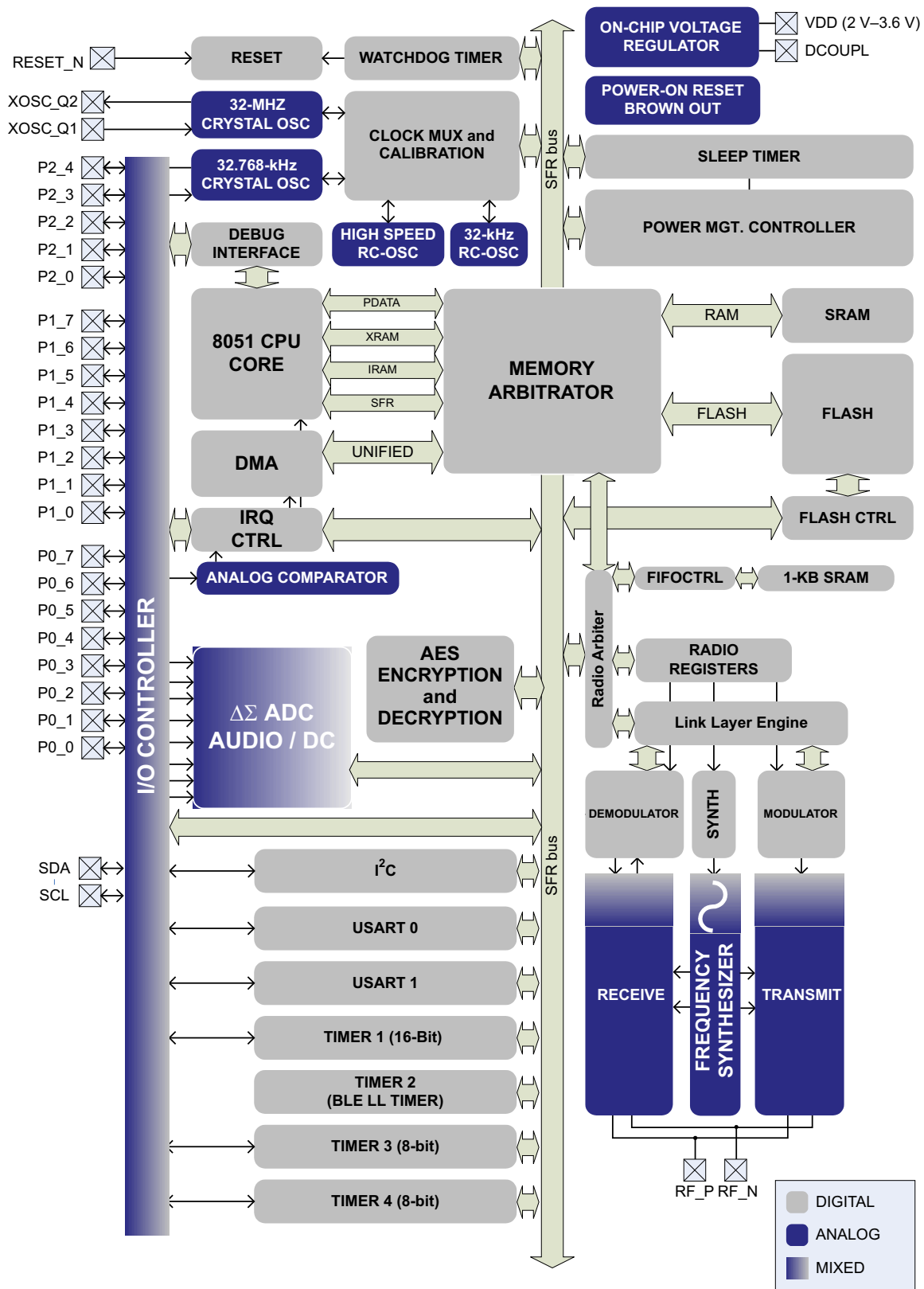


Figure 1-3. CC2541 Block Diagram

### 1.1.1 CPU and Memory

The **8051 CPU core** is a single-cycle 8051-compatible core. It has three different memory access buses (SFR, DATA, and CODE/XDATA) with single-cycle access to SFR, DATA, and the main SRAM. It also includes a debug interface and an 18-input extended interrupt unit. The detailed functionality of the CPU and the memory is addressed in [Chapter 2](#).

The **interrupt controller** services a total of 18 interrupt sources, divided into six interrupt groups, each of which is associated with one of four interrupt priorities. Any interrupt service request is serviced also when the device is in idle mode by going back to active mode. Some interrupts can also wake up the device from sleep mode (when in sleep mode, the device is in one of the three low-power modes PM1, PM2, or PM3); see [Chapter 4](#) for more details.

The **memory arbiter** is at the heart of the system, as it connects the CPU and DMA controller with the physical memories and all peripherals through the SFR bus. The memory arbiter has four memory access points, access of which can map to one of three physical memories: SRAM, flash memory, and XREG/SFR registers. The memory arbiter is responsible for performing arbitration and sequencing between simultaneous memory accesses to the same physical memory.

The **4-, 6-, or 8-KB SRAM** maps to the DATA memory space and to parts of the XDATA memory spaces. The SRAM is an ultralow-power SRAM that retains its contents in all power modes. This is an important feature for low-power applications.

The **32-, 64-, 96-, 128-, or 256-KB flash block** provides in-circuit programmable non-volatile program memory for the device, and maps into the CODE and XDATA memory spaces. In addition to holding program code and constants, the non-volatile memory allows the application to save data that must be preserved such that it is available after restarting the device. Using this feature one can, for example, use saved network-specific data to avoid the need for a full start-up and network find-and-join process.

### 1.1.2 Clocks and Power Management

The digital core and peripherals are powered by a 1.8-V low-dropout **voltage regulator** ([Chapter 26](#)). Additionally, the CC253x, CC2540, and CC2541 contain a power-management functionality that allows the use of different low-power modes (PM1, PM2, and PM3) for low-power applications with a long battery life (see [Chapter 4](#) for more details). Five different **reset** sources exist to reset the device; see [Chapter 5](#) for more details.

### 1.1.3 Peripherals

The CC253x, CC2540, and CC2541 include many different peripherals that allow the application designer to develop advanced applications. Not all peripherals are present on all devices. See [Table 0-1](#) for a listing of which peripherals are present on each device.

The **debug interface** ([Chapter 3](#)) implements a proprietary two-wire serial interface that is used for in-circuit debugging. Through this debug interface, it is possible to perform an erasure of the entire flash memory, control which oscillators are enabled, stop and start execution of the user program, execute supplied instructions on the 8051 core, set code breakpoints, and single-step through instructions in the code. Using these techniques, it is possible to perform in-circuit debugging and external flash programming elegantly.

The device contains flash memory for storage of program code. The flash memory is programmable from the user software and through the debug interface (as mentioned previously). The **flash controller** ([Chapter 6](#)) handles writing and erasing the embedded flash memory. The flash controller allows page-wise erasure and 4-bytewise programming.

The **I/O controller** ([Chapter 7](#)) is responsible for all general-purpose I/O pins. The CPU can configure whether peripheral modules control certain pins or whether they are under software control, and if so, whether each pin is configured as an input or output and if a pullup or pulldown resistor in the pad is connected. CPU interrupts can be enabled on each pin individually. Each peripheral that connects to the I/O pins can choose between two different I/O pin locations to ensure flexibility in various applications.

A versatile five-channel **DMA controller** ([Chapter 8](#)) is available in the system, accesses memory using the XDATA memory space, and thus has access to all physical memories. Each channel (trigger, priority, transfer mode, addressing mode, source and destination pointers, and transfer count) is configured with DMA descriptors anywhere in memory. Many of the hardware peripherals (AES core, flash controller, USARTs, timers, ADC interface) achieve highly efficient operation by using the DMA controller for data transfers between SFR or XREG addresses and flash or SRAM.

**Timer 1** ([Chapter 9](#)) is a 16-bit timer with timer, counter, and PWM functionality. Timer 1 has a programmable prescaler, a 16-bit period value, and five individually programmable counter or capture channels, each with a 16-bit compare value. Each of the counter or capture channels can be used as a PWM output or to capture the timing of edges on input signals. Timer 1 can also be configured in IR generation mode, where it counts Timer 3 periods and the output is ANDed with the output of Timer 3 to generate modulated consumer IR signals with minimal CPU interaction (see [Section 9.9](#)).

**Timer 2 (MAC Timer)** ([Chapter 22](#)) is specially designed for supporting an IEEE 802.15.4 MAC or other time-slotted protocol in software. The timer has a configurable timer period and a 24-bit overflow counter that can be used to keep track of the number of periods that have transpired. A 40-bit capture register is also used to record the exact time at which a start-of-frame delimiter is received or transmitted, or the exact time at which transmission ends, as well as two 16-bit output compare registers and two 24-bit overflow compare registers that can send various command strobes (start RX, start TX, etc.) at specific times to the radio modules.

**Timer 3 and Timer 4** ([Chapter 10](#)) are 8-bit timers with timer, counter, and PWM functionality. They have a programmable prescaler, an 8-bit period value, and one programmable counter channel with an 8-bit compare value. Each of the counter channels can be used as a PWM output.

The **Sleep Timer** ([Chapter 11](#)) is an ultralow-power timer that counts 32-kHz crystal oscillator or 32-kHz RC oscillator periods. The Sleep Timer runs continuously in all operating modes except power mode 3 (PM3). Typical applications of this timer are as a real-time counter or as a wake-up timer for coming out of power mode 1 (PM1) or power mode 2 (PM2).

The **ADC** ([Chapter 12](#)) supports 7 bits (30-kHz bandwidth) to 12 bits (4-kHz bandwidth) of resolution. DC and audio conversions with up to eight input channels (Port 0) are possible. The inputs can be selected as single-ended or differential. The reference voltage can be internal, AVDD, or a single-ended or differential external signal. The ADC also has a temperature-sensor input channel. The ADC can automate the process of periodic sampling or conversion over a sequence of channels.

The **battery monitor** ([Chapter 13](#)) (CC2533 only) enables simple voltage monitoring in devices that do not include an ADC. It is designed such that it is accurate in the voltage areas around 2 V, with lower resolution at higher voltages.

The **random-number generator** ([Chapter 14](#)) uses a 16-bit LFSR to generate pseudorandom numbers, which can be read by the CPU or used directly by the command strobe processor. The random-number generator can be seeded with random data from noise in the radio ADC.

The **AES coprocessor** ([Chapter 15](#)) allows the user to encrypt and decrypt data using the AES algorithm with 128-bit keys. The core is able to support the security operations required by IEEE 802.15.4 MAC security, the ZigBee network layer, and the application layer.

A built-in **Watchdog Timer** ([Chapter 16](#)) allows the device to reset itself in case the firmware hangs. When enabled by software, the Watchdog Timer must be cleared periodically; otherwise, it resets the device when it times out. It can alternatively be configured for use as a general 32-kHz timer.

**USART 0 and USART 1** ([Chapter 18](#)) are each configurable as either a SPI master or slave or as a UART. They provide double buffering on both RX and TX and hardware flow control, and are thus well suited to high-throughput full-duplex applications. Each has its own high-precision baud-rate generator, thus leaving the ordinary timers free for other uses.

The **I<sup>2</sup>C** module ([Chapter 20](#)) (CC2533 and CC2541) provides a digital peripheral connection with two pins and supports both master and slave operation.

The **USB 2.0 controller** ([Chapter 21](#)) (CC2531 and CC2540) operates at Full-Speed, 12 Mbps transfer rate. The controller has five bidirectional endpoints in addition to control endpoint 0. The endpoints support bulk, Interrupt, and Isochronous operation for implementation of a wide range of applications. The 1024 bytes of dedicated, flexible FIFO memory combined with DMA access ensures that a minimum of CPU involvement is needed for USB communication.

The **operational amplifier** ([Chapter 18](#)) (CC2530, CC2531, and CC2540) is intended to provide front-end buffering and gain for the ADC. Both the inputs as well as the output are available on pins, so the feedback network is fully customizable. A chopper-stabilized mode is available for applications that need good accuracy with high gain.

The ultralow-power **analog comparator** ([Chapter 19](#)) (CC2530, CC2531, CC2540, and CC2541) enables applications to wake up from PM2 or PM3 based on an analog signal. Both inputs are brought out to pins; the reference voltage must be provided externally. The comparator output is mapped into the digital I/O port and can be treated by the MCU as a regular digital input.

#### 1.1.4 Radio

The CC2540 and CC2541 provide a *Bluetooth* low energy-compliant radio transceiver. The RF core which controls the analog and digital radio modules is only indirectly accessible through API commands to the BLE stack. More details about the CC2540 or CC2541 BLE radio can be found in [Chapter 24](#). The CC2541 can also be run in proprietary modes; more details can be found in [Chapter 25](#).

The CC253x device family provides an **IEEE 802.15.4-compliant radio transceiver**. The RF core controls the analog radio modules. In addition, it provides an interface between the MCU and the radio which makes it possible to issue commands, read status, and automate and sequence radio events. The radio also includes a packet-filtering and address-recognition module. More details about the CC253x radio can be found in [Chapter 23](#).

## 1.2 Applications

As shown in the overview ([Section 1.1](#)), this user's guide focuses on the functionality of the different modules that are available to build different types of applications based on the CC253x, CC2540, and CC2541 device family. When looking at the complete application development process, additional information is useful. However, as this information and help is not device-specific (that is, not unique for the CC253x, CC2540, and 41 device family), see the additional information sources in the following paragraphs.

The first step is to set up the development environment (hardware, tools, and so forth) by purchasing a **development kit** (see the device-specific product Web site to find links to the relevant development kits). The development kits come with an out-of-the-box demonstration and information on how to set up the development environment; install required drivers (done easily by installing the **SmartRF software**, [Section 27.1](#)), set up the compiler tool chain, and so forth. As soon as one has installed the development environment, one is ready to start the application development.

The easiest way to write the application software is to base the application on one of the available standard protocols (**RemoTI** network protocol, [Section 27.2](#); **TIMAC** software, [Section 27.4](#); **Z-Stack** software for ZigBee-compliant solutions, [Section 27.5](#)); BLE stack software for *Bluetooth* low energy-compliant solutions [Section 27.6](#); or the proprietary **SimpliciTI** network protocol, [Section 27.3](#). They all come with several sample applications.

For the hardware layout design of the user-specific hardware, the designer can find reference designs on the different product pages ([Section B.1](#)). By copying these designs, the designer achieves optimal performance. The developed hardware can then be tested easily using the SmartRF Studio software ([Section 27.1](#)).

In case the final system should not have the expected performance, it is recommended to try out the developed software on the development kit hardware and see how it works there. To check the user-specific hardware, it is a good first step to use SmartRF Studio software to compare the development kit performance versus the user-specific hardware using the same settings.

The user can also find additional information and help by joining the **Low-Power RF Online Community** ([Section B.2](#)) and by subscribing to the **Low-Power RF eNewsletter** ([Section B.4](#)).

To contact a third-party to help with development or to use modules, check out the Texas Instruments **Low-Power RF Developer Network** ([Section B.3](#)).

## 8051 CPU

---

---

---

The System-on-Chip solution is based on an enhanced 8051 core. More details regarding the core, memory map, instruction set, and interrupts are described in the following subsections.

Topic	Page
2.1 8051 CPU Introduction .....	25
2.2 Memory .....	25
2.3 CPU Registers .....	34
2.4 Instruction Set Summary .....	36
2.5 Interrupts .....	40



## 2.1 8051 CPU Introduction

The enhanced 8051 core uses the standard 8051 instruction set. Instructions execute faster than the standard 8051 due to the following:

- One clock per instruction cycle is used as opposed to 12 clocks per instruction cycle in the standard 8051.
- Wasted bus states are eliminated.

Because an instruction cycle is aligned with memory fetch when possible, most of the single-byte instructions are performed in a single clock cycle. In addition to the speed improvement, the enhanced 8051 core also includes architectural enhancements:

- A second data pointer
- An extended 18-source interrupt unit

The 8051 core is object-code-compatible with the industry-standard 8051 microcontroller. That is, object code compiled with an industry-standard 8051 compiler or assembler executes on the 8051 core and is functionally equivalent. However, because the 8051 core uses a different instruction timing than many other 8051 variants, existing code with timing loops may require modification. Also, because the peripheral units such as timers and serial ports differ from those on other 8051 cores, code which includes instructions using the peripheral-unit SFRs does not work correctly.

Flash prefetching is not enabled by default, but improves CPU performance by up to 33%. This is at the expense of slightly increased power consumption, but in most cases improves energy consumption as it is faster. Flash prefetching can be enabled in the `FCTL` register.

## 2.2 Memory

The 8051 CPU architecture has four different memory spaces. The 8051 has separate memory spaces for program memory and data memory. The 8051 memory spaces are the following (see [Section 2.2.1](#) and [Section 2.2.2](#) for details):

**CODE.** A read-only memory space for program memory. This memory space addresses 64 KB.

**DATA.** A read-or-write data memory space that can be directly or indirectly accessed by a single-cycle CPU instruction. This memory space addresses 256 bytes. The lower 128 bytes of the DATA memory space can be addressed either directly or indirectly, the upper 128 bytes only indirectly.

**XDATA.** A read-and-write data memory space, access to which usually requires 4–5 CPU instruction cycles. This memory space addresses 64 KB. Access to XDATA memory is also slower than DATA access, as the CODE and XDATA memory spaces share a common bus on the CPU core, and instruction prefetch from CODE thus cannot be performed in parallel with XDATA accesses.

**SFR.** A read-or-write register memory space which can be directly accessed by a single CPU instruction. This memory space consists of 128 bytes. For SFR registers whose address is divisible by eight, each bit is also individually addressable.

The four different memory spaces are distinct in the 8051 architecture, but are partly overlapping in the device to ease DMA transfers and hardware debugger operation.

How the different memory spaces are mapped onto the three physical memories (flash program memory, SRAM, and memory-mapped registers) is described in [Section 2.2.1](#) and [Section 2.2.2](#).

### 2.2.1 Memory Map

The memory map differs from the standard 8051 memory map in two important aspects, as described in the following paragraphs.

First, in order to allow the DMA controller access to all physical memory and thus allow DMA transfers between the different 8051 memory spaces, parts of SFR and the DATA memory space are mapped into the XDATA memory space (see [Figure 2-1](#)).

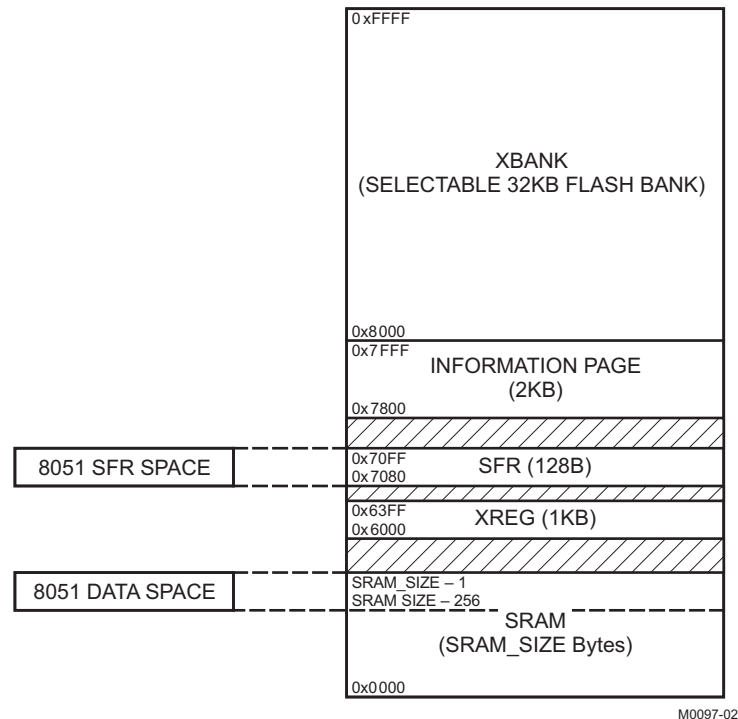
Second, two alternative schemes for CODE memory space mapping can be used. The first scheme is the standard 8051 mapping where only the program memory (that is, flash memory) is mapped to CODE memory space. This mapping is the default after a device reset and is shown in [Figure 2-2](#).

The second scheme is used for executing code from SRAM. In this mode, the SRAM is mapped into the region of 0x8000 through (0x8000 + SRAM\_SIZE – 1). The map is shown in [Figure 2-3](#). Executing code from SRAM improves performance and reduces power consumption.

The upper 32 KB of XDATA is a read-only area called XBANK (see [Figure 2-1](#)). Any of the available 32 KB flash banks can be mapped in here. This gives software access to the whole flash memory. This area is typically used to store additional constant data.

Details about mapping of all 8051 memory spaces are given in [Section 2.2.2](#).

The memory map showing how the different physical memories are mapped into the CPU memory spaces is given in [Figure 2-1](#) through [Figure 2-3](#). The number of available flash banks depends on the flash size option.



**Figure 2-1. XDATA Memory Space (Showing SFR and DATA Mapping)**

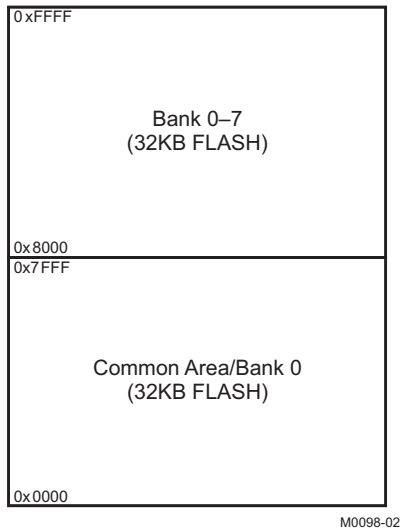


Figure 2-2. CODE Memory Space

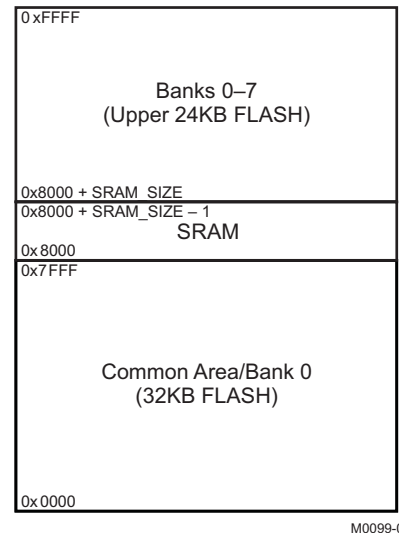


Figure 2-3. CODE Memory Space for Running Code From SRAM

## 2.2.2 CPU Memory Space

**XDATA memory space.** The XDATA memory map is given in Figure 2-1.

The SRAM is mapped into address range of 0x0000 through (SRAM\_SIZE – 1).

The XREG area is mapped into the 1 KB address range (0x6000–0x63FF). These registers are additional registers, effectively extending the SFR register space. Some peripheral registers and most of the radio control and data registers are mapped in here.

The SFR registers are mapped into address range (0x7080–0x70FF).

The flash information page (2 KB) is mapped into the address range (0x7800–0x7FFF). This is a read-only area and contains various information about the device.

The upper 32 KB of the XDATA memory space (0x8000–0xFFFF) is a read-only flash code bank (XBANK) and can be mapped to any of the available flash banks using the MEMCTR.XBANK[2:0] bits.

The mapping of flash memory, SRAM, and registers to XDATA allows the DMA controller and the CPU access to all the physical memories in a single unified address space.

Writing to unimplemented areas in the memory map (shaded in the figure) has no effect. Reading from unimplemented areas returns 0x00. Writes to read-only regions, that is, flash areas, are ignored.

**CODE memory space.** The CODE memory space is 64 KB and is divided into a common area (0x0000–0x7FFF) and a bank area (0x8000–0xFFFF) as shown in Figure 2-2. The common area is always mapped to the lower 32 KB of the physical flash memory (bank 0). The bank area can be mapped to any of the available 32-KB flash banks (from 0 to 7). The number of available flash banks depends on the flash size option. Use the flash-bank-select register, FMAP, to select the flash bank. On 32-KB devices, no flash memory can be mapped into the bank area. Reads from this region return 0x00 on these devices.

To allow program execution from SRAM, it is possible to map the available SRAM into the lower range of the bank area from 0x8000 through (0x8000 + SRAM\_SIZE – 1). The rest of the currently selected bank is still mapped into the address range from (0x8000 + SRAM\_SIZE) through 0xFFFF). Set the MEMCTR.XMAP bit to enable this feature.

**DATA memory space.** The 8-bit address range of DATA memory is mapped into the upper 256 bytes of the SRAM, that is, the address range from (SRAM\_SIZE – 256) through (SRAM\_SIZE – 1).

**SFR memory space.** The 128-entry hardware register area is accessed through this memory space. The SFR registers are also accessible through the XDATA address space at the address range (0x7080–0x70FF). Some CPU-specific SFR registers reside inside the CPU core and can only be accessed using the SFR memory space and not through the duplicate mapping into XDATA memory space. These specific SFR registers are listed in [SFR Registers](#).

### 2.2.3 Physical Memory

**RAM.** All devices contain static RAM. At power on, the content of RAM is undefined. RAM content is retained in all power modes.

**Flash Memory.** The on-chip flash memory is primarily intended to hold program code and constant data. The flash memory has the following features:

- Page size: 1 KB or 2 KB (details are given in the data sheet of the device.)
- Flash-page erase time: 20 ms
- Flash-chip (mass) erase time: 20 ms
- Flash write time (4 bytes): 20  $\mu$ s
- Data retention (at room temperature): 100 years
- Program and erase endurance: 20,000 cycles

The flash memory is organized as a set of 1 or 2 KB pages. The 16 bytes of the upper available page contain page-lock bits and the debug-lock bit. There is one lock bit for each page, except the lock-bit page which is implicitly locked when not in debug mode. When the lock bit for a page is 0, it is impossible to erase or write that page. When the debug lock bit is 0, most of the commands on the debug interface are ignored. The primary purpose of the debug lock bit is to protect the contents of the flash against read-out. The Flash Controller is used to write and erase the contents of the flash memory.

When the CPU reads instructions and constants from flash memory, it fetches the instructions through a cache. Four bytes of instructions and four bytes of constant data are cached, at 4-byte boundaries. That is, when the CPU reads from address 0x00F1 for example, bytes 0x00F0–0x00F3 are cached. A separate prefetch unit is capable of prefetching 4 additional bytes of instructions. The cache is provided mainly to reduce power consumption by reducing the amount of time the flash memory is accessed. The cache may be disabled with the `FCTL.CM[1:0]` register bits. Doing so increases power consumption and is not recommended. The execution time from flash is not cycle-accurate when using the default cache mode and the cache mode with prefetch; that is, one cannot determine exactly the number of clock cycles a set of instructions takes. To obtain cycle-accurate execution, enable the real-time cache mode and ensure all DMA transfers have low priority. The prefetch mode improves performance by up to 33%, at the expense of increased power consumption due to wasted flash reads. Typically, performance improves by 15%–20%. Total energy, however, may decrease (depending on the application) due to fewer wasted clock cycles waiting for the flash to return instructions and/or data. Prefetching is very application-dependent and requires the use of power modes to be effective.

The **Information Page** is a 2 KB read-only region that stores various device information. Among other things, it contains for IEEE 802.15.4 or *Bluetooth* low energy compliant devices a unique IEEE address from the TI range of addresses. For CC253x, this is a 64-bit IEEE address stored with least-significant byte first at XDATA address 0x780C. For CC2540 and CC2541, this is a 48-bit IEEE address stored with least-significant byte first at XDATA address 0x780E.

**SFR Registers.** The special function registers (SFRs) control several of the features of the 8051 CPU core and/or peripherals. Many of the 8051 core SFRs are identical to the standard 8051 SFRs. However, there are additional SFRs that control features that are not available in the standard 8051. The additional SFRs are used to interface with the peripheral units and RF transceiver.

[Table 2-1](#) shows the addresses of all SFRs in the device. The 8051 internal SFRs are shown with gray background, whereas the other SFRs are the SFRs specific to the device.

---

**NOTE:** All internal SFRs (shown with gray background in [Table 2-1](#)), can only be accessed through SFR space, as these registers are not mapped into XDATA space. One exception is the port registers (P0, P1, and P2) which are readable from XDATA.

---

**Table 2-1. SFR Overview**

Register Name	SFR Address	Module	Description
ADCCON1	0xB4	ADC	ADC control 1
ADCCON2	0xB5	ADC	ADC control 2
ADCCON3	0xB6	ADC	ADC control 3
ADCL	0xBA	ADC	ADC data low
ADCH	0xBB	ADC	ADC data high
RNDL	0xBC	ADC	Random number generator data low
RNDH	0xBD	ADC	Random number generator data high
ENCDI	0xB1	AES	Encryption or decryption input data
ENCDO	0xB2	AES	Encryption or decryption output data
ENCCS	0xB3	AES	Encryption or decryption control and status
P0	0x80	CPU	Port 0. Readable from XDATA (0x7080)
SP	0x81	CPU	Stack pointer
DPL0	0x82	CPU	Data pointer 0 low byte
DPH0	0x83	CPU	Data pointer 0 high byte
DPL1	0x84	CPU	Data pointer 1 low byte
DPH1	0x85	CPU	Data pointer 0 high byte
PCON	0x87	CPU	Power mode control
TCON	0x88	CPU	Interrupt flags
P1	0x90	CPU	Port 1. Readable from XDATA (0x7090)
DPS	0x92	CPU	Data pointer select
S0CON	0x98	CPU	Interrupt flags 2
IEN2	0x9A	CPU	Interrupt enable 2
S1CON	0x9B	CPU	Interrupt flags 3
P2	0xA0	CPU	Port 2. Readable from XDATA (0x70A0)
IEN0	0xA8	CPU	Interrupt enable 0
IP0	0xA9	CPU	Interrupt priority 0
IEN1	0xB8	CPU	Interrupt enable 1
IP1	0xB9	CPU	Interrupt priority 1
IRCON	0xC0	CPU	Interrupt flags 4
PSW	0xD0	CPU	Program status Word
ACC	0xE0	CPU	Accumulator
IRCON2	0xE8	CPU	Interrupt flags 5
B	0xF0	CPU	B register
DMAIRQ	0xD1	DMA	DMA interrupt flag
DMA1CFGH	0xD2	DMA	DMA channel 1–4 configuration address low
DMA1CFGH	0xD3	DMA	DMA channel 1–4 configuration address high
DMA0CFGH	0xD4	DMA	DMA channel 0 configuration address low
DMA0CFGH	0xD5	DMA	DMA channel 0 configuration address high
DMAARM	0xD6	DMA	DMA channel armed
DMAREQ	0xD7	DMA	DMA channel start request and status
—	0xAA	—	Reserved
—	0x8E	—	Reserved
—	0x99	—	Reserved
—	0xB0	—	Reserved
—	0xB7	—	Reserved
—	0xC8	—	Reserved
P0IFG	0x89	IOC	Port 0 interrupt status flag

**Table 2-1. SFR Overview (continued)**

Register Name	SFR Address	Module	Description
P1IFG	0x8A	IOC	Port 1 interrupt status flag
P2IFG	0x8B	IOC	Port 2 interrupt status flag
PICTL	0x8C	IOC	Port pins interrupt mask and edge
P0IEN	0xAB	IOC	Port 0 interrupt mask
P1IEN	0x8D	IOC	Port 1 interrupt mask
P2IEN	0xAC	IOC	Port 2 interrupt mask
P0INP	0x8F	IOC	Port 0 input mode
PERCFG	0xF1	IOC	Peripheral I/O control
APCFG	0xF2	IOC	Analog peripheral I/O configuration
P0SEL	0xF3	IOC	Port 0 function select
P1SEL	0xF4	IOC	Port 1 function select
P2SEL	0xF5	IOC	Port 2 function select
P1INP	0xF6	IOC	Port 1 input mode
P2INP	0xF7	IOC	Port 2 input mode
P0DIR	0xFD	IOC	Port 0 direction
P1DIR	0xFE	IOC	Port 1 direction
P2DIR	0xFF	IOC	Port 2 direction
PMUX	0xAE	IOC	Power-down signal mux
MPAGE	0x93	MEMORY	Memory page select
MEMCTR	0xC7	MEMORY	Memory system control
FMAP	0x9F	MEMORY	Flash-memory bank mapping
RFIRQF1	0x91	RF	RF interrupt flags MSB
RFD	0xD9	RF	RF data
RFST	0xE1	RF	RF command strobe
RFIRQF0	0xE9	RF	RF interrupt flags LSB
RFERRF	0xBF	RF	RF error interrupt flags
ST0	0x95	ST	Sleep Timer 0
ST1	0x96	ST	Sleep Timer 1
ST2	0x97	ST	Sleep Timer 2
STLOAD	0xAD	ST	Sleep-timer load status
SLEEPCMD	0xBE	PMC	Sleep-mode control command
SLEEPSTA	0x9D	PMC	Sleep-mode control status
CLKCONCMD	0xC6	PMC	Clock control command
CLKCONSTA	0x9E	PMC	Clock control status
T1CC0L	0xDA	Timer 1	Timer 1 channel 0 capture or compare value low
T1CC0H	0xDB	Timer 1	Timer 1 channel 0 capture or compare value high
T1CC1L	0xDC	Timer 1	Timer 1 channel 1 capture or compare value low
T1CC1H	0xDD	Timer 1	Timer 1 channel 1 capture or compare value high
T1CC2L	0xDE	Timer 1	Timer 1 channel 2 capture or compare value low
T1CC2H	0xDF	Timer 1	Timer 1 channel 2 capture or compare value high
T1CNTL	0xE2	Timer 1	Timer 1 counter low
T1CNTH	0xE3	Timer 1	Timer 1 counter high
T1CTL	0xE4	Timer 1	Timer 1 control and status
T1CCTL0	0xE5	Timer 1	Timer 1 channel 0 capture or compare control
T1CCTL1	0xE6	Timer 1	Timer 1 channel 1 capture or compare control
T1CCTL2	0xE7	Timer 1	Timer 1 channel 2 capture or compare control
T1STAT	0xAF	Timer 1	Timer 1 status

**Table 2-1. SFR Overview (continued)**

Register Name	SFR Address	Module	Description
T2CTRL	0x94	Timer 2	Timer 2 control
T2EVTCFG	0x9C	Timer 2	Timer 2 event configuration
T2IRQF	0xA1	Timer 2	Timer 2 interrupt flags
T2M0	0xA2	Timer 2	Timer 2 multiplexed register 0
T2M1	0xA3	Timer 2	Timer 2 multiplexed register 1
T2MOVFO	0xA4	Timer 2	Timer 2 multiplexed overflow register 0
T2MOVFI	0xA5	Timer 2	Timer 2 multiplexed overflow register 1
T2MOVFI2	0xA6	Timer 2	Timer 2 multiplexed overflow register 2
T2IRQM	0xA7	Timer 2	Timer 2 interrupt mask
T2MSEL	0xC3	Timer 2	Timer 2 multiplex select
T3CNT	0xCA	Timer 3	Timer 3 counter
T3CTL	0xCB	Timer 3	Timer 3 control
T3CCTL0	0xCC	Timer 3	Timer 3 channel 0 compare control
T3CC0	0xCD	Timer 3	Timer 3 channel 0 compare value
T3CCTL1	0xCE	Timer 3	Timer 3 channel 1 compare control
T3CC1	0xCF	Timer 3	Timer 3 channel 1 compare value
T4CNT	0xEA	Timer 4	Timer 4 counter
T4CTL	0xEB	Timer 4	Timer 4 control
T4CCTL0	0xEC	Timer 4	Timer 4 channel 0 compare control
T4CC0	0xED	Timer 4	Timer 4 channel 0 compare value
T4CCTL1	0xEE	Timer 4	Timer 4 channel 1 compare control
T4CC1	0xEF	Timer 4	Timer 4 channel 1 compare value
TIMIF	0xD8	TMINT	Timers 1,3, 4 joint interrupt mask or flags
U0CSR	0x86	USART 0	USART 0 control and status
U0DBUF	0xC1	USART 0	USART 0 receive and transmit data buffer
U0BAUD	0xC2	USART 0	USART 0 baud-rate control
U0UCR	0xC4	USART 0	USART 0 UART control
U0GCR	0xC5	USART 0	USART 0 generic control
U1CSR	0xF8	USART 1	USART 1 control and status
U1DBUF	0xF9	USART 1	USART 1 receive and transmit data buffer
U1BAUD	0xFA	USART 1	USART 1 baud-rate control
U1UCR	0xFB	USART 1	USART 1 UART control
U1GCR	0xFC	USART 1	USART 1 generic control
WDCTL	0xC9	WDT	Watchdog Timer control

**XREG Registers.** The XREG registers are additional registers in the XDATA memory space. These registers are mainly used for radio configuration and control. For more details regarding each register, see the corresponding module or peripheral chapter. [Table 2-2](#) gives a descriptive overview of the register address space.

**Table 2-2. Overview of XREG Registers**

XDATA Address	Register Name	Description
0x6000–0x61FF	—	Radio registers (see CC253x Radio <a href="#">Section 23.15</a> or CC2540 Radio <a href="#">Section 24.1</a> or CC2541 Radio <a href="#">Section 25.12</a> for complete list)
0x61A6	MONMUX	Battery monitor MUX (CC2533)
	OPAMPMC	Operational amplifier mode control (CC2530, CC2531)
0x61AD	OPAMPMC	Operational amplifier mode control (CC2540)
0x6200–0x622B	—	USB registers (see <a href="#">Section 21.12</a> for complete list)
0x6230	I2CCFG	I <sup>2</sup> C control
0x6231	I2CSTAT	I <sup>2</sup> C status
0x6232	I2CDATA	I <sup>2</sup> C data
0x6233	I2CADDR	I <sup>2</sup> C own slave address
0x6234	I2CWC	Wrapper control
0x6235	I2CIO	GPIO
0x6243	OBSSEL0	Observation output control register 0
0x6244	OBSSEL1	Observation output control register 1
0x6245	OBSSEL2	Observation output control register 2
0x6246	OBSSEL3	Observation output control register 3
0x6247	OBSSEL4	Observation output control register 4
0x6248	OBSSEL5	Observation output control register 5
0x6249	CHVER	Chip version
0x624A	CHIPID	Chip identification
0x624B	TR0	Test register 0
0x6260	DBGDATA	Debug interface write data
0x6262	SRCRC	Sleep reset CRC
0x6264	BATTMON	Battery monitor
0x6265	IVCTRL	Analog control register
0x6270	FCTL	Flash control
0x6271	FADDRL	Flash address low
0x6272	FADDRH	Flash address high
0x6273	FWDATA	Flash write data
0x6276	CHIPINFO0	Chip information byte 0
0x6277	CHIPINFO1	Chip information byte 1
0x6281	IRCTL	Timer 1 IR generation control
0x6290	CLD	Clock-loss detection
0x62A0	T1CCTL0	Timer 1 channel 0 capture or compare control (additional XREG mapping of SFR register)
0x62A1	T1CCTL1	Timer 1 channel 1 capture or compare control (additional XREG mapping of SFR register)
0x62A2	T1CCTL2	Timer 1 channel 2 capture or compare control (additional XREG mapping of SFR register)
0x62A3	T1CCTL3	Timer 1 channel 3 capture or compare control
0x62A4	T1CCTL4	Timer 1 channel 4 capture or compare control
0x62A6	T1CC0L	Timer 1 channel 0 capture or compare value low (additional XREG mapping of SFR register)
0x62A7	T1CC0H	Timer 1 channel 0 capture or compare value high (additional XREG mapping of SFR register)



**Table 2-2. Overview of XREG Registers (continued)**

XDATA Address	Register Name	Description
0x62A8	T1CC1L	Timer 1 channel 1 capture or compare value low (additional XREG mapping of SFR register)
0x62A9	T1CC1H	Timer 1 channel 1 capture or compare value high (additional XREG mapping of SFR register)
0x62AA	T1CC2L	Timer 1 channel 2 capture or compare value low (additional XREG mapping of SFR register)
0x62AB	T1CC2H	Timer 1 channel 2 capture or compare value high (additional XREG mapping of SFR register)
0x62AC	T1CC3L	Timer 1 channel 3 capture or compare value low
0x62AD	T1CC3H	Timer 1 channel 3 capture or compare value high
0x62AE	T1CC4L	Timer 1 channel 4 capture or compare value low
0x62AF	T1CC4H	Timer 1 channel 4 capture or compare value high
0x62B0	STCC	Sleep Timer capture control
0x62B1	STCS	Sleep Timer capture status
0x62B2	STCV0	Sleep Timer capture value byte 0
0x62B3	STCV1	Sleep Timer capture value byte 1
0x62B4	STCV2	Sleep Timer capture value byte 2
0x62C0	OPAMPC	Operational amplifier control
0x62C1	OPAMPS	Operational amplifier status
0x62D0	CMPCTL	Analog comparator control and status

## 2.2.4 XDATA Memory Access

The **MPAGE** register is used during instructions `MOVX A,@Ri` and `MOVX @Ri,A`. **MPAGE** gives the 8 most-significant address bits, whereas the register **Ri** gives the 8 least-significant bits.

In some 8051 implementations, this type of XDATA access is performed using **P2** to give the most-significant address bits. Existing software may therefore have to be adapted to make use of **MPAGE** instead of **P2**.

### MPAGE (0x93) – Memory Page Select

Bit	Name	Reset	R/W	Description
7:0	MPAGE[7:0]	0x00	R/W	Memory page, high-order bits of address in MOVX instruction

## 2.2.5 Memory Arbiter

The memory arbiter handles CPU and DMA access to all physical memory except the CPU internal registers. When an access conflict between the CPU and DMA occurs, the memory arbiter stalls one of the bus masters so that the conflict is resolved.

The control registers **MEMCTR** and **FMAP** are used to control various aspects of the memory subsystem. The **MEMCTR** and **FMAP** registers are described as follows.

**MEMCTR.XMAP** must be set to enable program execution from RAM.

The flash-bank map register, **FMAP**, controls mapping of physical 32-KB code banks to the program address region 0x8000–0xFFFF in CODE memory space.

**MEMCTR (0xC7) – Memory Arbiter Control**

Bit	Name	Reset	R/W	Description
7:4	—	0000	R0	Reserved
3	XMAP	0	R/W	XDATA map to code. When this bit is set, the SRAM XDATA region, from 0x0000 through (SRAM_SIZE – 1), is mapped into the CODE region from 0x8000 through (0x8000 + SRAM_SIZE – 1). This enables execution of program code from RAM. 0: SRAM map into CODE feature disabled 1: SRAM map into CODE feature enabled
2:0	XBANK[2:0]	000	R/W	XDATA bank select. Controls which code bank of the physical flash memory is mapped into the XDATA region (0x8000–0xFFFF). When set to 0, the root bank is mapped in. Valid settings depend on the flash size for the device. Writing an invalid setting is ignored, that is, no update to XBANK[2:0] is performed. 32-KB version: 0 only (that is, the root bank is always mapped in.) 64-KB version: 0–1 96-KB version: 0–2 128-KB version: 0–3 256-KB version: 0–7

**FMAP (0x9F) – Flash Bank Map**

Bit	Name	Reset	R/W	Description
7:3	—	0000 0	R0	Reserved
2:0	MAP[2:0]	001	R/W	Flash bank map. Controls which bank is mapped into the bank area of the CODE memory space (0x8000–0xFFFF). When set to 0, the root bank is mapped in. Valid settings depend on the flash size for the device. Writing an invalid setting is ignored, that is, no update to MAP[2:0] is performed. 32-KB version: No value can be written. Bank area is only used for running program code from SRAM. See MEMCTR.XMAP. 64-KB version: 0–1 96-KB version: 0–2 128-KB version: 0–3 256-KB version: 0–7

## 2.3 CPU Registers

This section describes the internal registers found in the CPU.

### 2.3.1 Data Pointers

Two data pointers, DPTR0 and DPTR1, exist to accelerate the movement of data blocks to and from memory. The data pointers are generally used to access CODE or XDATA space. For example:

```
MOVC A, @A+DPTR
MOV A, @DPTR.
```

The data pointer select bit, bit 0 in the data pointer select register DPS, chooses which data pointer is the active one during execution of an instruction that uses the data pointer, for example, in one of the preceding instructions.

The data pointers are two bytes wide, consisting of the following SFRs:

- DPTR0–DPH0:DPL0
- DPTR1–DPH1:DPL1

**DPH0 (0x83) – Data Pointer-0 High Byte**

Bit	Name	Reset	R/W	Description
7:0	DPH0[7:0]	0x00	R/W	Data pointer-0, high byte

**DPL0 (0x82) – Data Pointer-0 Low Byte**

Bit	Name	Reset	R/W	Description
7:0	DPL0[7:0]	0x00	R/W	Data pointer-0, low byte

**DPH1 (0x85) – Data Pointer-1 High Byte**

Bit	Name	Reset	R/W	Description
7:0	DPH1[7:0]	0x00	R/W	Data pointer-1, high byte

**DPL1 (0x84) – Data Pointer-1 Low Byte**

Bit	Name	Reset	R/W	Description
7:0	DPL1[7:0]	0x00	R/W	Data pointer-1, low byte

**DPS (0x92) – Data-Pointer Select**

Bit	Name	Reset	R/W	Description
7:1	–	0000 000	R0	Reserved
0	DPS	0	R/W	Data pointer select. Selects active data pointer. 0: DPTR0 1: DPTR1

### 2.3.2 Registers R0–R7

There are four register banks (not to be confused with CODE memory space banks that only apply to flash memory organization) of eight registers each. These register banks are mapped in the DATA memory space at addresses 0x00–0x07, 0x08–0x0F, 0x10–0x17, and 0x18–0x1F. Each register bank contains the eight 8-bit registers R0–R7. The register bank to be used is selected through the program status word PSW.RS[1:0]. Register bank 0 uses flip-flops internally for storing the values (SRAM is bypassed or unused), whereas banks 1–3 use SRAM for storage. This is done to save power. Typically, the current consumption goes down by approximately 200  $\mu$ A by using register bank 0 instead of register banks 1–3.

### 2.3.3 Program Status Word

The program status word (PSW) contains several bits that show the current state of the CPU. The PSW is accessible as an SFR, and it is bit-addressable. The PSW is shown as follows and contains the carry flag, auxiliary carry flag for BCD operations, register-select bits, overflow flag, and parity flag. Two bits in the PSW are uncommitted and can be used as user-defined status flags.

**PSW (0xD0) – Program Status Word**

Bit	Name	Reset	R/W	Description
7	CY	0	R/W	Carry flag. Set to 1 when the last arithmetic operation resulted in a carry (during addition) or borrow (during subtraction); otherwise, cleared to 0 by all arithmetic operations.
6	AC	0	R/W	Auxiliary carry flag for BCD operations. Set to 1 when the last arithmetic operation resulted in a carry into (during addition) or borrow from (during subtraction) the high-order nibble, otherwise cleared to 0 by all arithmetic operations.
5	F0	0	R/W	User-defined, bit-addressable
4:3	RS[1:0]	00	R/W	Register bank select bits. Selects which set of R7–R0 registers to use from four possible banks in DATA space. 00: Register bank 0, 0x00–0x07 01: Register bank 1, 0x08–0x0F 10: Register bank 2, 0x10–0x17 11: Register bank 3, 0x18–0x1F
2	OV	0	R/W	Overflow flag, set by arithmetic operations. Set to 1 when the last arithmetic operation is a carry (addition), borrow (subtraction), or overflow (multiply or divide). Otherwise, the bit is cleared to 0 by all arithmetic operations.
1	F1	0	R/W	User-defined, bit-addressable
0	P	0	R/W	Parity flag, parity of accumulator set by hardware to 1 if it contains an odd number of 1s; otherwise it is cleared to 0.

### 2.3.4 Accumulator

ACC is the accumulator. This is the source and destination of most arithmetic instructions, data transfers, and other instructions. The mnemonic for the accumulator (in instructions involving the accumulator) is A instead of ACC.

#### ACC (0xE0) – Accumulator

Bit	Name	Reset	R/W	Description
7:0	ACC[7:0]	0x00	R/W	Accumulator

### 2.3.5 B Register

The B register is used as the second 8-bit argument during execution of multiply and divide instructions. When not used for these purposes, it may be used as a scratchpad register to hold temporary data.

#### B (0xF0) – B Register

Bit	Name	Reset	R/W	Description
7:0	B[7:0]	0x00	R/W	B register. Used in MUL and DIV instructions

### 2.3.6 Stack Pointer

The stack resides in DATA memory space and grows upwards. The `PUSH` instruction first increments the stack pointer (`SP`) and then copies the byte into the stack. The `SP` is initialized to 0x07 after a reset, and it is incremented once to start from location 0x08, which is the first register (`R0`) of the second register bank. Thus, in order to use more than one register bank, the `SP` should be initialized to a different location not used for data storage.

#### SP (0x81) – Stack Pointer

Bit	Name	Reset	R/W	Description
7:0	SP[7:0]	0x07	R/W	Stack pointer

## 2.4 Instruction Set Summary

The 8051 instruction set is summarized in [Table 2-3](#). All mnemonics copyrighted © Intel Corporation, 1980.

The following conventions are used in the instruction set summary:

- `Rn` – Register `R7–R0` of the currently selected register bank
- `Direct` – 8-bit internal data-location address. This can be DATA area (0x00–0x7F) or SFR area (0x80–0xFF).
- `@Ri` – 8-bit internal data location, DATA area (0x00–0xFF) addressed indirectly through register `R1` or `R0`
- `#data` – 8-bit constant included in instruction
- `#data16` – 16-bit constant included in instruction
- `addr16` – 16-bit destination address. Used by `LCALL` and `LJMP`. A branch can be anywhere within the 64 KB CODE memory space.
- `addr11` – 11-bit destination address. Used by `ACALL` and `AJMP`. The branch is within the same 2 KB page of program memory as the first byte of the following instruction.
- `rel` – Signed (2s-complement) 8-bit offset byte. Used by `SJMP` and all conditional jumps. Range is –128 to 127 bytes relative to first byte of the following instruction.
- `bit` – Direct addressed bit in DATA area or SFR

The instructions that affect CPU flag settings located in PSW are listed in [Table 2-4](#). Note that operations on the PSW register or bits in PSW also affect the flag settings. Also note that the cycle count for many instructions assumes single-cycle access to the memory element being accessed, that is, the best-case situation. This is not always the case. Reads from flash may take 1–3 cycles, for example.

**Table 2-3. Instruction Set Summary**

Mnemonic	Description	Hex Opcode	Bytes	Cycles
<b>ARITHMETIC OPERATIONS</b>				
ADD A,Rn	Add register to accumulator	28–2F	1	1
ADD A,direct	Add direct byte to accumulator	25	2	2
ADD A,@Ri	Add indirect RAM to accumulator	26–27	1	2
ADD A,#data	Add immediate data to accumulator	24	2	2
ADDC A,Rn	Add register to accumulator with carry flag	38–3F	1	1
ADDC A,direct	Add direct byte to A with carry flag	35	2	2
ADDC A,@Ri	Add indirect RAM to A with carry flag	36–37	1	2
ADDC A,#data	Add immediate data to A with carry flag	34	2	2
SUBB A,Rn	Subtract register from A with borrow	98–9F	1	1
SUBB A,direct	Subtract direct byte from A with borrow	95	2	2
SUBB A,@Ri	Subtract indirect RAM from A with borrow	96–97	1	2
SUBB A,#data	Subtract immediate data from A with borrow	94	2	2
INC A	Increment accumulator	04	1	1
INC Rn	Increment register	08–0F	1	2
INC direct	Increment direct byte	05	2	3
INC @Ri	Increment indirect RAM	06–07	1	3
INC DPTR	Increment data pointer	A3	1	1
DEC A	Decrement accumulator	14	1	1
DEC Rn	Decrement register	18–1F	1	2
DEC direct	Decrement direct byte	15	2	3
DEC @Ri	Decrement indirect RAM	16–17	1	3
MUL AB	Multiply A and B	A4	1	5
DIV A	Divide A by B	84	1	5
DA A	Decimal adjust accumulator	D4	1	1
<b>LOGICAL OPERATIONS</b>				
ANL A,Rn	AND register to accumulator	58–5F	1	1
ANL A,direct	AND direct byte to accumulator	55	2	2
ANL A,@Ri	AND indirect RAM to accumulator	56–57	1	2
ANL A,#data	AND immediate data to accumulator	54	2	2
ANL direct,A	AND accumulator to direct byte	52	2	3
ANL direct,#data	AND immediate data to direct byte	53	3	4
ORL A,Rn	OR register to accumulator	48–4F	1	1
ORL A,direct	OR direct byte to accumulator	45	2	2
ORL A,@Ri	OR indirect RAM to accumulator	46–47	1	2
ORL A,#data	OR immediate data to accumulator	44	2	2
ORL direct,A	OR accumulator to direct byte	42	2	3
ORL direct,#data	OR immediate data to direct byte	43	3	4
XRL A,Rn	Exclusive OR register to accumulator	68–6F	1	1
XRL A,direct	Exclusive OR direct byte to accumulator	65	2	2
XRL A,@Ri	Exclusive OR indirect RAM to accumulator	66–67	1	2
XRL A,#data	Exclusive OR immediate data to accumulator	64	2	2
XRL direct,A	Exclusive OR accumulator to direct byte	62	2	3

**Table 2-3. Instruction Set Summary (continued)**

Mnemonic	Description	Hex Opcode	Bytes	Cycles
XRL direct,#data	Exclusive OR immediate data to direct byte	63	3	4
CLR A	Clear accumulator	E4	1	1
CPL A	Complement accumulator	F4	1	1
RL A	Rotate accumulator left	23	1	1
RLC A	Rotate accumulator left through carry	33	1	1
RR A	Rotate accumulator right	03	1	1
RRC A	Rotate accumulator right through carry	13	1	1
SWAP A	Swap nibbles within the accumulator	C4	1	1
<b>DATA TRANSFERS</b>				
MOV A,Rn	Move register to accumulator	E8–EF	1	1
MOV A,direct	Move direct byte to accumulator	E5	2	2
MOV A,@Ri	Move indirect RAM to accumulator	E6–E7	1	2
MOV A,#data	Move immediate data to accumulator	74	2	2
MOV Rn,A	Move accumulator to register	F8–FF	1	2
MOV Rn,direct	Move direct byte to register	A8–AF	2	4
MOV Rn,#data	Move immediate data to register	78–7F	2	2
MOV direct,A	Move accumulator to direct byte	F5	2	3
MOV direct,Rn	Move register to direct byte	88–8F	2	3
MOV direct1,direct2	Move direct byte to direct byte	85	3	4
MOV direct,@Ri	Move indirect RAM to direct byte	86–87	2	4
MOV direct,#data	Move immediate data to direct byte	75	3	3
MOV @Ri,A	Move accumulator to indirect RAM	F6–F7	1	3
MOV @Ri,direct	Move direct byte to indirect RAM	A6–A7	2	5
MOV @Ri,#data	Move immediate data to indirect RAM	76–77	2	3
MOV DPTR,#data16	Load data pointer with a 16-bit constant	90	3	3
MOVC A,@A+DPTR	Move code byte relative to DPTR to accumulator	93	1	3
MOVC A,@A+PC	Move code byte relative to PC to accumulator	83	1	3
MOVX A,@Ri	Move external RAM (8-bit address) to A	E2–E3	1	3
MOVX A,@DPTR	Move external RAM (16-bit address) to A	E0	1	3
MOVX @Ri,A	Move A to external RAM (8-bit address)	F2–F3	1	4
MOVX @DPTR,A	Move A to external RAM (16-bit address)	F0	1	4
PUSH direct	Push direct byte onto stack	C0	2	4
POP direct	Pop direct byte from stack	D0	2	3
XCH A,Rn	Exchange register with accumulator	C8–CF	1	2
XCH A,direct	Exchange direct byte with accumulator	C5	2	3
XCH A,@Ri	Exchange indirect RAM with accumulator	C6–C7	1	3
XCHD A,@Ri	Exchange low-order nibble indirect. RAM with A	D6–D7	1	3
<b>PROGRAM BRANCHING</b>				
ACALL addr11	Absolute subroutine call	xxx11	2	6
LCALL addr16	Long subroutine call	12	3	6
RET	Return from subroutine	22	1	4
RETI	Return from interrupt	32	1	4
AJMP addr11	Absolute jump	xxx01	2	3
LJMP addr16	Long jump	02	3	4
SJMP rel	Short jump (relative address)	80	2	3
JMP @A+DPTR	Jump indirect relative to the DPTR	73	1	2
JZ rel	Jump if accumulator is zero	60	2	3

**Table 2-3. Instruction Set Summary (continued)**

Mnemonic	Description	Hex Opcode	Bytes	Cycles
JNZ rel	Jump if accumulator is not zero	70	2	3
JC rel	Jump if carry flag is set	40	2	3
JNC	Jump if carry flag is not set	50	2	3
JB bit,rel	Jump if direct bit is set	20	3	4
JNB bit,rel	Jump if direct bit is not set	30	3	4
JBC bit,direct rel	Jump if direct bit is set and clear bit	10	3	4
CJNE A,direct rel	Compare direct byte to A and jump if not equal	B5	3	4
CJNE A,#data rel	Compare immediate to A and jump if not equal	B4	3	4
CJNE Rn,#data rel	Compare immediate to reg. and jump if not equal	B8–BF	3	4
CJNE @Ri,#data rel	Compare immediate to indirect and jump if not equal	B6–B7	3	4
DJNZ Rn,rel	Decrement register and jump if not zero	D8–DF	1	3
DJNZ direct,rel	Decrement direct byte and jump if not zero	D5	3	4
NOP	No operation	00	1	1
<b>Boolean VARIABLE OPERATIONS</b>				
CLR C	Clear carry flag	C3	1	1
CLR bit	Clear direct bit	C2	2	3
SETB C	Set carry flag	D3	1	1
SETB bit	Set direct bit	D2	2	3
CPL C	Complement carry flag	B3	1	1
CPL bit	Complement direct bit	B2	2	3
ANL C,bit	AND direct bit to carry flag	82	2	2
ANL C,/bit	AND complement of direct bit to carry	B0	2	2
ORL C,bit	OR direct bit to carry flag	72	2	2
ORL C,/bit	OR complement of direct bit to carry	A0	2	2
MOV C,bit	Move direct bit to carry flag	A2	2	2
MOV bit,C	Move carry flag to direct bit	92	2	3

**Table 2-4. Instructions That Affect Flag Settings<sup>(1)</sup>**

Instruction	CY	OV	AC
ADD	x	x	x
ADDC	x	x	x
SUBB	x	x	x
MUL	0	x	–
DIV	0	x	–
DA	x	–	–
RRC	x	–	–
RLC	x	–	–
SETB C	1	–	–
CLR C	x	–	–
CPLC	x	–	–
ANL C,bit	x	–	–
ANL C,/bit	x	–	–
ORL C,bit	x	–	–
ORL C,/bit	x	–	–
MOV C,bit	x	–	–
CJNE	x	–	–

<sup>(1)</sup> 0 = set to 0, 1 = set to 1, x = set to 0 or 1, – = not affected

## 2.5 Interrupts

The CPU has 18 interrupt sources. Each source has its own request flag located in a set of interrupt-flag SFR registers. Each interrupt requested by the corresponding flag can be individually enabled or disabled. The definitions of the interrupt sources and the interrupt vectors are given in [Table 2-5](#).

The interrupts are grouped into a set of priority-level groups with selectable priority levels.

The interrupt-enable registers are described in [Section 2.5.1](#) and the interrupt priority settings are described in [Section 2.5.3](#).






## 2.5.1 Interrupt Masking

Each interrupt can be individually enabled or disabled by the interrupt-enable bits in the interrupt-enable SFRs `IEN0`, `IEN1`, and `IEN2`. The CPU interrupt-enable SFRs are described as follows and summarized in [Table 2-5](#).

Note that some peripherals have several events that can generate the interrupt request associated with that peripheral. This applies to Port 0, Port 1, Port 2, Timer 1, Timer 2, Timer 3, Timer 4, DMA controller, and Radio. These peripherals have interrupt mask bits for each internal interrupt source in the corresponding SFR or XREG register.

In order to enable any of the interrupts, the following steps must be taken:

1. Clear interrupt flags.
2. Set individual interrupt-enable bit in the peripherals SFR register, if any.
3. Set the corresponding individual interrupt-enable bit in the `IEN0`, `IEN1`, or `IEN2` register to 1.
4. Enable global interrupt by setting the EA bit in `IEN0` to 1.
5. Begin the interrupt service routine at the corresponding vector address of that interrupt. See [Table 2-5](#) for addresses.

[Figure 2-4](#) gives a complete overview of all interrupt sources and associated control and state registers. Shaded boxes in [Figure 2-4](#) are interrupt flags that are automatically cleared by hardware when the interrupt service routine is called.  indicates a one-shot, either due to the level source or due to edge shaping. Interrupts missing this are to be treated as level-triggered (apply to ports P0, P1, and P2). The switch boxes are shown in the default state, and  or  indicates rising- or falling-edge detection, that is, at what time instance the interrupt is generated. As a general rule for pulsed or edge-shaped interrupt sources, one should clear CPU interrupt flag registers prior to clearing the source flag bit, if available, for flags that are not automatically cleared. For level sources, one must clear the source prior to clearing the CPU flag.

Note that when clearing source interrupt flags in a register that contains several flags, interrupts may be lost if a read-modify-write operation is done (even in a single assembly instruction), as it also clears interrupt flags that became active between the read and write operation. The source interrupt flags (with the exception of the USB controller interrupt flags) have the access mode R/W0. This means that writing 1 to a bit has no effect, so 1 should be written to an interrupt flag that is not to be cleared. For instance, to clear the `TIMER2_OVF_PERF` bit (bit 3) of `T2IRQF` in C code, one should do:

```
T2IRQF = ~(1 << 3);
and not:
T2IRQF &= ~(1 << 3); // wrong!
```

**Table 2-5. Interrupts Overview**

Interrupt Number	Description	Interrupt Name	Interrupt Vector	Interrupt Mask, CPU	Interrupt Flag, CPU
0	RF core-error situation	RFERR	0x03	<code>IEN0.RFERRIE</code>	<code>TCON.RFERRIF</code> <sup>(1)</sup>
1	ADC end of conversion	ADC	0x0B	<code>IEN0.ADCIE</code>	<code>TCON.ADCIF</code> <sup>(1)</sup>
2	USART 0 RX complete	URX0	0x13	<code>IEN0.URX0IE</code>	<code>TCON.URX0IF</code> <sup>(1)</sup>
3	USART 1 RX complete	URX1	0x1B	<code>IEN0.URX1IE</code>	<code>TCON.URX1IF</code> <sup>(1)</sup>
4	AES encryption or decryption complete	ENC	0x23	<code>IEN0.ENCIE</code>	<code>SOCON.ENCIF</code>
5	Sleep Timer compare	ST	0x2B	<code>IEN0.STIE</code>	<code>IRCON.STIF</code>
6	Port-2 inputs, USB, or I <sup>2</sup> C	P2INT	0x33	<code>IEN2.P2IE</code>	<code>IRCON2.P2IF</code> <sup>(2)</sup>
7	USART 0 TX complete	UTX0	0x3B	<code>IEN2.UTX0IE</code>	<code>IRCON2.UTX0IF</code>
8	DMA transfer complete	DMA	0x43	<code>IEN1.DMAIE</code>	<code>IRCON.DMAIF</code>
9	Timer 1 (16-bit) capture, compare, overflow	T1	0x4B	<code>IEN1.T1IE</code>	<code>IRCON.T1IF</code> <sup>(1) (2)</sup>
10	Timer 2	T2	0x53	<code>IEN1.T2IE</code>	<code>IRCON.T2IF</code> <sup>(1) (2)</sup>

<sup>(1)</sup> Hardware-cleared when interrupt service routine is called

<sup>(2)</sup> Additional IRQ mask and IRQ flag bits exist.

**Table 2-5. Interrupts Overview (continued)**

Interrupt Number	Description	Interrupt Name	Interrupt Vector	Interrupt Mask, CPU	Interrupt Flag, CPU
11	Timer 3 (8-bit) capture, compare, overflow	T3	0x5B	IEN1.T3IE	IRCON.T3IF <sup>(1) (2)</sup>
12	Timer 4 (8-bit) capture, compare, overflow	T4	0x63	IEN1.T4IE	IRCON.T4IF <sup>(1) (2)</sup>
13	Port 0 inputs	P0INT	0x6B	IEN1.P0IE	IRCON.P0IF <sup>(2)</sup>
14	USART 1 TX complete	UTX1	0x73	IEN2.UTX1IE	IRCON2.UTX1IF
15	Port 1 inputs	P1INT	0x7B	IEN2.P1IE	IRCON2.P1IF <sup>(2)</sup>
16	RF general interrupts	RF	0x83	IEN2.RFIE	S1CON.RFIF <sup>(2)</sup>
17	Watchdog overflow in timer mode	WDT	0x8B	IEN2.WDTIE	IRCON2.WDTIF

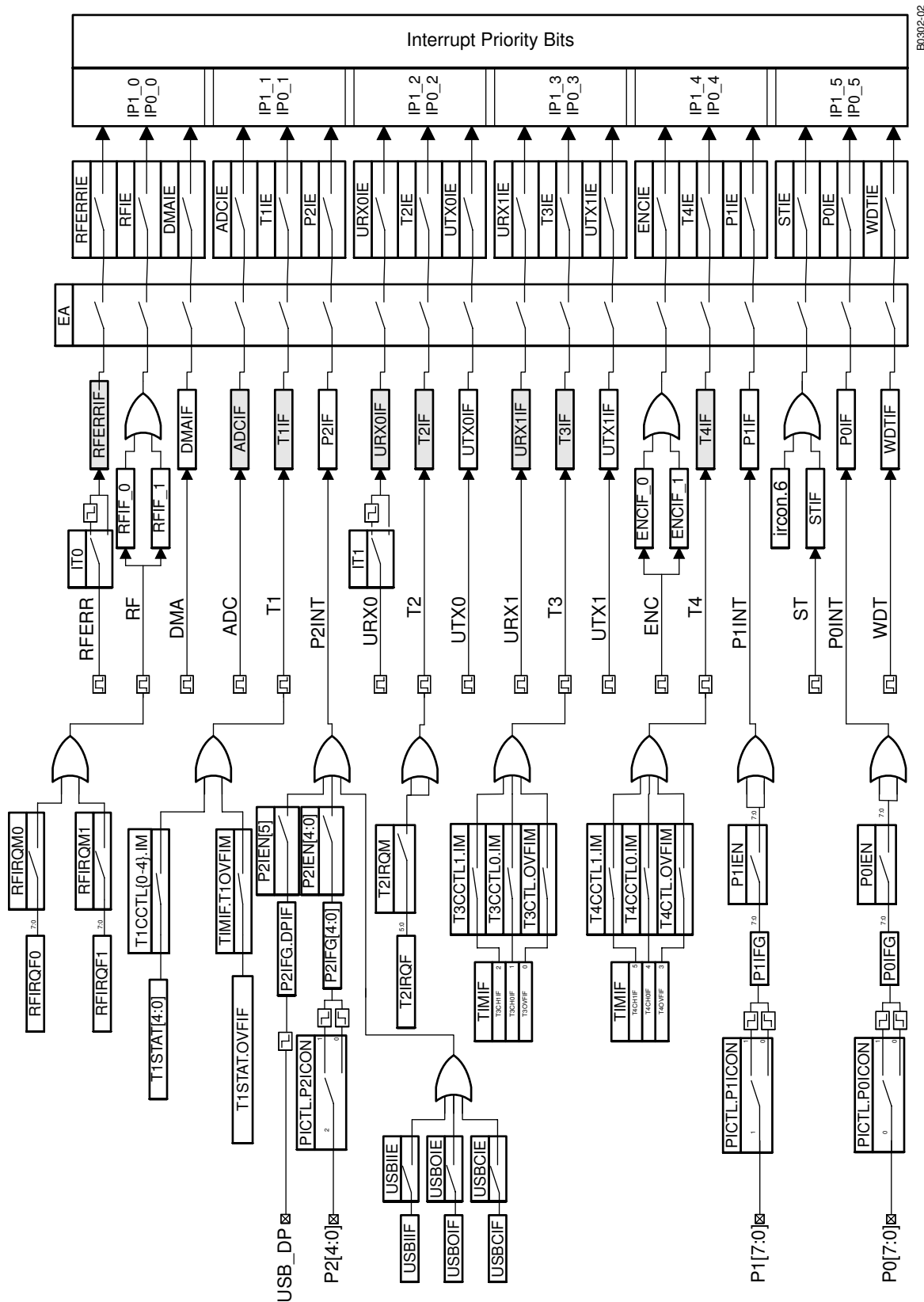


Figure 2-4. Interrupt Overview

**IEN0 (0xA8) – Interrupt Enable 0**

Bit	Name	Reset	R/W	Description
7	EA	0	R/W	Disables all interrupts. 0: No interrupt is acknowledged. 1: Each interrupt source is individually enabled or disabled by setting its corresponding enable bit.
6	–	0	R0	Reserved. Read as 0
5	STIE	0	R/W	Sleep Timer interrupt enable 0: Interrupt disabled 1: Interrupt enabled
4	ENCIE	0	R/W	AES encryption and decryption interrupt enable 0: Interrupt disabled 1: Interrupt enabled
3	URX1IE	0	R/W	USART 1 RX interrupt enable 0: Interrupt disabled 1: Interrupt enabled
2	URX0IE	0	R/W	USART0 RX interrupt enable 0: Interrupt disabled 1: Interrupt enabled
1	ADCIE	0	R/W	ADC interrupt enable 0: Interrupt disabled 1: Interrupt enabled
0	RFERRIE	0	R/W	RF core error interrupt enable 0: Interrupt disabled 1: Interrupt enabled

**IEN1 (0xB8) – Interrupt Enable 1**

Bit	Name	Reset	R/W	Description
7:6	–	00	R0	Reserved. Read as 0
5	P0IE	0	R/W	Port 0 interrupt enable 0: Interrupt disabled 1: Interrupt enabled
4	T4IE	0	R/W	Timer 4 interrupt enable 0: Interrupt disabled 1: Interrupt enabled
3	T3IE	0	R/W	Timer 3 interrupt enable 0: Interrupt disabled 1: Interrupt enabled
2	T2IE	0	R/W	Timer 2 interrupt enable 0: Interrupt disabled 1: Interrupt enabled
1	T1IE	0	R/W	Timer 1 interrupt enable 0: Interrupt disabled 1: Interrupt enabled
0	DMAIE	0	R/W	DMA transfer interrupt enable 0: Interrupt disabled 1: Interrupt enabled

**IEN2 (0x9A) – Interrupt Enable 2**

Bit	Name	Reset	R/W	Description
7:6	–	00	R0	Reserved. Read as 0
5	WDTIE	0	R/W	Watchdog Timer interrupt enable 0: Interrupt disabled 1: Interrupt enabled
4	P1IE	0	R/W	Port 1 interrupt enable 0: Interrupt disabled 1: Interrupt enabled
3	UTX1IE	0	R/W	USART 1 TX interrupt enable 0: Interrupt disabled 1: Interrupt enabled
2	UTX0IE	0	R/W	USART 0 TX interrupt enable 0: Interrupt disabled 1: Interrupt enabled
1	P2IE	0	R/W	Port 2 and USB interrupt enable 0: Interrupt disabled 1: Interrupt enabled
0	RFIE	0	R/W	RF general interrupt enable 0: Interrupt disabled 1: Interrupt enabled

### 2.5.2 Interrupt Processing

When an interrupt occurs, the CPU vectors to the interrupt-vector address as shown in [Table 2-5](#). Once an interrupt service has begun, it can be interrupted only by a higher-priority interrupt. The interrupt service is terminated by an `RETI` (return-from-interrupt instruction). When an `RETI` is performed, the CPU returns to the instruction that would have been next when the interrupt occurred.

When the interrupt condition occurs, the CPU also indicates this by setting an interrupt flag bit in the interrupt flag registers. This bit is set regardless of whether the interrupt is enabled or disabled. If the interrupt is enabled when an interrupt flag is set, then on the next instruction cycle, the interrupt is acknowledged by hardware, forcing an `LCALL` to the appropriate vector address.

Interrupt response requires a varying amount of time, depending on the state of the CPU when the interrupt occurs. If the CPU is performing an interrupt service with equal or greater priority, the new interrupt is pending until it becomes the interrupt with highest priority. In other cases, the response time depends on current instruction. The fastest possible response to an interrupt is seven machine cycles. This includes one machine cycle for detecting the interrupt and six cycles to perform the `LCALL`.

---

**NOTE:** If an interrupt is disabled and the interrupt flag is polled, the 8051 assembly instruction `JBC` must not be used to poll the interrupt flag and clear it when set. If the `JBC` instruction is used, the interrupt flag may be re-asserted immediately.

---



---

**NOTE:** If the assembly instruction `XCH A, IEN0` is used to clear the global interrupt enable flag `EA`, the CPU may enter the interrupt routine on the cycle following this instruction. If that happens, the interrupt routine is executed with `EA` set to 0, which may delay the service of higher-priority interrupts.

---

**TCON (0x88) – Interrupt Flags**

Bit	Name	Reset	R/W	Description
7	URX1IF	0	R/W H0	USART 1 RX interrupt flag. Set to 1 when USART 1 RX interrupt occurs and cleared when CPU vectors to the interrupt service routine. 0: Interrupt not pending 1: Interrupt pending
6	–	0	R/W	Reserved
5	ADCIF	0	R/W H0	ADC interrupt flag. Set to 1 when ADC interrupt occurs and cleared when CPU vectors to the interrupt service routine. 0: Interrupt not pending 1: Interrupt pending
4	–	0	R/W	Reserved
3	URX0IF	0	R/W H0	USART 0 RX interrupt flag. Set to 1 when USART 0 interrupt occurs and cleared when CPU vectors to the interrupt service routine. 0: Interrupt not pending 1: Interrupt pending
2	IT1	1	R/W	Reserved. Must always be set to 1. Setting a zero enables low-level interrupt detection, which is almost always the case (one-shot when interrupt request is initiated).
1	RFERRIF	0	R/W H0	RF core error interrupt flag. Set to 1 when RFERR interrupt occurs and cleared when CPU vectors to the interrupt service routine. 0: Interrupt not pending 1: Interrupt pending
0	IT0	1	R/W	Reserved. Must always be set to 1. Setting a zero enables low-level interrupt detection, which is almost always the case (one-shot when interrupt request is initiated).

**S0CON (0x98) – Interrupt Flags 2**

Bit	Name	Reset	R/W	Description
7:2	–	0000 00	R/W	Reserved
1	ENCIF_1	0	R/W	AES interrupt. ENC has two interrupt flags, ENCIF_1 and ENCIF_0. Setting one of these flags requests interrupt service. Both flags are set when the AES coprocessor requests the interrupt. 0: Interrupt not pending 1: Interrupt pending
0	ENCIF_0	0	R/W	AES interrupt. ENC has two interrupt flags, ENCIF_1 and ENCIF_0. Setting one of these flags requests interrupt service. Both flags are set when the AES coprocessor requests the interrupt. 0: Interrupt not pending 1: Interrupt pending

**S1CON (0x9B) – Interrupt Flags 3**

Bit	Name	Reset	R/W	Description
7:2	–	0000 00	R/W	Reserved
1	RFIF_1	0	R/W	RF general interrupt. RF has two interrupt flags, RFIF_1 and RFIF_0. Setting one of these flags requests interrupt service. Both flags are set when the radio requests the interrupt. 0: Interrupt not pending 1: Interrupt pending
0	RFIF_0	0	R/W	RF general interrupt. RF has two interrupt flags, RFIF_1 and RFIF_0. Setting one of these flags requests interrupt service. Both flags are set when the radio requests the interrupt. 0: Interrupt not pending 1: Interrupt pending

**IRCON (0xC0) – Interrupt Flags 4**

Bit	Name	Reset	R/W	Description
7	STIF	0	R/W	Sleep Timer interrupt flag 0: Interrupt not pending 1: Interrupt pending
6	–	0	R/W	Must be written 0. Writing a 1 always enables the interrupt source.
5	P0IF	0	R/W	Port 0 interrupt flag 0: Interrupt not pending 1: Interrupt pending
4	T4IF	0	R/W H0	Timer 4 interrupt flag. Set to 1 when Timer 4 interrupt occurs and cleared when CPU vectors to the interrupt service routine. 0: Interrupt not pending 1: Interrupt pending
3	T3IF	0	R/W H0	Timer 3 interrupt flag. Set to 1 when Timer 3 interrupt occurs and cleared when CPU vectors to the interrupt service routine. 0: Interrupt not pending 1: Interrupt pending
2	T2IF	0	R/W H0	Timer 2 interrupt flag. Set to 1 when Timer 2 interrupt occurs and cleared when CPU vectors to the interrupt service routine. 0: Interrupt not pending 1: Interrupt pending
1	T1IF	0	R/W H0	Timer 1 interrupt flag. Set to 1 when Timer 1 interrupt occurs and cleared when CPU vectors to the interrupt service routine. 0: Interrupt not pending 1: Interrupt pending
0	DMAIF	0	R/W	DMA-complete interrupt flag 0: Interrupt not pending 1: Interrupt pending

**IRCON2 (0xE8) – Interrupt Flags 5**

Bit	Name	Reset	R/W	Description
7:5	–	000	R/W	Reserved
4	WDTIF	0	R/W	Watchdog Timer interrupt flag 0: Interrupt not pending 1: Interrupt pending
3	P1IF	0	R/W	Port 1 interrupt flag 0: Interrupt not pending 1: Interrupt pending
2	UTX1IF	0	R/W	USART 1 TX interrupt flag 0: Interrupt not pending 1: Interrupt pending
1	UTX0IF	0	R/W	USART 0 TX interrupt flag 0: Interrupt not pending 1: Interrupt pending
0	P2IF	0	R/W	Port 2 interrupt flag 0: Interrupt not pending 1: Interrupt pending

**2.5.3 Interrupt Priority**

The interrupts are grouped into six interrupt priority groups, and the priority for each group is set by registers IP0 and IP1. In order to assign a higher priority to an interrupt, that is, to its interrupt group, the corresponding bits in IP0 and IP1 must be set as shown in [Table 2-6](#).

The interrupt priority groups with assigned interrupt sources are shown in [Table 2-7](#). Each group is assigned one of four priority levels. While an interrupt service request is in progress, it cannot be interrupted by a lower- or same-level interrupt.

In the case when interrupt requests of the same priority level are received simultaneously, the polling sequence shown in [Table 2-8](#) is used to resolve the priority of each request. Note that the polling sequence in [Figure 2-4](#) is the algorithm found in [Table 2-8](#), not that polling is among the IP bits as listed in the figure.

**IP1 (0xB9) – Interrupt Priority 1**

Bit	Name	Reset	R/W	Description
7:6	–	00	R/W	Reserved
5	IP1_IPG5	0	R/W	Interrupt group 5, priority control bit 1, see <a href="#">Table 2-7: Interrupt Priority Groups</a>
4	IP1_IPG4	0	R/W	Interrupt group 4, priority control bit 1, see <a href="#">Table 2-7: Interrupt Priority Groups</a>
3	IP1_IPG3	0	R/W	Interrupt group 3, priority control bit 1, see <a href="#">Table 2-7: Interrupt Priority Groups</a>
2	IP1_IPG2	0	R/W	Interrupt group 2, priority control bit 1, see <a href="#">Table 2-7: Interrupt Priority Groups</a>
1	IP1_IPG1	0	R/W	Interrupt group 1, priority control bit 1, see <a href="#">Table 2-7: Interrupt Priority Groups</a>
0	IP1_IPG0	0	R/W	Interrupt group 0, priority control bit 1, see <a href="#">Table 2-7: Interrupt Priority Groups</a>

**IP0 (0xA9) – Interrupt Priority 0**

Bit	Name	Reset	R/W	Description
7:6	–	00	R/W	Reserved
5	IP0_IPG5	0	R/W	Interrupt group 5, priority control bit 0, see <a href="#">Table 2-7: Interrupt Priority Groups</a>
4	IP0_IPG4	0	R/W	Interrupt group 4, priority control bit 0, see <a href="#">Table 2-7: Interrupt Priority Groups</a>
3	IP0_IPG3	0	R/W	Interrupt group 3, priority control bit 0, see <a href="#">Table 2-7: Interrupt Priority Groups</a>
2	IP0_IPG2	0	R/W	Interrupt group 2, priority control bit 0, see <a href="#">Table 2-7: Interrupt Priority Groups</a>
1	IP0_IPG1	0	R/W	Interrupt group 1, priority control bit 0, see <a href="#">Table 2-7: Interrupt Priority Groups</a>
0	IP0_IPG0	0	R/W	Interrupt group 0, priority control bit 0, see <a href="#">Table 2-7: Interrupt Priority Groups</a>

**Table 2-6. Priority Level Setting**

IP1_x	IP0_x	Priority Level
0	0	0 – lowest
0	1	1
1	0	2
1	1	3 – highest

**Table 2-7. Interrupt Priority Groups**

Group	Interrupts		
IPG0	RFERR	RF	DMA
IPG1	ADC	T1	P2INT
IPG2	URX0	T2	UTX0
IPG3	URX1	T3	UTX1
IPG4	ENC	T4	P1INT
IPG5	ST	P0INT	WDT



**Table 2-8. Interrupt Polling Sequence**

Interrupt Number	Interrupt Name	
0	RFERR	Polling sequence ↓
16	RF	
8	DMA	
1	ADC	
9	T1	
2	URX0	
10	T2	
3	URX1	
11	T3	
4	ENC	
12	T4	
5	ST	
13	P0INT	
6	P2INT	
7	UTX0	
14	UTX1	
15	P1INT	
17	WDT	

## Debug Interface

---



---

The two-wire debug interface allows programming of the on-chip flash, and it provides access to memory and register contents and debug features such as breakpoints, single-stepping, and register modification.

The debug interface uses I/O pins P2.1 and P2.2 as debug data and debug clock, respectively, during debug mode. These I/O pins can be used as general-purpose I/O only while the device is not in debug mode. Thus, the debug interface does not interfere with any peripheral I/O pins.

Topic	Page
<b>3.1 Debug Mode .....</b>	<b>51</b>
<b>3.2 Debug Communication .....</b>	<b>51</b>
<b>3.3 Debug Commands .....</b>	<b>53</b>
<b>3.4 Flash Programming .....</b>	<b>57</b>
<b>3.5 Debug Interface and Power Modes .....</b>	<b>57</b>
<b>3.6 Registers.....</b>	<b>59</b>

### 3.1 Debug Mode

Debug mode is entered by forcing two falling-edge transitions on pin P2.2 (debug clock) while the RESET\_N input is held low. When RESET\_N is set high, the device is in debug mode.

On entering debug mode, the CPU is in the halted state with the program counter reset to address 0x0000.

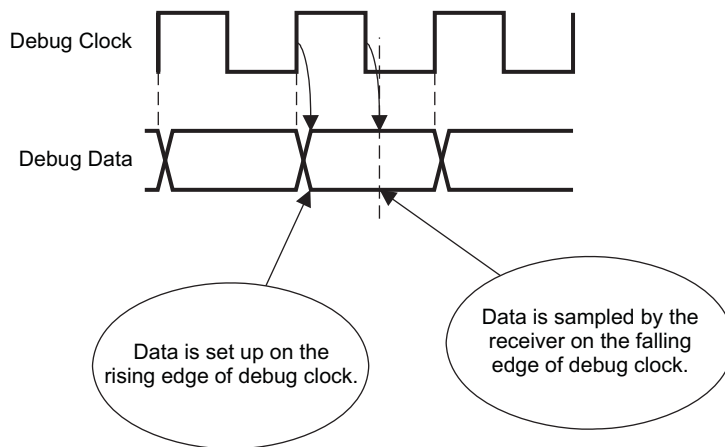
While in debug mode, pin P2.1 is the debug-data bidirectional pin, and P2.2 is the debug-clock input pin.

**NOTE:** Note that the debugger cannot be used with a divided system clock. When running the debugger, the value of CLKCONCMD.CLKSPD should be set to 000 when CLKCONCMD.OSC = 0 or to 001 when CLKCONCMD.OSC = 1.

### 3.2 Debug Communication

The debug interface uses a SPI-like two-wire interface consisting of the P2.1 (debug data) and P2.2 (debug clock) pins. Data is driven on the bidirectional debug-data pin at the positive edge of the debug clock, and data is sampled on the negative edge of this clock.

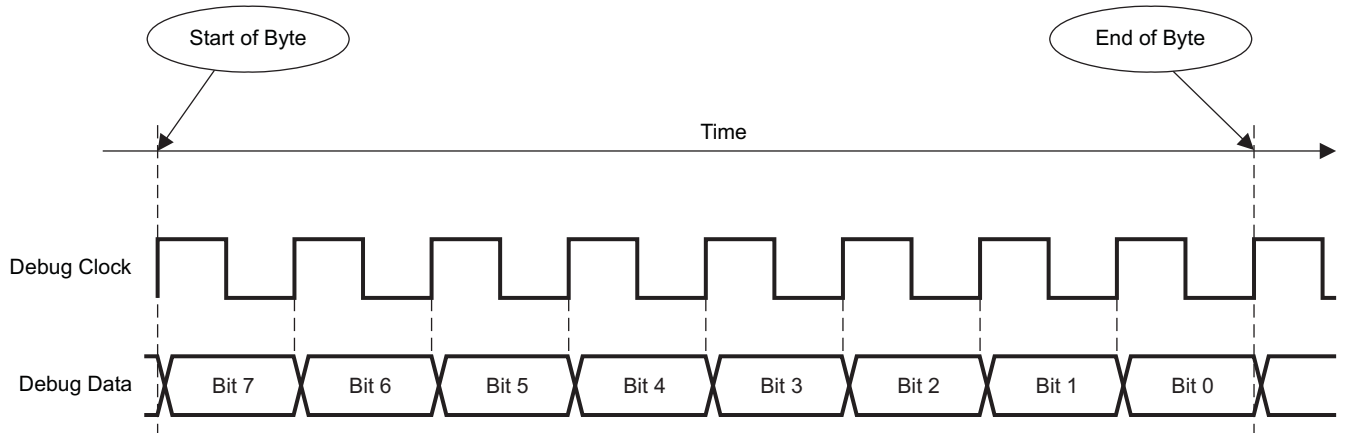
The direction of the debug-data pin depends on the command being issued. Data is driven on the positive edge of the debug clock and sampled on the negative edge. Figure 3-1 shows how data is sampled.



T0302-01

Figure 3-1. External Debug Interface Timing

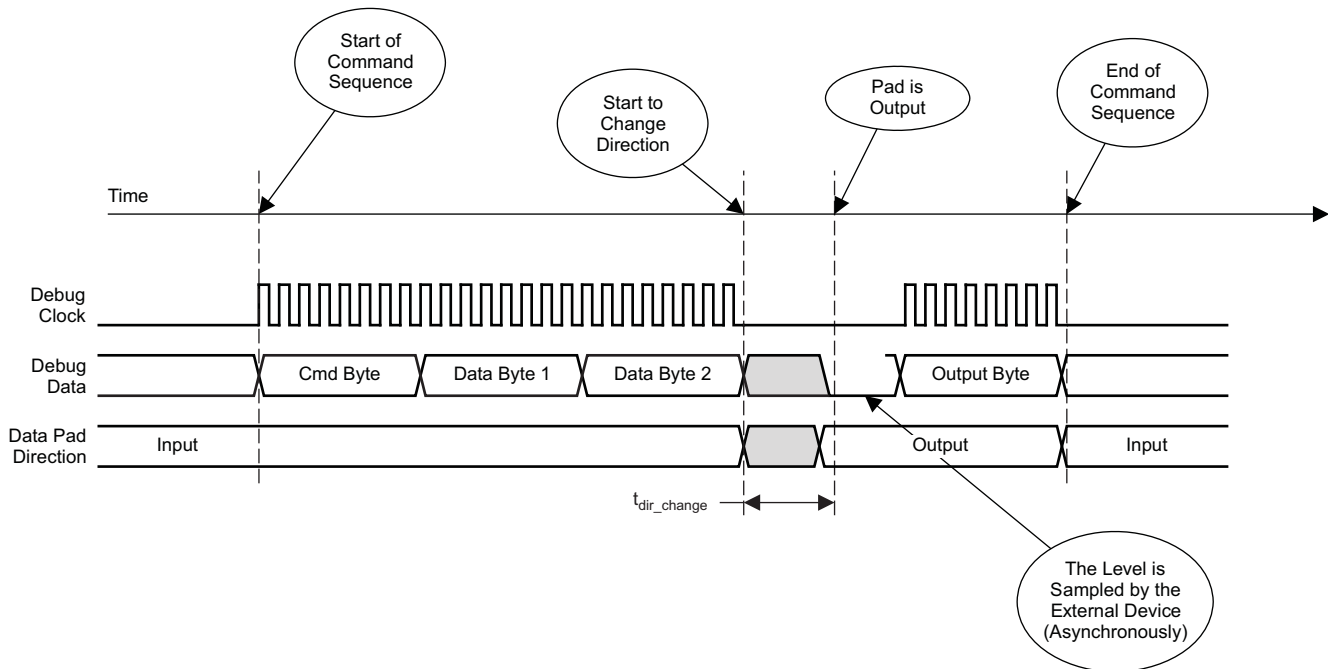
The data is byte-oriented and is transmitted MSB-first. A sequence of one byte is shown in Figure 3-2.



T0303-01

Figure 3-2. Transmission of One Byte

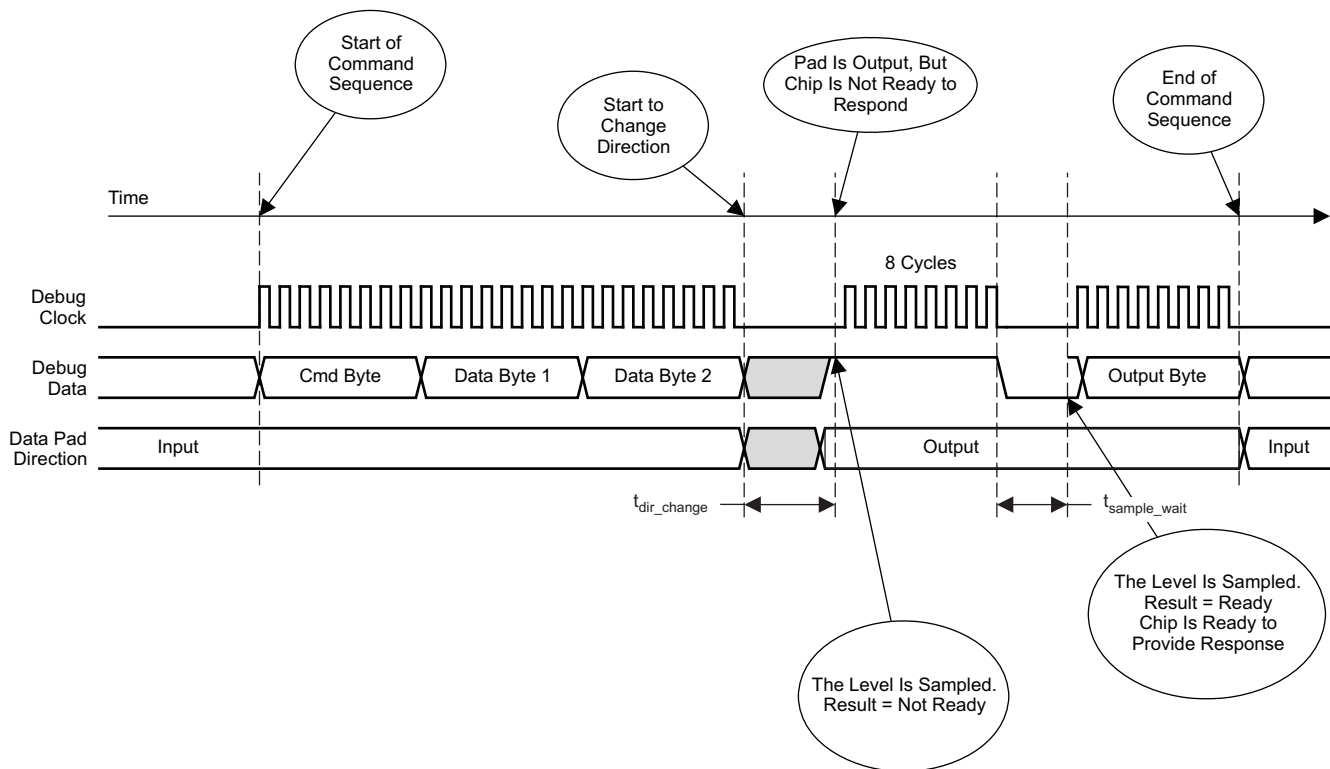
A debug command sequence always starts with the host transmitting a command through the serial interface. This command encodes the number of bytes containing further parameters to follow, and whether a response is required. Based on this command, the debug module controls the direction of the debug data pad. A typical command sequence is shown in [Figure 3-3](#). Note that the debug-data signal is simplified for the clarity of the figure, not showing each individual bit change. The direction is not explicitly indicated to the outside world, but must be derived by the host from the command protocol.



T0304-01

**Figure 3-3. Typical Command Sequence—No Extra Wait for Response**

For commands that require a response, there must be a small idle period between the command and the response to allow the pad to change direction. After the minimum waiting time ( $t_{dir\_change}$ ) of 83 ns, the chip indicates whether it is ready to deliver the response data by pulling the data pad low. The external debugger, which is sampling the data pad, detects this and begins to clock out the response data. If the data pad is high after the waiting time, it is an indication to the debugger that the chip is not ready yet. [Figure 3-4](#) shows how the wait works.



T0305-01

**Figure 3-4. Typical Command Sequence. Wait for Response**

If the debug interface indicates by pulling the data line high that it is not ready to return data, the external device must issue exactly eight clock pulses before it samples the ready level again. This must be repeated until the level is low. The wait cycle is equivalent to reading a byte from the debug interface, but ignoring the result. Note that the pad starts to change direction on the falling edge of the debug clock. Thus, the pad driver drives against the driver in the programmer until the programmer changes pad direction. This duration should be minimized in a programmer implementation.

### 3.3 Debug Commands

The debug commands are shown in [Table 3-1](#). Some of the debug commands are described in further detail in the following subsections.

The 3 least-significant bits (the Xs) are don't care values.

**Table 3-1. Debug Commands**

Command	Instruction Byte	Additional Input Bytes	Output Bytes	Description
CHIP_ERASE	0001 0XXX	0	1	Perform flash chip erase (mass erase) and clear lock bits. If any other command except READ_STATUS is issued, then the use of CHIP_ERASE is disabled. Input byte: none Output byte: Debug status byte. See <a href="#">Table 3-3</a> .
WR_CONFIG	0001 1XXX	1	1	Write debug configuration data. Input byte: See <a href="#">Table 3-2</a> for details. Output byte: Debug status byte. See <a href="#">Table 3-3</a> .
RD_CONFIG	0010 0XXX	0	1	Read debug configuration data. Input byte: none. Output byte: Returns value set by WR_CONFIG command. See <a href="#">Table 3-2</a> .

**Table 3-1. Debug Commands (continued)**

Command	Instruction Byte	Additional Input Bytes	Output Bytes	Description
GET_PC	0010 1XXX	0	2	Return value of 16-bit program counter. Input byte: none Output bytes: Returns 2 bytes.
READ_STATUS	0011 0XXX	0	1	Read status byte. Input byte: none Output byte: Debug status byte. See <a href="#">Table 3-3</a> .
SET_HW_BRKPNT	0011 1XXX	3	1	Set hardware breakpoint. Input bytes: See <a href="#">Section 3.3.3</a> for details. Output byte: Debug status byte. See <a href="#">Table 3-3</a> .
HALT	0100 0XXX	0	1	Halt CPU operation Input byte: none Output byte: Debug status byte. See <a href="#">Table 3-3</a> . If the CPU was already halted, the output is undefined.
RESUME	0100 1XXX	0	1	Resume CPU operation. The CPU must be in the halted state for this command to be run. Input byte: none Output byte: Debug status byte. See <a href="#">Table 3-3</a> .
DEBUG_INSTR	0101 0Xyy	1–3	1	Run debug instruction. The supplied instruction is executed by the CPU without incrementing the program counter. The CPU must be in halted state for this command to be run. Note that yy is number of bytes following the command byte, i.e., how many bytes the CPU instruction has (see <a href="#">Table 2-3</a> ). Input byte(s): CPU instruction Output byte: The resulting accumulator register value after the instruction has been executed
STEP_INSTR	0101 1XXX	0	1	Step CPU instruction. The CPU executes the next instruction from program memory and increments the program counter after execution. The CPU must be in the halted state for this command to be run. Input byte: none Output byte: The resulting accumulator register value after the instruction has been executed
GET_BM	0110 0XXX	0	1	This command does the same thing as GET_PC, except that it returns the memory bank. It returns one byte, where the 3 least-significant bits are the currently used memory bank. Input byte: none Output byte: Memory bank (current value of <code>FMAP.MAP</code> )
GET_CHIP_ID	0110 1XXX	0	2	Return value of 16-bit chip ID and version number. Input byte: none. Output bytes: The <code>CHIPID</code> and <code>CHVER</code> register values
BURST_WRITE	1000 0kkk	2–2049	1	This command writes a sequence of 1–2048 bytes to the <code>DBGDATA</code> register. Each time the register is updated, a <code>DBG_BW</code> DMA trigger is generated.  The number of parameters to the <code>BURST_WRITE</code> command is variable. The number of data bytes in the burst is indicated using the 3 last bits of the command byte (kkk), and the whole next byte. The command sequence is shown in <a href="#">Figure 3-5</a> . The burst length is indicated by an 11-bit value (b10–b0). After these two bytes, the given number of data bytes must be appended. The value 0 means 2048 data bytes; thus, the smallest number of bytes to transfer is 1. Input bytes: Command sequence Output byte: Debug status byte. See <a href="#">Table 3-3</a> .

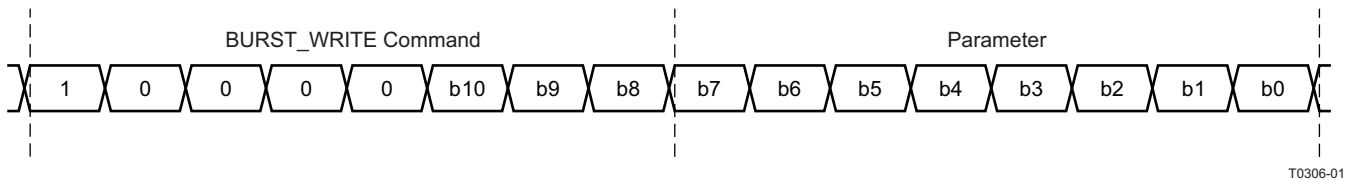


Figure 3-5. Burst Write Command (First 2 Bytes)

T0306-01

### 3.3.1 Debug Configuration

The commands WR\_CONFIG and RD\_CONFIG are used to access the debug-configuration data byte. The format and description of this configuration data are shown in Table 3-2.

Table 3-2. Debug Configuration

Bit	Name	Reset	Description
7:6	–	00	Reserved
5	SOFT_POWER_MODE	1	When set, the digital voltage regulator is not turned off during PM2 and PM3. If this bit is cleared, the debug interface is reset during PM2 and PM3.
4	–	0	Reserved
3	TIMERS_OFF	0	Disable timers. Disable timer operation. This overrides the TIMER_SUSPEND bit and its function. 0: Do not disable timers 1: Disable timers
2	DMA_PAUSE	1	DMA pause. The DMA registers must not be accessed while this bit is set. 0: Enable DMA transfers 1: Pause all DMA transfers
1	TIMER_SUSPEND	1	Suspend timers. Suspend timers when the chip is halted. The timers are also suspended during debug instructions. When executing a STEP, the timers receive exactly (or as close as possible) as many ticks as they would if the program were free-running. 0: Do not suspend timers 1: Suspend timers
0	–	0	Reserved. Always write 0.

### 3.3.2 Debug Status

A debug-status byte is read using the READ\_STATUS command. The format and description of this debug status is shown in Table 3-3.

The READ\_STATUS command is, for example, used for:

- Polling the status of the chip erase (CHIP\_ERASE\_BUSY) after a CHIP\_ERASE command.
- Checking whether the oscillator is stable (OSCILLATOR\_STABLE); required for debug commands HALT, RESUME, DEBUG\_INSTR, STEP\_REPLACE, and STEP\_INSTR.

Table 3-3. Debug Status

Bit	Name	Reset	Description
7	CHIP_ERASE_BUSY	0	Flash chip erase busy The signal is only high when a chip erase is in progress. It goes high immediately after a CHIP_ERASE command is received and returns to low when the flash is fully erased. 0: – 1: Chip erase in progress

**Table 3-3. Debug Status (continued)**

Bit	Name	Reset	Description
6	PCON_IDLE	0	PCON idle. See also <a href="#">Table 3-4</a> . 0: CPU is running. Chip in operational mode controlled by debugger. 1: CPU is not running. Chip is in power mode defined by SLEEP_CMD.MODE register setting. See <a href="#">Section 4.1–Section 4.3</a> for details.
5	CPU_HALTED	0	CPU was halted 0: CPU is running. 1: CPU was halted from a breakpoint or from a HALT debug command.
4	PM_ACTIVE	0	Chip is active. Note that PM0 and PM1 are not supported in debug mode. See also <a href="#">Table 3-4</a> . 0: Chip is in normal operation with CPU running (if not halted). 1: Chip is out of normal operation (active mode) and either in transition up or down from power mode or stable in the power mode defined by the SLEEP_CMD.MODE register setting. See <a href="#">Section 4.1–Section 4.3</a> for details.
3	HALT_STATUS	0	Halt status. Returns cause of last CPU halt 0: CPU was halted by HALT debug command. 1: CPU was halted by hardware breakpoint.
2	DEBUG_LOCKED	0	Debug interface is locked. Returns value of DBGLOCK bit. See <a href="#">Section 3.4.1</a> . 0: Debug interface is not locked. 1: Debug interface is locked.
1	OSCILLATOR_STABLE	0	System clock oscillator stable 0: Oscillators not stable 1: Oscillators stable
0	STACK_OVERFLOW	0	Stack overflow. This bit indicates when the CPU writes to DATA memory space at address 0xFF, which is possibly a stack overflow. 0: No stack overflow 1: Stack overflow

**Table 3-4. Relation Between PCON\_IDLE and PM\_ACTIVE**

PCON_IDLE	PM_ACTIVE	Description
0	0	Chip in normal operation with CPU running (if not halted)
0	1	Chip in transition to start-up from power mode
1	0	Chip in transition to enter power mode
1	1	Chip stable in power mode

### 3.3.3 Hardware Breakpoints

The debug command SET\_HW\_BRKPNT is used to set one of the four available hardware breakpoints. When a hardware breakpoint is enabled, it compares the CPU address bus with the breakpoint. When a match occurs, the CPU is halted.

When issuing the SET\_HW\_BRKPNT, the external host must supply three data bytes that define the hardware breakpoint. The hardware breakpoint itself consists of 19 bits, whereas three bits are used for control purposes. The format of the three data bytes for the SET\_HW\_BRKPNT command is as follows.

The first data byte consists of the following:

- Bits 7–6: Unused
- Bits 5–4: Breakpoint number, 0–3
- Bit 3: 1 = enable, 0 = disable
- Bits 2–0: Memory bank bits. Bits 18–16 of hardware breakpoint.

The second data byte consists of bits 15–8 of the hardware breakpoint.



The third data byte consists of bits 7–0 of the hardware breakpoint. Thus, the second and third data bytes set the CPU CODE address at which to stop execution.

### 3.4 Flash Programming

Programming of the on-chip flash is performed via the debug interface. The external host must initially send instructions using the `DEBUG_INSTR` debug command to perform the flash programming with the flash controller.

#### 3.4.1 Lock Bits

For software and/or access protection, a set of lock bits can be written to the upper available flash page—the lock-bit page. The lock-bit structure consists of 128 bits where the first (FLASH\_PAGES-1) each corresponds to the first flash pages available in the device. The last bit (at the highest address) is the debug lock bit (see [Table 3-5](#)). The structure starts at address 0x7FF0 (address 0xFFFF0 in XDATA) when the upper flash bank is mapped in, and occupies 16 bytes. The rest of the lock-bit page can be used to store code or constants, but cannot be changed without entering debug mode.

The `PAGELOCK[FLASH_PAGES-2:0]` lock-protect bits are used to enable erase and write protection for individual flash memory pages (2 KB; 1 KB on CC2533). There is one bit for each available page.

When the debug-lock bit, `DBGLOCK`, is set to 0 (see [Table 3-5](#)), all debug commands except `CHIP_ERASE`, `READ_STATUS`, and `GET_CHIP_ID` are disabled. The status of the debug-lock bit can be read using the `READ_STATUS` command (see [Section 3.3.2](#)).

Note that after the debug-lock bit has changed due to a write to the lock-bit page or a `CHIP_ERASE` command, the device must be reset to lock or unlock the debug interface.

Issuing a `CHIP_ERASE` command is the only way to clear the debug-lock bit, thereby unlocking the debug interface.

[Table 3-5](#) defines the 16-byte structure containing the flash lock-protection bits. Bit 0 of the first byte contains the lock bit for page 0, bit 1 of the first byte contains the lock bit for page 1, and so on. Bit 7 of the last byte in the flash is the `DBGLOCK` bit (bit 127 in the structure).

**Table 3-5. Flash Lock-Protection Bit Structure Definition**

Bit	Name	Description
127	DBGLOCK	Debug-lock bit 0: Disable debug commands 1: Enable debug commands
126:FLASH_PAGES-1	FREE SPACE	On devices with less than 256 KB memory: Code space available for storing code or constants.
FLASH_PAGES-2:0	PAGELOCK[FLASH_PAGES-2:0]	Page-lock bits. There is one bit for each of the up to 128 pages. Page-lock bits for unavailable pages are not used. 0: Page locked 1: Page not locked

---

**NOTE:** It is recommended to lock all pages that are not to be in-system programmed. This is to prevent erroneous code from unintentionally altering code or constants. This can only be changed while in debug mode.

---

### 3.5 Debug Interface and Power Modes

Power modes PM2 and PM3 may be handled in two different ways when the chip is in debug mode. The default behavior is never to turn off the digital voltage regulator. This emulates power modes while maintaining debug mode operation. The clock sources are turned off as in ordinary power modes. The other option is to turn off the 1.8-V internal digital power. This leads to a complete shutdown of the digital part, which disables debug mode. When the chip is in debug mode, the two options are controlled by configuration bit 5 (`SOFT_POWER_MODE`).

The debug interface still responds to a reduced set of commands while in one of the power modes. The chip can be woken up from sleep mode by issuing a HALT command to the debug interface. The HALT command brings the chip up from sleep mode in the halted state. The RESUME command must be issued to resume software execution.

The debug status may be read when in power modes. The status must be checked when leaving a power mode by issuing a HALT command. The time needed to power up depends on which power mode the chip is in, and must be checked in the debug status. The debug interface only accepts commands that are available in sleep mode before the chip is operational.

---

**NOTE:** Debugging in Idle mode and PM1 is not supported. It is recommended to use active mode or another power mode when debugging.

---

### 3.6 Registers

#### DBGDATA (0x6260) – Debug Data

Bit	Name	Reset	R/W	Description
7:0	BYTE[7:0]	0	R	Debug data from BURST_WRITE command This register is updated each time a new byte has been transferred to the debug interface using the BURST_WRITE command. A DBG_BW DMA trigger is generated when this byte is updated. This allows the DMA controller to fetch the data.

#### CHVER (0x6249) – Chip Version

Bit	Name	Reset	R/W	Description
7:0	VERSION[7:0]	Chip dependent	R	Chip revision number

#### CHIPID (0x624A) – Chip ID

Bit	Name	Reset	R/W	Description
7:0	CHIPID[7:0]	Chip dependent	R	Chip identification number. CC2530: 0xA5 CC2531: 0xB5 CC2533: 0x95 CC2540: 0x8D CC2541: 0x41

#### CHIPINFO0 (0x6276) – Chip Information Byte 0

Bit	Name	Reset	R/W	Description
7	–	0	R0	Reserved. Always 0.
6:4	FLASHSIZE[2:0]	Chip dependent	R	Flash Size. 001 – 32 KB, 010 – 64 KB, 011 – 128 KB (for CC2533: 011 – 96 KB), 100 – 256 KB
3	USB	Chip dependent	R	1 if chip has USB, 0 otherwise
2	–	1	R1	Reserved. Always 1
1:0	–	00	R0	Reserved. Always 00

#### CHIPINFO1 (0x6277) – Chip Information Byte 1

Bit	Name	Reset	R/W	Description
7:3	–	Chip dependent	R	Reserved.
2:0	SRAMSIZE[2:0]	Chip dependent	R	SRAM size in KB minus 1. For example, a 4-KB device has this field set to 011. Add 1 to the number to get the number of KB available.

## Power Management and Clocks

---

---

Low-power operation is enabled through different operating modes (power modes). The various operating modes are referred to as active mode, idle mode, and power modes 1, 2, and 3 (PM1–PM3).

Topic	Page
<b>4.1 Power Management Introduction</b> .....	<b>61</b>
<b>4.2 Power-Management Control</b> .....	<b>62</b>
<b>4.3 Power-Management Registers</b> .....	<b>63</b>
<b>4.4 Oscillators and Clocks</b> .....	<b>66</b>
<b>4.5 Timer Tick Generation</b> .....	<b>69</b>
<b>4.6 Data Retention</b> .....	<b>69</b>

## 4.1 Power Management Introduction

Different operating modes, or power modes, are used to allow low-power operation. Ultralow-power operation is obtained by turning off the power supply to modules to avoid static (leakage) power consumption and also by using clock gating and turning off oscillators to reduce dynamic power consumption.

The five various operating modes (power modes) are called active mode, idle mode, PM1, PM2, and PM3 (PM1, PM2, and PM3 are also referred to as sleep modes). Active mode is the normal operating mode, whereas PM3 has the lowest power consumption. The impact of the different power modes on system operation is shown in [Table 4-1](#), together with voltage regulator and oscillator options.

**Table 4-1. Power Modes**

Power Mode	High-Frequency Oscillator		Low-Frequency Oscillator		Voltage Regulator (Digital)
Configuration	A	32-MHz XOSC	C	32-kHz XOSC	
	B	16-MHz RCOSC	D	32-kHz RCOSC	
Active or idle mode	A or B		C or D		ON
PM1	None		C or D		ON
PM2	None		C or D		OFF
PM3	None		None		OFF

**Active mode:** The fully functional mode. The voltage regulator to the digital core is on, and either the 16-MHz RC oscillator or the 32-MHz crystal oscillator or both are running. Either the 32-kHz RCOSC or the 32-kHz XOSC is running.

**Idle mode:** Identical to active mode, except that the CPU core stops operating (is idle).

**PM1:** The voltage regulator to the digital part is on. Neither the 32-MHz XOSC nor the 16-MHz RCOSC is running. Either the 32-kHz RCOSC or the 32-kHz XOSC is running. The system goes to active mode on reset, an external interrupt, or when the Sleep Timer expires.

**PM2:** The voltage regulator to the digital core is turned off. Neither the 32-MHz XOSC nor the 16-MHz RCOSC is running. Either the 32-kHz RCOSC or the 32-kHz XOSC is running. The system goes to active mode on reset, an external interrupt, or when the Sleep Timer expires.

**PM3:** The voltage regulator to the digital core is turned off. None of the oscillators is running. The system goes to active mode on reset or an external interrupt.

The POR is active in PM2 and PM3, but the BOD is powered down, which gives limited voltage supervision. If the supply voltage is lowered to below 1.4 V during PM2 or PM3, at temperatures of 70°C or higher, and then brought back up to good operating voltage before active mode is re-entered, registers and RAM contents that are saved in PM2 or PM3 may become altered. Hence, care should be taken in the design of the system power supply to ensure that this does not occur. The voltage can be periodically supervised accurately by entering active mode, as a BOD reset is triggered if the supply voltage is below approximately 1.7 V.

The CC2533 and CC2541 have functionality to perform automatically a CRC check of the retained configuration register values in PM2 and PM3 to check that the device state was not altered during sleep. The bits in `SRCRC.CRC_RESULT` indicate whether there were any changes, and by enabling `SRCRC.CRC_RESET_EN`, the device immediately resets itself with a watchdog reset if `SRCRC.CRC_RESULT` is not 00 (= CRC of retained registers passed) after wakeup from PM2 or PM3. The `SRCRC` register also contains the `SRCRC.FORCE_RESET` bit that can be used by software to trigger a watchdog reset immediately to reboot the device.

For CC2533 and CC2541, additional analog reset architecture adds another brownout detector (the 3VBOD) that senses on the unregulated voltage. The purpose of this 3VBOD is to reduce the current consumption of the device when supplied with voltages well below the operating voltage.

### 4.1.1 Active and Idle Modes

Active mode is the fully functional mode of operation where the CPU, peripherals, and RF transceiver are active. The digital voltage regulator is turned on.

Active mode is used for normal operation. By enabling the `PCON.IDLE` bit while in active mode (`SLEEP_CMD.MODE = 0x00`), the CPU core stops operating and the idle mode is entered. All other peripherals function normally, and any enabled interrupt wakes up the CPU core (to transition back from idle mode to active mode).

### 4.1.2 PM1

In PM1, the high-frequency oscillators are powered down (32-MHz XOSC and 16-MHz RCOSC). The voltage regulator and the enabled 32-kHz oscillator are on. When PM1 is entered, a power-down sequence is run.

PM1 is used when the expected time until a wakeup event is relatively short (less than 3 ms), because PM1 uses a fast power-down and power-up sequence.

### 4.1.3 PM2

PM2 has the second-lowest power consumption. In PM2, the power-on reset, external interrupts, selected 32-kHz oscillator, and Sleep Timer peripherals are active. I/O pins retain the I/O mode and output value set before entering PM2. All other internal circuits are powered down. The voltage regulator is also turned off. When PM2 is entered, a power-down sequence is run.

PM2 is typically entered when using the Sleep Timer as the wakeup event, and also combined with external interrupts. PM2 should typically be chosen, compared to PM1, when expected sleep time exceeds 3 ms. Using less sleep time does not reduce system power consumption compared to using PM1.

### 4.1.4 PM3

PM3 is used to achieve the operating mode with the lowest power consumption. In PM3, all internal circuits that are powered from the voltage regulator are turned off (basically all digital modules; the only exceptions are interrupt detection and POR level sensing). The internal voltage regulator and all oscillators are also turned off.

Reset (POR or external) and external I/O port interrupts are the only functions that operate in this mode. I/O pins retain the I/O mode and output value set before entering PM3. A reset condition or an enabled external I/O interrupt event wakes the device up and places it into active mode (an external interrupt starts from where it entered PM3, whereas a reset returns to start-of-program execution). The content of RAM and registers is partially preserved in this mode (see [Section 4.6](#)). PM3 uses the same power-down and power-up sequence as PM2.

PM3 is used to achieve ultralow-power consumption when waiting for an external event. It should be used when expected sleep time exceeds 3 ms.

## 4.2 Power-Management Control

The required power mode is selected by the MODE bits in the `SLEEP_CMD` control register and the `PCON.IDLE` bit. Setting the SFR register `PCON.IDLE` bit enters the mode selected by `SLEEP_CMD.MODE`.

An enabled interrupt from port pins or Sleep Timer or a power-on reset wakes the device from other power modes and brings it into active mode.

When PM1, PM2, or PM3 is entered, a power-down sequence is run. When the device is taken out of PM1, PM2, or PM3, it starts at 16 MHz and automatically changes to 32 MHz if `CLKCONCMD.OSC` was 0 when entering the power mode (setting `PCON.IDLE`). If `CLKCONCMD.OSC` was 1 when `PCON.IDLE` was set, when entering the power mode, it continues to run at 16 MHz.

The instruction that sets the `PCON.IDLE` bit must be aligned in a certain way for correct operation. The first byte of the assembly instruction immediately following this instruction must not be placed on a 4-byte boundary. Furthermore, cache must not be disabled (see `CM` in the `FCTL` register description in [Chapter 6](#)). Failure to comply with this requirement may cause higher current consumption. Provided this requirement is fulfilled, the first assembly instruction after the instruction that sets the `PCON.IDLE` bit is performed before the ISR of the interrupt that caused the system to wake up, but after the system woke up. If this instruction is a global interrupt disable, it is possible to have it followed by code for execution after wakeup, but before the ISR is serviced.

An example of how this can be done in the IAR compiler is shown as follows. The command for setting `PCON` to 1 is placed in a function written in assembly code. In a C file calling this function, a declaration such as `extern void`

```
EnterSleepModeDisableInterruptsOnWakeup(void);
```

is used. The `RSEG NEAR_CODE:CODE:NOROOT(2)` statement ensures that the `MOV PCON,#1` instruction is placed on a 2-byte boundary. It is a 3-byte instruction, so the following instruction is not placed on a 4-byte boundary, as required. In the following example, this instruction is `CLR EA`, which disables all interrupts. That means that the ISR of the interrupt that woke up the system is not executed until after the `IEN0.EA` bit has been set again later in the code. If this functionality is not wanted, the `CLR EA` instruction can be replaced by a `NOP`.

```
PUBLIC EnterSleepModeDisableInterruptsOnWakeup FUNCTION
EnterSleepModeDisableInterruptsOnWakeup,0201H RSEG NEAR_CODE:CODE:NOROOT(2)
EnterSleepModeDisableInterruptsOnWakeup: MOV PCON,#1 CLR EA RET
```

### 4.3 Power-Management Registers

This section describes the power-management registers. All register bits retain their previous values when entering PM2 or PM3.

#### SRCRC (0x6262) – Sleep Reset CRC (CC2533 and CC2541 only)

Bit	Name	Reset	R/W	Description
7	XOSC_AMP_DET_EN	0	R/W	0: Disable 1: Enable the amplitude detector for the 32-MHz XOSC, CC2533 only
6	–	0	R0	Reserved. Always read 0.
5	FORCE_RESET	0	R/W	0: No action 1: Force watchdog reset.
4	–	0	R	Reserved
3:2	CRC_RESULT	00	R/W0	00: CRC of retained registers passed 01: Low CRC value failed 10: High CRC value failed 11: Both CRC values failed
1	–	0	R	Reserved
0	CRC_RESET_EN	0	R/W	0: Disable reset of chip due to CRC. 1: Enable reset of chip if <code>CRC_RESULT != 00</code> after wakeup from PM2 or PM3.

#### PCON (0x87) – Power Mode Control

Bit	Name	Reset	R/W	Description
7:1	–	0000 000	R/W	Reserved, always write as 0000 000.
0	IDLE	0	R0/W H0	Power mode control. Writing 1 to this bit forces the device to enter the power mode set by <code>SLEEP_CMD.MODE</code> (note that <code>MODE = 0x00 AND IDLE = 1</code> stops the CPU core activity). This bit is always read as 0. All enabled interrupts clear this bit when active, and the device re-enters active mode.

**SLEEP\_CMD (0xBE) – Sleep-Mode Control Command**

Bit	Name	Reset	R/W	Description
7	OSC32K_CALDIS	0	R/W	Disable 32-kHz RC oscillator calibration 0: 32-kHz RC oscillator calibration is enabled. 1: 32-kHz RC oscillator calibration is disabled. This setting can be written at any time, but does not take effect before the chip has been running on the 16-MHz high-frequency RC oscillator.
6:3	–	000 0	R0	Reserved
2	–	1	R/W	Reserved. Always write as 1
1:0	MODE[1:0]	00	R/W	Power-mode setting 00: Active or Idle mode 01: Power mode 1 (PM1) 10: Power mode 2 (PM2) 11: Power mode 3 (PM3)

**SLEEP\_STA (0x9D) – Sleep-Mode Control Status**

Bit	Name	Reset	R/W	Description
7	OSC32K_CALDIS	0	R	32-kHz RC oscillator calibration status SLEEP_STA.OSC32K_CALDIS shows the current status of disabling of the 32-kHz RC calibration. The bit is not set to the same value as SLEEP_CMD.OSC32K_CALDIS before the chip has been run on the 32-kHz RC oscillator.
6:5	–	00	R	Reserved
4:3	RST[1:0]	XX	R	Status bit indicating the cause of the last reset. If there are multiple resets, the register only contains the last event. 00: Power-on reset and brownout detection 01: External reset 10: Watchdog Timer reset 11: Clock loss reset
2:1	–	00	R	Reserved
0	CLK32K	0	R	The 32-kHz clock signal (synchronized to the system clock)



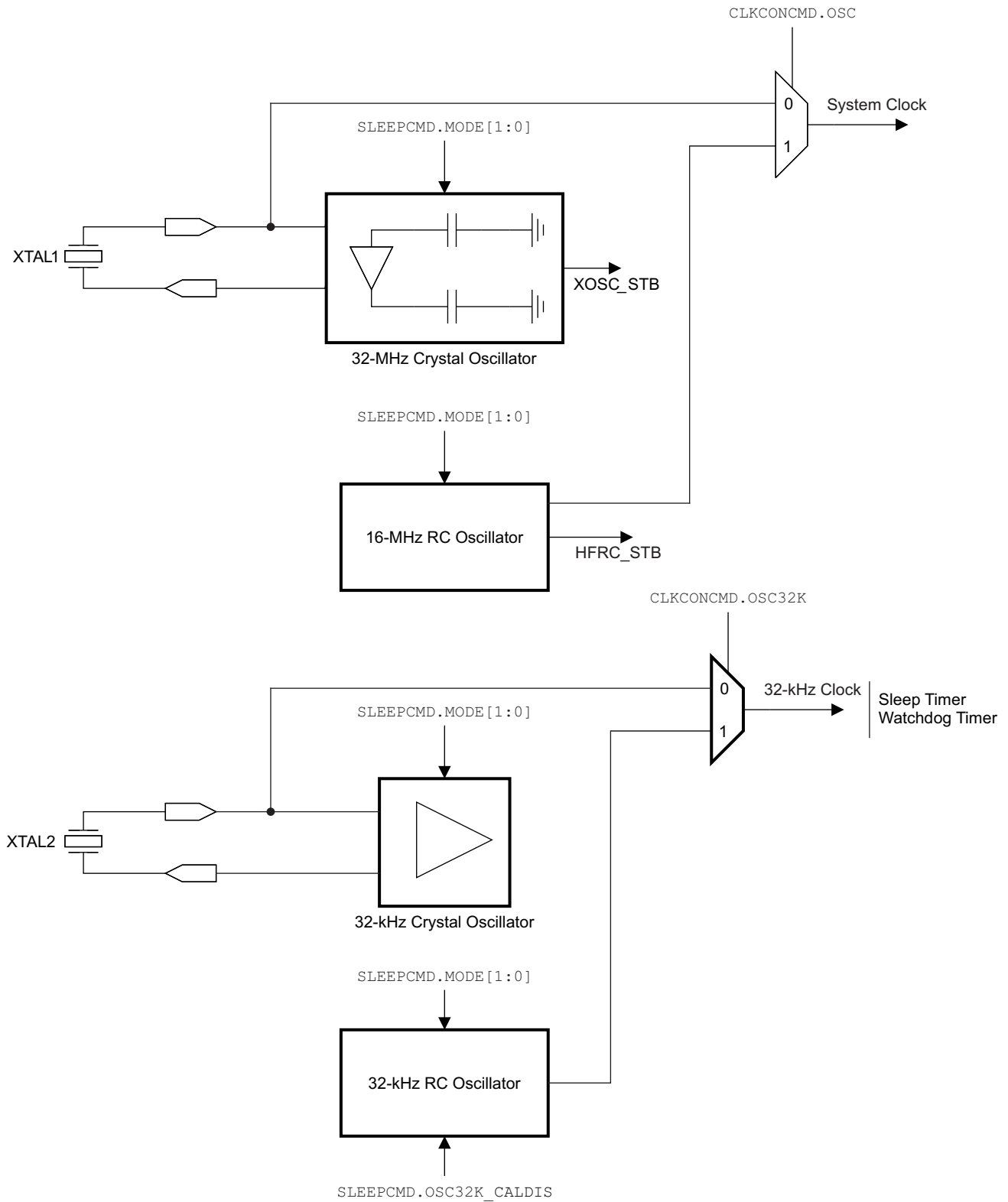


Figure 4-1. Clock System Overview

B0303-02

## 4.4 Oscillators and Clocks

The device has one internal system clock, or main clock. The source for the system clock can be either the 16-MHz RC oscillator or the 32-MHz crystal oscillator. Clock control is performed using the `CLKCONCMD` SFR register.

There is also one 32-kHz clock source that can either be an RC oscillator or a crystal oscillator, also controlled by the `CLKCONCMD` register.

The `CLKCONSTA` register is a read-only register used for getting the current clock status.

The choice of oscillator allows a trade-off between high accuracy in the case of the crystal oscillator and low power consumption when the RC oscillator is used. Note that operation of the RF transceiver requires that the 32-MHz crystal oscillator is used.

In the CC2533, CC2540 and CC2541, an additional module for detection of 32-MHz XOSC stability is available. This amplitude detector can be useful in environments with significant noise on the power supply, to ensure that the clock source is not used until the clock signal is stable. In the CC2533, this module can be enabled by setting the `SRCRC.XOSC_AMP_DET_EN` bit, and this adds around 20  $\mu$ s to the 32-MHz XOSC startup time. In the CC2540 and CC2541, the module is always enabled.

### 4.4.1 Oscillators

Figure 4-1 gives an overview of the clock system with available clock sources.

Two high-frequency oscillators are present in the device:

- 32-MHz crystal oscillator
- 16-MHz RC oscillator

The 32-MHz crystal-oscillator start-up time may be too long for some applications; therefore, the device can run on the 16-MHz RC oscillator until the crystal oscillator is stable. The 16-MHz RC oscillator consumes less power than the crystal oscillator, but because it is not as accurate as the crystal oscillator it cannot be used for RF transceiver operation.

Two low-frequency oscillators are present in the device:

- 32-kHz crystal oscillator
- 32-kHz RC oscillator.

The 32-kHz XOSC is designed to operate at 32.768 kHz and provide a stable clock signal for systems requiring time accuracy. The 32-kHz RCOSC runs at 32.753 kHz when calibrated. The calibration can only take place when the 32-MHz XOSC is enabled, and this calibration can be disabled by enabling the `SLEEP_CMD.OSC32K_CALDIS` bit. The 32-kHz RCOSC should be used to reduce cost and power consumption compared to the 32-kHz XOSC solution. The two 32-kHz oscillators cannot be operated simultaneously.

### 4.4.2 System Clock

The system clock is derived from the selected system clock source, which is the 32-MHz XOSC or the 16-MHz RCOSC. The `CLKCONCMD.OSC` bit selects the source of the system clock. Note that to use the RF transceiver, the 32-MHz crystal oscillator must be selected and stable.

Note that changing the `CLKCONCMD.OSC` bit does not cause the system clock to change instantly. The clock source change first takes effect when `CLKCONSTA.OSC = CLKCONCMD.OSC`. This is due to the requirement to have stable clocks prior to actually changing the clock source. Also note that the `CLKCONCMD.CLKSPD` bit reflects the frequency of the system clock and thus is a mirror of the `CLKCONCMD.OSC` bit.

The 16 MHz RC oscillator is calibrated once after the 32-MHz XOSC has been selected and is stable, that is, when the `CLKCONSTA.OSC` bit switches from 1 to 0.

---

**NOTE:** The change from the 16-MHz clock source to the 32-MHz clock source (and vice versa) aligns with the `CLKCONCMD.TICKSPD` setting. A slow `CLKCONCMD.TICKSPD` setting when `CLKCONCMD.OSC` is changed results in a longer time before the actual source change takes effect. The fastest switching is obtained when `CLKCONCMD.TICKSPD` equals 000.

---

**NOTE:** After coming up from PM1, PM2, or PM3, the CPU must wait for `CLKCONSTA.OSC` to be 0 before operations requiring the system to run on the 32-MHz XOSC (such as the radio) are started.

---

### 4.4.3 32-kHz Oscillators

Two 32-kHz oscillators are present in the device as clock sources for the 32-kHz clock:

- 32-kHz XOSC
- 32-kHz RCOSC

By default, after a reset, the 32-kHz RCOSC is enabled and selected as the 32-kHz clock source. The RCOSC consumes less power, but is less accurate compared to the 32-kHz XOSC. The chosen 32-kHz clock source drives the Sleep Timer, generates the tick for the Watchdog Timer, and is used as a strobe in Timer 2 to calculate the Sleep Timer sleep time. The `CLKCONCMD.OSC32K` register bit selects the oscillator to be used as the 32-kHz clock source. This bit does not give an indication of the stability of the 32-kHz XOSC.

The `CLKCONCMD.OSC32K` register bit can be written at any time, but does not take effect before the 16-MHz RCOSC is the active system clock source. When system clock is changed from the 16-MHz RCOSC to the 32-MHz XOSC (`CLKCONCMD.OSC` from 1 to 0), calibration of the 32-kHz RCOSC starts up and is performed once if the 32-kHz RCOSC is selected. During calibration, a divided version of the 32-MHz XOSC is used. The result of the calibration is that the 32-kHz RCOSC is running at 32.753 kHz. The 32-kHz RCOSC calibration may take up to 2 ms to complete. Calibration can be disabled by setting `SLEEP_CMD.OSC32K_CALDIS` to 1. At the end of the calibration, an extra pulse may occur on the 32-kHz clock source, which causes the sleep timer to be incremented by 1.

Note that after having switched to the 32-kHz XOSC and when coming up from PM3 with the 32-kHz XOSC enabled, the oscillator requires up to 500 ms to stabilize on the correct frequency. The Sleep Timer, Watchdog Timer and clock-loss detector should not be used before the 32-kHz XOSC is stable.

### 4.4.4 Oscillator and Clock Registers

This section describes the oscillator and clock registers. All register bits retain their previous values when entering PM2 or PM3.

**CLKCONCMD (0xC6) – Clock Control Command**

Bit	Name	Reset	R/W	Description
7	OSC32K	1	R/W	32-kHz clock-source select. Setting this bit initiates a clock-source change only. CLKCONSTA.OSC32K reflects the current setting. The 16-MHz RCOSC must be selected as system clock when this bit is to be changed. This bit does not give an indication of the stability of the 32-kHz XOSC. 0: 32 kHz XOSC 1: 32 kHz RCOSC
6	OSC	1	R/W	System clock-source select. Setting this bit initiates a clock-source change only. CLKCONSTA.OSC reflects the current setting. 0: 32 MHz XOSC 1: 16 MHz RCOSC
5:3	TICKSPD[2:0]	001	R/W	Timer ticks output setting. Cannot be higher than system clock setting given by OSC bit setting. 000: 32 MHz 001: 16 MHz 010: 8 MHz 011: 4 MHz 100: 2 MHz 101: 1 MHz 110: 500 kHz 111: 250 kHz Note that CLKCONCMD . TICKSPD can be set to any value, but the effect is limited by the CLKCONCMD . OSC setting; that is, if CLKCONCMD . OSC = 1 and CLKCONCMD . TICKSPD = 000, CLKCONSTA . TICKSPD reads 001, and the real TICKSPD is 16 MHz.
2:0	CLKSPD	001	R/W	Clock speed. Cannot be higher than system clock setting given by the OSC bit setting. Indicates current system-clock frequency 000: 32 MHz 001: 16 MHz 010: 8 MHz 011: 4 MHz 100: 2 MHz 101: 1 MHz 110: 500 kHz 111: 250 kHz Note that CLKCONCMD . CLKSPD can be set to any value, but the effect is limited by the CLKCONCMD . OSC setting; that is, if CLKCONCMD . OSC = 1 and CLKCONCMD . CLKSPD = 000, CLKCONSTA . CLKSPD reads 001, and the real CLKSPD is 16 MHz. Note also that the debugger cannot be used with a divided system clock. When running the debugger, the value of CLKCONCMD . CLKSPD should be set to 000 when CLKCONCMD . OSC = 0 or to 001 when CLKCONCMD . OSC = 1.

**CLKCONSTA (0x9E) – Clock Control Status**

Bit	Name	Reset	R/W	Description
7	OSC32K	1	R	Current 32-kHz clock source selected: 0: 32-kHz XOSC 1: 32-kHz RCOSC
6	OSC	1	R	Current system clock selected: 0: 32-MHz XOSC 1: 16-MHz RCOSC
5:3	TICKSPD[2:0]	001	R	Current timer ticks output setting 000: 32 MHz 001: 16 MHz 010: 8 MHz 011: 4 MHz 100: 2 MHz 101: 1 MHz 110: 500 kHz 111: 250 kHz
2:0	CLKSPD	001	R	Current clock speed 000: 32 MHz 001: 16 MHz 010: 8 MHz 011: 4 MHz 100: 2 MHz 101: 1 MHz 110: 500 kHz 111: 250 kHz

#### 4.5 Timer Tick Generation

The value of the `CLKCONCMD.TICKSPD` register controls a global prescaler for Timer 1, Timer 3, and Timer 4. The prescaler value can be set to a value from 0.25 MHz to 32 MHz. It should be noted that if `CLKCONCMD.TICKSPD` indicates a higher frequency than the system clock, the actual prescaler value indicated in `CLKCONSTA.TICKSPD` is the same as the system clock.

#### 4.6 Data Retention

In power modes PM2 and PM3, power is removed from most of the internal circuitry. However, SRAM retains its contents, and the content of internal registers is also retained in PM2 and PM3.

All CPU, RF, and peripheral registers retain their contents in PM2 and PM3, except the AES, I<sup>2</sup>C, and USB registers, OBSSEL0–OBSSEL5, TR0, and in the CC2541, LLECTRL.

Switching to the PM2 or PM3 low-power modes appears transparent to software. Note that the value of the Sleep Timer is not preserved in PM3.

All registers retain their values in PM1.

## Reset

The device has five reset sources. The following events generate a reset:

- Forcing the RESET\_N input pin low
- A power-on reset condition
- A brownout reset condition
- Watchdog Timer reset condition
- Clock-loss reset condition

The initial conditions after a reset are as follows:

- I/O pins are configured as inputs with pullups (P1.0 and P1.1 are inputs, but do not have a pullup or pulldown)
- CPU program counter is loaded with 0x0000 and program execution starts at this address
- All peripheral registers are initialized to their reset values (see register descriptions)
- Watchdog Timer is disabled
- Clock-loss detector is disabled

During reset, the I/O pins are configured as inputs with pullups (P1.0 and P1.1 are inputs, but do not have a pullup or pulldown). The RESET\_N input is always configured as an input with pullup.

In the CC2533 and CC2541, a watchdog reset can be generated immediately in software by writing the SRCRC.FORCE\_RESET bit to 1 (see [Section 4.3](#) for the register description). In the other devices in the family, a watchdog reset can be triggered from software by enabling the watchdog timer with the shortest time-out and waiting for it to trigger.

Topic	Page
<b>5.1 Power-On Reset and Brownout Detector .....</b>	<b>71</b>
<b>5.2 Clock-Loss Detector .....</b>	<b>71</b>

## 5.1 Power-On Reset and Brownout Detector

The device includes a power-on reset (POR), providing correct initialization during device power on. It also includes a brownout detector (BOD) operating on the regulated 1.8-V digital power supply only. The BOD protects the memory contents during supply voltage variations which cause the regulated 1.8-V power to drop below the minimum level required by digital logic, flash memory, and SRAM.

When power is initially applied, the POR and BOD hold the device in the reset state until the supply voltage rises above the power-on-reset and brownout voltages.

The cause of the last reset can be read from the register bits `SLEEPSTA.RST`. It should be noted that a BOD reset is read as a POR reset.

## 5.2 Clock-Loss Detector

The clock-loss detector can be used in safety-critical systems to detect that one of the XOSC clock sources (32-MHz XOSC or 32-kHz XOSC) has stopped. This can typically happen due to damage to the external crystal or supporting components. When the clock-loss detector is enabled, the two clocks monitor each other continuously. If one of the clocks stops toggling, a clock-loss detector reset is generated within a certain maximum time-out period. The time-out depends on which clock stops. If the 32-kHz clock stops, the time-out period is 0.5 ms. If the 32-MHz clock stops, the time-out period is 0.25 ms. When the system comes up again from reset, software can detect the cause of the reset by reading `SLEEPSTA.RST[1:0]`. After a reset, the internal RC oscillators are used. Thus, the system is able to start up again and can then be powered down gracefully. The clock-loss detector is enabled or disabled with the `CLD.EN` bit. It is assumed that the 32-MHz XOSC is selected as system clock source when using the clock-loss detector. The 32-kHz clock can be 32-kHz RCOSC (should be calibrated for accurate reset timeout) or 32-kHz XOSC.

In power modes 1 and 2, the clock-loss detector is automatically stopped and restarted when the clocks start up again.

Before entering power mode 3, switch to the 16-MHz RCOSC and disable the clock-loss detector. When entering active mode again, turn on the clock-loss detector and then switch back to the 32-MHz XOSC.

**CLD (0x6290) – Clock-Loss Detection**

Bit	Name	Reset	R/W	Description
7:1	–	0000 000	R0	Reserved
0	EN	0	R/W	Clock-loss detector enable 0: Detector disabled 1: Detector enabled

## Flash Controller

The device contains flash memory for storage of program code. The flash memory is programmable from the user software and through the debug interface.

The flash controller handles writing and erasing the embedded flash memory. The embedded flash memory consists of up to 128 pages of 2048 bytes (CC2530, CC2531, CC2540, and CC2541) or 1024 bytes (CC2533) each.

The flash controller has the following features:

- 32-bit word programmable
- Page erase
- Lock bits for write protection and code security
- Flash-page erase timing 20 ms
- Flash-chip erase timing 20 ms
- Flash-write timing (4 bytes) 20  $\mu$ s

Topic	Page
<b>6.1 Flash Memory Organization</b> .....	<b>73</b>
<b>6.2 Flash Write</b> .....	<b>73</b>
<b>6.3 Flash Page Erase</b> .....	<b>75</b>
<b>6.4 Flash DMA Trigger</b> .....	<b>76</b>
<b>6.5 Flash Controller Registers</b> .....	<b>76</b>



## 6.1 Flash Memory Organization

The flash memory is divided into 2048-byte or 1024-byte flash pages. A flash page is the smallest erasable unit in the memory, whereas a 32-bit word is the smallest writable unit that can be written to the flash.

When performing write operations, the flash memory is word-addressable using a 16-bit address written to the address registers `FADDRH:FADDRL`.

When performing page-erase operations, the flash memory page to be erased is addressed through the register bits `FADDRH[7:1]` (CC2530, CC2531, CC2540, and CC2541) or `FADDRH[6:0]` (CC2533).

Note the difference in addressing the flash memory; when accessed by the CPU to read code or data, the flash memory is byte-addressable. When accessed by the flash controller, the flash memory is word-addressable, where a word consists of 32 bits.

The following sections describe the procedures for flash write and flash page-erase in detail.

## 6.2 Flash Write

The flash is programmed serially with a sequence of one or more 32-bit words (4 bytes), starting at the start address (set by `FADDRH:FADDRL`). In general, a page must be erased before writing can begin. The page-erase operation sets all bits in the page to 1. The chip-erase command (through the debug interface) erases all pages in the flash. This is the only way to set bits in the flash to 1. When writing a word to the flash, the 0-bits are programmed to 0 and the 1-bits are ignored (leaves the bit in the flash unchanged). Thus, bits are erased to 1 and can be written to 0. It is possible to write multiple times to a word. This is described in [Section 6.2.2](#).

### 6.2.1 Flash-Write Procedure

The flash-write sequence algorithm is as follows:

1. Set `FADDRH:FADDRL` to the start address. (This is the 16 MSBs of the 18-bit byte address).
2. Set `FCTL.WRITE` to 1. This starts the write-sequence state machine.
3. Write four times to `FWDATA` within 20  $\mu$ s (since the last time `FCTL.FULL` became 0, if not first iteration). LSB is written first. (`FCTL.FULL` goes high after the last byte.)
4. Wait until `FCTL.FULL` goes low. (The flash controller has started programming the 4 bytes written in step 3 and is ready to buffer the next 4 bytes).
5. Optional status check step:
  - If the 4 bytes were not written fast enough in step 3, the operation has timed out and `FCTL.BUSY` (and `FCTL.WRITE`) are 0 at this stage.
  - If the 4 bytes could not be written to the flash due to the page being locked, `FCTL.BUSY` (and `FCTL.WRITE`) are 0 and `FCTL.ABORT` is 1.
6. If this was the last 4 bytes then quit, otherwise go to step 3.

The write operation is performed using one of two methods:

- Using DMA transfer (preferred method)
- Using CPU, running code from SRAM

The CPU cannot access the flash, for example, to read program code while a flash-write operation is in progress. Therefore, the program code executing the flash write must be executed from RAM. See [Section 2.2.1](#) for a description of how to run code from RAM.

When a flash-write operation is executed from RAM, the CPU continues to execute code from the next instruction after initiation of the flash-write operation (`FCTL.WRITE = 1`).

Power mode 1, 2, or 3 must not be entered while writing to the flash. Also, the system clock source (XOSC/RCOSC) must not be changed while writing. Note that setting `CLKCONSTA.CLKSPD` to a high value makes it impossible to meet the timing requirement of 20- $\mu$ s write timing. With `CLKCONSTA.CLKSPD = 111`, the clock period is only 4  $\mu$ s. It is therefore recommended to keep `CLKCONSTA.CLKSPD` at 000 or 001 while writing to the flash.

## 6.2.2 Writing Multiple Times to a Word

The following rules apply when writing multiple times to a 32-bit word between erase:

- Writing 0 to a bit within a 32-bit flash word, which has been set to 1 by the last erase operation, changes the state of the bit to 0, subject to the last bullet below.
- It is possible to write 0 to a bit within a 32-bit word repeatedly (subject to the last bullet below) once the bit has been written with 0. This does not change the state of the bit.
- Writing 1 to a bit does not change the state of the bit, subject to the last bullet below.
- The following limitations apply to writes subsequent to the last page erase:
  - A 0 must not be written more than two times to a single bit.
  - A 32-bit word shall not be written more than 8 times.
  - A page must not be written more than 1024 times.

The state of any bit of a 32-bit flash word is nondeterministic if these limitations are violated.

This makes it possible to write up to 4 new bits to a 32-bit word 8 times. One example write sequence to a word is shown in [Table 6-1](#). Here  $b_n$  represents the 4 new bits written to the word for each update. This technique is useful to maximize the lifetime of the flash for data-logging applications.

**Table 6-1. Example Write Sequence**

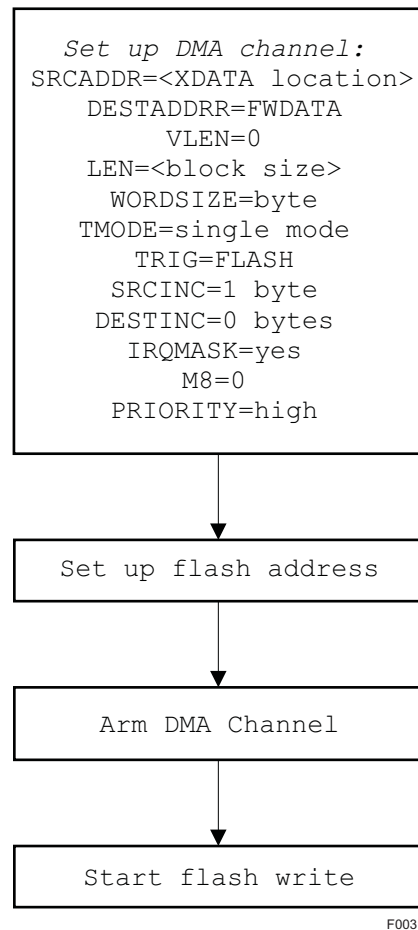
Step	Value Written	FLASH Contents After Writing	Comment
1	(page erase)	0xFFFFFFFF	The erase sets all bits to 1.
2	0xFFFFFFFF $b_0$	0xFFFFFFFF $b_0$	Only the bits written 0 are set to 0, whereas all bits written 1 are ignored.
3	0xFFFFFFFF $b_1$ F	0xFFFFFFFF $b_1$ $b_0$	Only the bits written 0 are set to 0, whereas all bits written 1 are ignored.
4	0xFFFFFFFF $b_2$ FF	0xFFFFFFFF $b_2$ $b_1$ $b_0$	Only the bits written 0 are set to 0, whereas all bits written 1 are ignored.
5	0xFFFFFFFF $b_3$ FFF	0xFFFFFFFF $b_3$ $b_2$ $b_1$ $b_0$	Only the bits written 0 are set to 0, whereas all bits written 1 are ignored.
6	0xFF $b_4$ FFFF	0xFF $b_4$ $b_3$ $b_2$ $b_1$ $b_0$	Only the bits written 0 are set to 0, whereas all bits written 1 are ignored.
7	0xFF $b_5$ FFFFF	0xFF $b_5$ $b_4$ $b_3$ $b_2$ $b_1$ $b_0$	Only the bits written 0 are set to 0, whereas all bits written 1 are ignored.
8	0xF $b_6$ FFFFFF	0xF $b_6$ $b_5$ $b_4$ $b_3$ $b_2$ $b_1$ $b_0$	Only the bits written 0 are set to 0, whereas all bits written 1 are ignored.
9	0xb $_7$ FFFFFFF	0xb $_7$ $b_6$ $b_5$ $b_4$ $b_3$ $b_2$ $b_1$ $b_0$	Only the bits written 0 are set to 0, whereas all bits written 1 are ignored.

## 6.2.3 DMA Flash Write

When using DMA write operations, the data to be written into flash is stored in the XDATA memory space (RAM or registers). A DMA channel is configured to read the data to be written from the memory source address and write this data to the flash write-data register (FWDATA) fixed destination address, with the DMA trigger event FLASH (TRIG[4:0] = 1 0010 in DMA configuration) enabled. Thus, the flash controller triggers a DMA transfer when the flash write-data register, FWDATA, is ready to receive new data. The DMA channel should be configured to perform single-mode, byte-size transfers with the source address set to start-of-data block and destination address to fixed FWDATA (note that the block size, LEN in configuration data, must be divisible by 4; otherwise, the last word is not written to the flash). High priority should also be ensured for the DMA channel, so it is not interrupted in the write process. If interrupted for more than 20  $\mu$ s, the write operation may time out, and the write bit, FCTL.WRITE, is set to 0.

When the DMA channel is armed, starting a flash write by setting FCTL.WRITE to 1 triggers the first DMA transfer (DMA and flash controller handle the reset of the transfer).

[Figure 6-1](#) shows an example of how a DMA channel is configured and how a DMA transfer is initiated to write a block of data from a location in XDATA to flash memory.



**Figure 6-1. Flash Write Using DMA**

### 6.2.4 CPU Flash Write

To write to the flash using the CPU, a program executing from SRAM must implement the steps outlined in the procedure described in [Section 6.2.1](#). Disable interrupts to ensure the operation does not time out.

### 6.3 Flash Page Erase

The flash page-erase operation sets all bits in the page to 1.

A page erase is initiated by setting `FCTL.ERASE` to 1. The page addressed by `FADDRH[7:1]` (CC2530, CC2531, CC2540, and CC2541) or `FADDRH[6:0]` (CC2533) is erased when a page erase is initiated. Note that if a page erase is initiated simultaneously with a page write, that is, `FCTL.WRITE` is set to 1, the page erase is performed before the page-write operation starts. The `FCTL.BUSY` bit can be polled to see when the page erase has completed.

Power mode 1, 2, or 3 must not be entered while erasing a page. Also, the system clock source (XOSC/RCOSC) must not be changed while erasing.

---

**NOTE:** If a flash page-erase operation is performed from within flash memory and the Watchdog Timer is enabled, a Watchdog Timer interval must be selected that is longer than 20 ms, the duration of the flash page-erase operation, so that the CPU can clear the Watchdog Timer.

---

### 6.3.1 Performing Flash Erase From Flash Memory

Note that while executing program code from within flash memory, when a flash erase or write operation is initiated the CPU stalls, and program execution resumes from the next instruction when the flash controller has completed the operation.

The following code example of how to erase one flash page in the CC2530 is given for use with the IAR compiler:

```
#include <ioCC2530.h>

unsigned char erase_page_num = 3; /* page number to erase, here: flash page #3 */

/* Erase one flash page */
EA = 0; /* disable interrupts */
while (FCTL & 0x80); /* poll FCTL.BUSY and wait until flash controller is ready */
FADDRH = erase_page_num << 1; /* select the flash page via FADDRH[7:1] bits */
FCTL |= 0x01; /* set FCTL.ERASE bit to start page erase */
while (FCTL & 0x80); /* optional: wait until flash write has completed (~20 ms) */
EA = 1; /* enable interrupts */
```

### 6.3.2 Different Flash Page Size on CC2533

The flash page size has been reduced from 2 KB (2048 bytes) on CC2530, CC2531, CC2540, and CC2541 to 1 KB (1024 bytes) on CC2533. When performing page-erase operations on the flash memory, the page to be erased is addressed with the register bits `FADDRH[6:0]` on CC2533 as opposed to `FADDRH[7:1]` on CC2530, CC2531, CC2540, and CC2541. The page-lock bits are still placed in the upper 16 bytes of the last accessible flash page.

## 6.4 Flash DMA Trigger

The flash DMA trigger is activated when flash data written to the `FWDATA` register has been written to the specified location in the flash memory, thus indicating that the flash controller is ready to accept new data to be written to `FWDATA`. Four trigger pulses are generated. In order to start the first transfer, one must set the `FCTL.WRITE` bit to 1. The DMA and the flash controller then handle all transfers automatically for the defined block of data (`LEN` in DMA configuration). It is further important that the DMA is armed prior to setting the `FCTL.WRITE` bit, that the trigger source is set to `FLASH` (`TRIG[4:0] = 1 0010`), and that the DMA has high priority so the transfer is not interrupted. If interrupted for more than 20  $\mu$ s, the write operation times out and `FCTL.WRITE` bit is cleared.

## 6.5 Flash Controller Registers

The flash controller registers are described in this section.

**FCTL (0x6270) – Flash Control**

Bit	Name	Reset	R/W	Description
7	BUSY	0	R	Indicates that write or erase is in operation. This flag is set when the WRITE or ERASE bit is set. 0: No write or erase operation active 1: Write or erase operation activated
6	FULL		R/H0	Write buffer-full status. This flag is set when 4 bytes have been written to FWDATA during flash write. The write buffer is then full and does not accept more data; that is, writes to FWDATA are ignored when the FULL flag is set. The FULL flag is cleared when the write buffer again is ready to receive 4 more bytes. This flag is only needed when the CPU is used to write to the flash. 0: Write buffer can accept more data. 1: Write buffer full
5	ABORT	0	R/H0	Abort status. This bit is set when a write operation or page erase is aborted. An operation is aborted when the page accessed is locked. The abort bit is cleared when a write or page erase is started.
4	–	0	R	Reserved
3:2	CM[1:0]	01	R/W	Cache mode 00: Cache disabled 01: Cache enabled 10: Cache enabled, prefetch mode 11: Cache enabled, real-time mode  Cache mode. Disabling the cache increases the power consumption and reduces performance. Prefetching, for most applications, improves performance by up to 33% at the expense of potentially increased power consumption. Real-time mode provides predictable flash-read access time; the execution time is equal to that in cache-disabled mode, but the power consumption is lower.  <i>Note: The value read always represents the current cache mode. Writing a new cache mode starts a cache mode-change request that may take several clock cycles to complete. Writing to this register is ignored if there is a current cache-change request in progress.</i>
1	WRITE	0	R/W1/H0	Write. Start writing word at location given by FADDRH:FADDRL. The WRITE bit stays at 1 until the write completes. The clearing of this bit indicates that the erase has completed, that is, it has timed out or aborted.  If ERASE is also set to 1, a page erase of the whole page addressed by FADDRH[7:1] is performed before the write. Setting WRITE to 1 when ERASE is 1 has no effect.
0	ERASE	0	R/W1/H0	Page erase. Erase the page that is given by FADDRH[7:1] (CC2530, CC2531, CC2540, and CC2541) or FADDRH[6:0] (CC2533). The ERASE bit stays at 1 until the erase completes. The clearing of this bit indicates that the erase has completed successfully or aborted.  Setting ERASE to 1 when WRITE is 1 has no effect.

**FWDATA (0x6273) – Flash Write Data**

Bit	Name	Reset	R/W	Description
7:0	FWDATA[7:0]	0x00	R0/W	Flash write data. This register can only be written to when FCTL.WRITE is 1.

**FADDRH (0x6272) – Flash-Address High Byte**

Bit	Name	Reset	R/W	Description
7:0	FADDRH[7:0]	0x00	R/W	Page address and high byte of flash word address Bits [7:1] (CC2530, CC2531, CC2540, and CC2541) or bits [6:0] (CC2533) select which page to access.

**FADDRL (0x6271) – Flash-Address Low Byte**

Bit	Name	Reset	R/W	Description
7:0	FADDRL[7:0]	0x00	R/W	Low byte of flash word address

## I/O Ports

There are 21 digital input/output pins that can be configured as general-purpose digital I/O or as peripheral I/O signals connected to the ADC, timers, or USART peripherals. The use of the I/O ports is fully configurable from user software through a set of configuration registers.

The I/O ports have the following key features:

- 21 digital input/output pins
- General-purpose I/O or peripheral I/O
- Pullup or pulldown capability on inputs
- External interrupt capability

The external interrupt capability is available on all 21 I/O pins. Thus, external devices may generate interrupts if required. The external interrupt feature can also be used to wake the device up from sleep mode (power modes PM1, PM2, and PM3).

Topic	Page
<b>7.1 Unused I/O Pins .....</b>	<b>79</b>
<b>7.2 Low I/O Supply Voltage .....</b>	<b>79</b>
<b>7.3 General-Purpose I/O .....</b>	<b>79</b>
<b>7.4 General-Purpose I/O Interrupts .....</b>	<b>79</b>
<b>7.5 General-Purpose I/O DMA .....</b>	<b>80</b>
<b>7.6 Peripheral I/O .....</b>	<b>80</b>
<b>7.7 Debug Interface.....</b>	<b>83</b>
<b>7.8 32-kHz XOSC Input .....</b>	<b>83</b>
<b>7.9 Radio Test Output Signals.....</b>	<b>84</b>
<b>7.10 Power-Down Signal MUX (PMUX) .....</b>	<b>84</b>
<b>7.11 I/O Registers.....</b>	<b>84</b>

## 7.1 Unused I/O Pins

Unused I/O pins should have a defined level and not be left floating. One way to do this is to leave the pin unconnected and configure the pin as a general-purpose I/O input with pullup resistor. This is also the state of all pins during and after reset (except P1.0 and P1.1, which do not have pullup or pulldown capability). Alternatively, the pin can be configured as a general-purpose I/O output. In either case, the pin should not be connected directly to VDD or GND, in order to avoid excessive power consumption.

## 7.2 Low I/O Supply Voltage

In applications where the digital I/O power supply voltage pins, DVDD1 and DVDD2, are below 2.6 V, the register bit `PIC1L.PADSC` should be set to 1 in order to obtain the output dc characteristics specified in the *DC Characteristics* table in the device data sheet ([Appendix C](#)).

## 7.3 General-Purpose I/O

When used as general-purpose I/O, the pins are organized as three 8-bit ports, Port 0, Port 1, and Port 2; denoted P0, P1, and P2. P0 and P1 are complete 8-bit-wide ports, whereas P2 has only five usable bits. All ports are both bit- and byte-addressable through the SFR registers `P0`, `P1`, and `P2`. Each port pin can individually be set to operate as a general-purpose I/O or as a peripheral I/O.

The output drive strength is 4 mA on all outputs, except for the two high-drive outputs, P1.0 and P1.1, which each have 20-mA output drive strength.

The registers `PxSEL`, where `x` is the port number 0–2, are used to configure each pin in a port as either a general-purpose I/O pin or as a peripheral I/O signal. By default, after a reset, all digital input/output pins are configured as general-purpose input pins.

To change the direction of a port pin, the registers `PxDIR` are used to set each port pin to be either an input or an output. Thus, by setting the appropriate bit within `PxDIR` to 1, the corresponding pin becomes an output.

When reading the port registers `P0`, `P1`, and `P2`, the logic values on the input pins are returned regardless of the pin configuration. This does not apply during the execution of read-modify-write instructions. The read-modify-write instructions are: `ANL`, `ORL`, `XRL`, `JBC`, `CPL`, `INC`, `DEC`, `DJNZ`, `MOV`, `CLR`, and `SETB`.

Operating on a port register, the following is true: When the destination is an individual bit in port register `P0`, `P1`, or `P2`, the value of the register, not the value on the pin, is read, modified, and written back to the port register.

When used as an input, the general-purpose I/O port pins can be configured to have a pullup, pulldown or three-state mode of operation. By default, after a reset, inputs are configured as inputs with pullup. To deselect the pullup or pulldown function on an input, the appropriate bit within the `PxINP` must be set to 1. The I/O port pins P1.0 and P1.1 do not have pullup or pulldown capability. Note that pins configured as peripheral I/O signals do not have pullup or pulldown capability, even if the peripheral function is an input.

In power modes PM1, PM2, and PM3, the I/O pins retain the I/O mode and output value (if applicable) that was set when PM1, PM2, or PM3 was entered.

## 7.4 General-Purpose I/O Interrupts

General-purpose I/O pins configured as inputs can be used to generate interrupts. The interrupts can be configured to trigger on either a rising or falling edge of the external signal. Each of the `P0`, `P1`, and `P2` ports has port interrupt-enable bits common for all bits within the port located in the `IEN1–IEN2` registers as follows:

- `IEN1.P0IE`: P0 interrupt enable
- `IEN2.P1IE`: P1 interrupt enable
- `IEN2.P2IE`: P2 interrupt enable

In addition to these common interrupt enables, the bits within each port have individual interrupt enables located in SFR registers `P0IEN`, `P1IEN`, and `P2IEN`. Even I/O pins configured as peripheral I/O or general-purpose outputs have interrupts generated when enabled.

When an interrupt condition occurs on one of the I/O pins, the interrupt status flag in the corresponding P0–P2 interrupt flag register, P0IFG, P1IFG, or P2IFG, is set to 1. The interrupt status flag is set regardless of whether the pin has its interrupt enable set. When an interrupt is serviced, the interrupt status flag is cleared by writing a 0 to that flag. This flag must be cleared prior to clearing the CPU port interrupt flag (PxIF). This is illustrated in [Figure 2-4](#): There is an edge detect between the input line and PxIFG, but no edge detect or one-shot between PxIFG and PxINT. The practical impact of this is what is written in [Section 2.5.1](#)

The SFR registers used for interrupts are described later in this section. The registers are summarized as follows:

- P0IEN: P0 interrupt enables
- P1IEN: P1 interrupt enables
- P2IEN: P2 interrupt enables
- PICTL: P0, P1, and P2 edge configuration
- P0IFG: P0 interrupt flags
- P1IFG: P1 interrupt flags
- P2IFG: P2 interrupt flags

## 7.5 General-Purpose I/O DMA

When used as general-purpose I/O pins, the P0 and P1 ports are each associated with one DMA trigger. These DMA triggers are IOC\_0 for P0 and IOC\_1 for P1, as shown in [Table 8-1](#).

The IOC\_0 trigger is activated when an interrupt occurs on the P0 pins. The IOC\_1 trigger is activated when an interrupt occurs on the P1 pins.

## 7.6 Peripheral I/O

This section describes how the digital I/O pins are configured as peripheral I/Os. For each peripheral unit that can interface with an external system through the digital input/output pins, a description of how peripheral I/Os are configured is given in the following subsections.

For USART and timer I/O, setting the appropriate P<sub>x</sub>SEL bits to 1 is required for the output signals on a digital I/O pin to be controlled by the peripheral. For peripheral inputs from digital I/O pins, this is optional. P<sub>x</sub>SEL = 1 overrides the pullup or pulldown setting of a pin, so to be able to control pullup and pulldown with the P<sub>x</sub>INP bits, the P<sub>x</sub>SEL bit should be set to 0 for that pin.

Note that peripheral units have two alternative locations for their I/O pins; see [Table 7-1](#). Priority can be set between peripherals if conflicting settings regarding I/O mapping are present (using the P2SEL.PRIxP1 and P2DIR.PRIp0 bits). All combinations not causing conflicts can be used.

Note that a peripheral normally is present at the selected location even if it is not used, and another peripheral that is to use the pins must be given higher priority. The exception is the RTS and CTS pins of a USART in UART mode with flow control disabled and the SSN pin of a USART configured in SPI master mode.

Note also that peripheral units that have input pins receive an input from the pin regardless of the P<sub>x</sub>INP setting, and this may influence on the state of the peripheral unit. For instance, a UART should be flushed before use if there may have been activity on the RX pin prior to taking it in use as a UART pin.



**Table 7-1. Peripheral I/O Pin Mapping**

Periphery/ Function	P0								P1								P2				
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	4	3	2	1	0
ADC	A7	A6	A5	A4	A3	A2	A1	A0													T
Operational amplifier						O	-	+													
Analog comparator			+	-																	
USART 0 SPI			C	SS	MO	MI															
Alt. 2											M0	MI	C	SS							
USART 0 UART			RT	CT	TX	RX															
Alt. 2											TX	RX	RT	CT							
USART 1 SPI			MI	M0	C	SS															
Alt. 2									MI	M0	C	SS									
USART 1 UART			RX	TX	RT	CT															
Alt. 2									RX	TX	RT	CT									
TIMER 1		4	3	2	1	0															
Alt. 2	3	4												0	1	2					
TIMER 3												1	0								
Alt. 2									1	0											
TIMER 4															1	0					
Alt. 2																		1			0
32-kHz XOSC																	Q1	Q2			
DEBUG																			DC	DD	
OBSSEL										5	4	3	2	1	0						

### 7.6.1 Timer 1

PERCFG.T1CFG selects whether to use alternative 1 or alternative 2 locations.

In [Table 7-1](#), the Timer 1 signals are shown as the following:

- 0: Channel 0 capture or compare pin
- 1: Channel 1 capture or compare pin
- 2: Channel 2 capture or compare pin
- 3: Channel 3 capture or compare pin
- 4: Channel 4 capture or compare pin

P2DIR.PRI0 selects the order of precedence when assigning several peripherals to Port 0. When set to 10, Timer 1 channels 0–1 have precedence, and when set to 11, Timer 1 channels 2–3 have precedence. To have all Timer 1 channels visible in the alternative 1 location, move both USART 0 and USART 1 to the alternative 2 location.

P2SEL.PRI1P1 and P2SEL.PRI0P1 select the order of precedence when assigning several peripherals to Port 1. The Timer 1 channels have precedence when the former is set low and the latter is set high.

### 7.6.2 Timer 3

PERCFG.T3CFG selects whether to use alternative 1 or alternative 2 locations.

In [Table 7-1](#), the Timer 3 signals are shown as the following:

- 0: Channel 0 capture or compare pin
- 1: Channel 1 capture or compare pin

`P2SEL.PRI2P1` and `P2SEL.PRI3P1` select the order of precedence when assigning several peripherals to Port 1. The Timer 3 channels have precedence when both bits are set high. If `P2SEL.PRI2P1` is set high and `P2SEL.PRI3P1` is set low, the Timer 3 channels have precedence over USART 1, but USART 0 has precedence over the Timer 3 channels as well as over USART 1.

### 7.6.3 Timer 4

`PERCFG.T4CFG` selects whether to use alternative 1 or alternative 2 locations.

In [Table 7-1](#), the Timer 4 signals are shown as the following:

- 0: Channel 0 capture or compare pin
- 1: Channel 1 capture or compare pin

`P2SEL.PRI1P1` selects the order of precedence when assigning several peripherals to Port 1. The Timer 4 channels have precedence when the bit is set.

### 7.6.4 USART 0

The SFR register bit `PERCFG.U0CFG` selects whether to use alternative 1 or alternative 2 locations.

In [Table 7-1](#), the USART 0 signals are shown as follows:

UART:

- RX: RXDATA
- TX: TXDATA
- RT: RTS
- CT: CTS

SPI:

- MI: MISO
- MO: MOSI
- C: SCK
- SS: SSN

`P2DIR.PRIP0` selects the order of precedence when assigning several peripherals to Port 0. When set to 00, USART 0 has precedence. Note that if UART mode is selected and hardware flow control is disabled, USART 1 or Timer 1 has precedence to use ports P0.4 and P0.5.

`P2SEL.PRI3P1` and `P2SEL.PRI0P1` select the order of precedence when assigning several peripherals to Port 1. USART 0 has precedence when both are set to 0. Note that if UART mode is selected and hardware flow control is disabled, Timer 1 or Timer 3 has precedence to use ports P1.2 and P1.3.

### 7.6.5 USART 1

The SFR register bit `PERCFG.U1CFG` selects whether to use alternative 1 or alternative 2 locations.

In [Table 7-1](#), the USART 1 signals are shown as follows:

UART:

- RX: RXDATA
- TX: TXDATA
- RT: RTS
- CT: CTS

SPI:

- MI: MISO
- MO: MOSI
- C: SCK
- SS: SSN

`P2DIR.PRIP0` selects the order of precedence when assigning several peripherals to Port 0. When set to 01, USART 1 has precedence. Note that if UART mode is selected and hardware flow control is disabled, USART 0 or Timer 1 has precedence to use ports P0.2 and P0.3.

`P2SEL.PRI3P1` and `P2SEL.PRI2P1` select the order of precedence when assigning several peripherals to Port 1. USART 1 has precedence when the former is set to 1 and the latter is set to 0. Note that if UART mode is selected and hardware flow control is disabled, USART 0 or Timer 3 has precedence to use ports P1.4 and P1.5.

### 7.6.6 ADC

In [Table 7-1](#), the ADC signals are shown as follows:

- A0: ADC input 0
- A1: ADC input 1
- A2: ADC input 2
- A3: ADC input 3
- A4: ADC input 4
- A5: ADC input 5
- A6: ADC input 6
- A7: ADC input 7
- T: ADC external trigger pin

When using the ADC, Port 0 pins must be configured as ADC inputs. Up to eight ADC inputs can be used. To configure a Port 0 pin to be used as an ADC input, the corresponding bit in the `APCFG` register must be set to 1. The default values in this register select the Port 0 pins as non-ADC input, i.e., digital input/outputs.

The settings in the `APCFG` register override the settings in `P0SEL`.

The ADC can be configured to use the general-purpose I/O pin P2.0 as an external trigger to start conversions. P2.0 must be configured as a general-purpose I/O in input mode when being used for ADC external trigger.

### 7.6.7 Operational Amplifier and Analog Comparator

When using the operational amplifier and analog comparator, the corresponding Port 0 pins must be configured as ADC inputs (see [Table 7-1](#)). To configure a Port 0 pin to be used as an ADC input, the corresponding bit in the `APCFG` register must be set to 1. The default values in this register select the Port 0 pins as non-ADC input, that is, digital input/outputs.

The settings in the `APCFG` register override the settings in `P0SEL`.

## 7.7 Debug Interface

Ports P2.1 and P2.2 are used for debug data and clock signals, respectively. These are shown as DD (debug data) and DC (debug clock) in [Table 7-1](#). When in debug mode, the debug interface controls the direction of these pins. Pullup and pulldown are disabled on these pins while in debug mode.

## 7.8 32-kHz XOSC Input

Ports P2.3 and P2.4 can be used to connect an external 32-kHz crystal. These port pins are used by the 32-kHz XOSC when `CLKCONCMD.OSC32K` is low, regardless of register settings. The port pins are set in analog mode when `CLKCONCMD.OSC32K` is low.

## 7.9 Radio Test Output Signals

By using the OBSSEL<sub>x</sub> registers (OBSSEL0–OBSSEL5) the user can output different signals from the RF Core to GPIO pins. These signals can be useful for debugging of low-level protocols or control of external PA, LNA, or switches. The control registers OBSSEL0–OBSSEL5 can be used to override the standard GPIO behavior and output RF Core signals (`rfc_obs_sig0`, `rfc_obs_sig1`, and `rfc_obs_sig2`) on the pins P1[0:5]. For a list of available signals, see the respective RFC\_OBS\_CTRL<sub>x</sub> registers in [Section 23.15.3](#) for CC253x or [Section 24.1](#) for CC2540 or [Chapter 25](#) for CC2541.

## 7.10 Power-Down Signal MUX (PMUX)

The PMUX register can be used to output the 32-kHz clock and/or the digital voltage regulator status.

The selected 32-kHz clock source can be output on one of the P0 pins. The enable bit CKOEN enables the output on P0, and the pin of P0 is selected using the CKOPIN (see the PMUX register description for details). When CKOEN is set, all other configurations for the selected pin are overridden. The clock is output in all power modes; however, in PM3 the clock stops (see PM3 in [Chapter 4](#)).

Furthermore, the digital voltage regulator status can be output on one of the P1 pins. When the DREGSTA bit is set, the status of the digital voltage regulator is output. DREGSTAPIN selects the P1 pin (see the PMUX register description for details). When DREGSTA is set, all other configurations for the selected pin are overridden. The selected pin outputs 1 when the 1.8-V on-chip digital voltage regulator is powered up (chip has regulated power). The selected pin outputs 0 when the 1.8-V on-chip digital voltage regulator is powered down, that is, in PM2 and PM3.

## 7.11 I/O Registers

The registers for the I/O ports are described in this section. The registers are:

- P0: Port 0
- P1: Port 1
- P2: Port 2
- PERCFG: Peripheral-control register
- APCFG: Analog peripheral I/O configuration
- POSEL: Port 0 function-select register
- P1SEL: Port 1 function-select register
- P2SEL: Port 2 function-select register
- P0DIR: Port 0 direction register
- P1DIR: Port 1 direction register
- P2DIR: Port 2 direction register
- P0INP: Port 0 input-mode register
- P1INP: Port 1 input-mode register
- P2INP: Port 2 input-mode register
- P0IFG: Port 0 interrupt-status flag register
- P1IFG: Port 1 interrupt-status flag register
- P2IFG: Port 2 interrupt-status flag register
- PICTL: Interrupt edge register
- P0IEN: Port 0 interrupt-mask register
- P1IEN: Port 1 interrupt-mask register
- P2IEN: Port 2 interrupt-mask register
- PMUX: Power-down signal-mux register
- OBSSEL0: Observation output control register 0
- OBSSEL1: Observation output control register 1
- OBSSEL2: Observation output control register 2

- OBSSEL3: Observation output control register 3
- OBSSEL4: Observation output control register 4
- OBSSEL5: Observation output control register 5

**P0 (0x80) – Port 0**

Bit	Name	Reset	R/W	Description
7:0	P0[7:0]	0xFF	R/W	Port 0. General-purpose I/O port. Bit-addressable from SFR. This CPU-internal register is readable, but not writable, from XDATA (0x7080).

**P1 (0x90) – Port 1**

Bit	Name	Reset	R/W	Description
7:0	P1[7:0]	0xFF	R/W	Port 1. General-purpose I/O port. Bit-addressable from SFR. This CPU-internal register is readable, but not writable, from XDATA (0x7090).

**P2 (0xA0) – Port 2**

Bit	Name	Reset	R/W	Description
7:5	–	000	R0	Reserved
4:0	P2[4:0]	1 1111	R/W	Port 2. General-purpose I/O port. Bit-addressable from SFR. This CPU-internal register is readable, but not writable, from XDATA (0x70A0).

**PERCFG (0xF1) – Peripheral Control**

Bit	Name	Reset	R/W	Description
7	–	0	R0	Reserved
6	T1CFG	0	R/W	Timer 1 I/O location 0: Alternative 1 location 1: Alternative 2 location
5	T3CFG	0	R/W	Timer 3 I/O location 0: Alternative 1 location 1: Alternative 2 location
4	T4CFG	0	R/W	Timer 4 I/O location 0: Alternative 1 location 1: Alternative 2 location
3:2	–	00	R/W	Reserved
1	U1CFG	0	R/W	USART 1 I/O location 0: Alternative 1 location 1: Alternative 2 location
0	U0CFG	0	R/W	USART 0 I/O location 0: Alternative 1 location 1: Alternative 2 location

**APCFG (0xF2) – Analog Peripheral I/O Configuration**

Bit	Name	Reset	R/W	Description
7:0	APCFG[7:0]	0x00	R/W	Analog Peripheral I/O configuration . APCFG[7:0] select P0.7–P0.0 as analog I/O. 0: Analog I/O disabled 1: Analog I/O enabled

**P0SEL (0xF3) – Port 0 Function Select**

Bit	Name	Reset	R/W	Description
7:0	SELP0_[7:0]	0x00	R/W	P0.7 to P0.0 function select 0: General-purpose I/O 1: Peripheral function

**P1SEL (0xF4) – Port 1-Function Select**

Bit	Name	Reset	R/W	Description
7:0	SELP1_[7:0]	0x00	R/W	P1.7 to P1.0 function select 0: General-purpose I/O 1: Peripheral function

**P2SEL (0xF5) – Port 2 Function Select and Port 1 Peripheral Priority Control**

Bit	Name	Reset	R/W	Description
7	–	0	R0	Reserved
6	PRI3P1	0	R/W	Port 1 peripheral priority control. This bit determines which module has priority in the case when modules are assigned to the same pins. 0: USART 0 has priority. 1: USART 1 has priority.
5	PRI2P1	0	R/W	Port 1 peripheral priority control. This bit determines the order of priority in the case when PERCFG assigns USART 1 and Timer 3 to the same pins. 0: USART 1 has priority. 1: Timer 3 has priority.
4	PRI1P1	0	R/W	Port 1 peripheral priority control. This bit determines the order of priority in the case when PERCFG assigns Timer 1 and Timer 4 to the same pins. 0: Timer 1 has priority. 1: Timer 4 has priority.
3	PRI0P1	0	R/W	Port 1 peripheral priority control. This bit determines the order of priority in the case when PERCFG assigns USART 0 and Timer 1 to the same pins. 0: USART 0 has priority. 1: Timer 1 has priority.
2	SELP2_4	0	R/W	P2.4 function select 0: General-purpose I/O 1: Peripheral function
1	SELP2_3	0	R/W	P2.3 function select 0: General-purpose I/O 1: Peripheral function
0	SELP2_0	0	R/W	P2.0 function select 0: General-purpose I/O 1: Peripheral function

**P0DIR (0xFD) – Port 0 Direction**

Bit	Name	Reset	R/W	Description
7:0	DIRP0_[7:0]	0x00	R/W	P0.7 to P0.0 I/O direction 0: Input 1: Output

**P1DIR (0xFE) – Port 1 Direction**

Bit	Name	Reset	R/W	Description
7:0	DIRP1_[7:0]	0x00	R/W	P1.7 to P1.0 I/O direction 0: Input 1: Output

**P2DIR (0xFF) – Port 2 Direction and Port 0 Peripheral Priority Control**

Bit	Name	Reset	R/W	Description
7:6	PRIP0[1:0]	00	R/W	Port 0 peripheral priority control. These bits determine the order of priority in the case when PERCFG assigns several peripherals to the same pins.  Detailed priority list:  00: 1st priority: USART 0 2nd priority: USART 1 3rd priority: Timer 1  01: 1st priority: USART 1 2nd priority: USART 0 3rd priority: Timer 1  10: 1st priority: Timer 1 channels 0–1 2nd priority: USART 1 3rd priority: USART 0 4th priority: Timer 1 channels 2–3  11: 1st priority: Timer 1 channels 2–3 2nd priority: USART 0 3rd priority: USART 1 4th priority: Timer 1 channels 0–1
5	–	0	R0	Reserved
4:0	DIRP2_[4:0]	0 0000	R/W	P2.4 to P2.0 I/O direction  0: Input 1: Output

**P0INP (0x8F) – Port 0 Input Mode**

Bit	Name	Reset	R/W	Description
7:0	MDP0_[7:0]	0x00	R/W	P0.7 to P0.0 I/O input mode  0: Pullup or pulldown [see P2INP (0xF7) – Port 2 input mode] 1: 3-state

**P1INP (0xF6) – Port 1 Input Mode**

Bit	Name	Reset	R/W	Description
7:2	MDP1_[7:2]	0000 00	R/W	P1.7 to P1.2 I/O input mode  0: Pullup or pulldown [see P2INP (0xF7) – Port 2 input mode] 1: 3-state
1:0	–	00	R0	Reserved

**P2INP (0xF7) – Port 2 Input Mode**

Bit	Name	Reset	R/W	Description
7	PDUP2	0	R/W	Port 2 pullup-or-pulldown select. Selects function for all Port 2 pins configured as pullup-or-pulldown inputs. 0: Pullup 1: Pulldown
6	PDUP1	0	R/W	Port 1 pullup-or-pulldown select. Selects function for all Port 1 pins configured as pullup-or-pulldown inputs. 0: Pullup 1: Pulldown
5	PDUP0	0	R/W	Port 0 pullup-or-pulldown select. Selects function for all Port 0 pins configured as pullup-or-pulldown inputs. 0: Pullup 1: Pulldown
4:0	MDP2_[4:0]	0 0000	R/W	P2.4 to P2.0 I/O input mode 0: Pullup or pulldown 1: 3-state

**P0IFG (0x89) – Port 0 Interrupt Status Flag**

Bit	Name	Reset	R/W	Description
7:0	P0IF[7:0]	0x00	R/W0	Port 0, inputs 7 to 0 interrupt status flags. When an input port pin has an interrupt request pending, the corresponding flag bit is set.

**P1IFG (0x8A) – Port 1 Interrupt Status Flag**

Bit	Name	Reset	R/W	Description
7:0	P1IF[7:0]	0x00	R/W0	Port 1, inputs 7 to 0 interrupt status flags. When an input port pin has an interrupt request pending, the corresponding flag bit is set.

**P2IFG (0x8B) – Port 2 Interrupt Status Flag**

Bit	Name	Reset	R/W	Description
7:6	–	00	R0	Reserved
5	DPIF	0	R/W0	USB D+ interrupt-status flag. This flag is set when the D+ line has an interrupt request pending and is used to detect USB resume events in USB suspend state. This flag is not set when the USB controller is not suspended.
4:0	P2IF[4:0]	0 0000	R/W0	Port 2, inputs 4 to 0 interrupt status flags. When an input port pin has an interrupt request pending, the corresponding flag bit is set.



**PICTL (0x8C) – Port Interrupt Control**

Bit	Name	Reset	R/W	Description
7	PADSC	0	R/W	Drive strength control for I/O pins in output mode. Selects output drive strength enhancement to account for low I/O supply voltage on pin DVDD (this to ensure the same drive strength at lower voltages as at higher). 0: Minimum drive strength enhancement. DVDD1 and DVDD2 equal to or greater than 2.6 V 1: Maximum drive strength enhancement. DVDD1 and DVDD2 less than 2.6 V
6:4	–	000	R0	Reserved
3	P2ICON	0	R/W	Port 2, inputs 4 to 0 interrupt configuration. This bit selects the interrupt request condition for Port 2 inputs 4 to 0. 0: Rising edge on input gives interrupt. 1: Falling edge on input gives interrupt.
2	P1ICONH	0	R/W	Port 1, inputs 7 to 4 interrupt configuration. This bit selects the interrupt request condition for the high nibble of Port 1 inputs. 0: Rising edge on input gives interrupt. 1: Falling edge on input gives interrupt
1	P1ICONL	0	R/W	Port 1, inputs 3 to 0 interrupt configuration. This bit selects the interrupt request condition for the low nibble of Port 1 inputs. 0: Rising edge on input gives interrupt. 1: Falling edge on input gives interrupt.
0	P0ICON	0	R/W	Port 0, inputs 7 to 0 interrupt configuration. This bit selects the interrupt request condition for all Port 0 inputs. 0: Rising edge on input gives interrupt. 1: Falling edge on input gives interrupt.

**P0IEN (0xAB) – Port 0 Interrupt Mask**

Bit	Name	Reset	R/W	Description
7:0	P0_[7:0]IEN	0x00	R/W	Port P0.7 to P0.0 interrupt enable 0: Interrupts are disabled. 1: Interrupts are enabled.

**P1IEN (0x8D) – Port 1 Interrupt Mask**

Bit	Name	Reset	R/W	Description
7:0	P1_[7:0]IEN	0x00	R/W	Port P1.7 to P1.0 interrupt enable 0: Interrupts are disabled. 1: Interrupts are enabled.

**P2IEN (0xAC) – Port 2 Interrupt Mask**

Bit	Name	Reset	R/W	Description
7:6	–	00	R0	Reserved
5	DPIEN	0	R/W	USB D+ interrupt enable 0: USB D+ interrupt disabled 1: USB D+ interrupt enabled
4:0	P2_[4:0]IEN	0 0000	R/W	Port P2.4 to P2.0 interrupt enable 0: Interrupts are disabled. 1: Interrupts are enabled.

**PMUX (0xAE) – Power-Down Signal Mux**

Bit	Name	Reset	R/W	Description
7	CKOEN	0	R/W	Clock Out Enable. When this bit is set, the selected 32-kHz clock is output on one of the P0 pins. CKOPIN selects the pin to use. This overrides all other configurations for the selected pin. The clock is output in all power modes; however, in PM3 the clock stops (see PM3 in <a href="#">Chapter 4</a> ).
6:4	CKOPIN[2:0]	000	R/W	Clock Out Pin. Selects which P0 pin is to be used to output the selected 32-kHz clock.
3	DREGSTA	0	R/W	Digital Voltage Regulator Status. When this bit is set, the status of the digital voltage regulator is output on one of the P1 pins. DREGSTAPIN selects the pin. When DREGSTA is set, all other configurations for the selected pin are overridden. The selected pin outputs 1 when the 1.8-V on-chip digital voltage regulator is powered up (chip has regulated power). The selected pin outputs 0 when the 1.8-V on-chip digital voltage regulator is powered down.
2:0	DREGSTAPIN[2:0]	000	R/W	Digital Voltage Regulator Status Pin. Selects which P1 pin is to be used to output the DREGSTA signal.

Note that registers OBSSEL0 through OBSSEL5 do not retain data in states PM2 and PM3.

**OBSSEL0 (0x6243) – Observation Output Control Register 0**

Bit	Name	Reset	R/W	Description
7	EN	0	R/W	Bit controlling the observation output 0 on P1[0]. 0 – Observation output disabled 1 – Observation output enabled Note: If enabled, this overwrites the standard GPIO behavior of P1.0.
6:0	SEL[6:0]	000 0000	R/W	Select output signal on observation output 0 111 1011 (123): rfc_obs_sig0 111 1100 (124): rfc_obs_sig1 111 1101 (125): rfc_obs_sig2 Others: Reserved

**OBSSEL1 (0x6244) – Observation Output Control Register 1**

Bit	Name	Reset	R/W	Description
7	EN	0	R/W	Bit controlling observation output 1 on P1[1]. 0 – Observation output disabled 1 – Observation output enabled Note: If enabled, this overwrites the standard GPIO behavior of P1.1.
6:0	SEL[6:0]	000 0000	R/W	Select output signal on observation output 1 111 1011 (123): rfc_obs_sig0 111 1100 (124): rfc_obs_sig1 111 1101 (125): rfc_obs_sig2 Others: Reserved

**OBSSEL2 (0x6245) – Observation Output Control Register 2**

Bit	Name	Reset	R/W	Description
7	EN	0	R/W	Bit controlling observation output 2 on P1[2]. 0 – Observation output disabled 1 – Observation output enabled Note: If enabled, this overwrites the standard GPIO behavior of P1.2.
6:0	SEL[6:0]	000 0000	R/W	Select output signal on observation output 2 111 1011 (123): rfc_obs_sig0 111 1100 (124): rfc_obs_sig1 111 1101 (125): rfc_obs_sig2 Others: Reserved

**OBSSEL3 (0x6246) – Observation Output Control Register 3**

Bit	Name	Reset	R/W	Description
7	EN	0	R/W	Bit controlling observation output 3 on P1[3]. 0 – Observation output disabled 1 – Observation output enabled Note: If enabled, this overwrites the standard GPIO behavior of P1.3.
6:0	SEL[6:0]	000 0000	R/W	Select output signal on observation output 3 111 1011 (123): rfc_obs_sig0 111 1100 (124): rfc_obs_sig1 111 1101 (125): rfc_obs_sig2 Others: Reserved

**OBSSEL4 (0x6247) – Observation Output Control Register 4**

Bit	Name	Reset	R/W	Description
7	EN	0	R/W	Bit controlling observation output 4 on P1[4]. 0 – Observation output disabled 1 – Observation output enabled Note: If enabled, this overwrites the standard GPIO behavior of P1.4.
6:0	SEL[6:0]	000 0000	R/W	Select output signal on observation output 4 111 1011 (123): rfc_obs_sig0 111 1100 (124): rfc_obs_sig1 11 11101 (125): rfc_obs_sig2 Others: Reserved

**OBSSEL5 (0x6248) – Observation Output Control Register 5**

Bit	Name	Reset	R/W	Description
7	EN	0	R/W	Bit controlling the observation output 5 on P1[5]. 0 – Observation output disabled 1 – Observation output enabled Note: If enabled, this overwrites the standard GPIO behavior of P1.5.
6:0	SEL[6:0]	000 0000	R/W	Select output signal on observation output 5 111 1011 (123): rfc_obs_sig0 111 1100 (124): rfc_obs_sig1 111 1101 (125): rfc_obs_sig2 Others: Reserved

## DMA Controller

The Direct Memory Access (DMA) Controller can be used to relieve the 8051 CPU core of handling data movement operations, thus achieving high overall performance with good power efficiency. The DMA controller can move data from a peripheral unit such as ADC or RF transceiver to memory with minimum CPU intervention.

The DMA controller coordinates all DMA transfers, ensuring that DMA requests are prioritized appropriately relative to each other and to CPU memory access. The DMA controller contains a number of programmable DMA channels for memory-memory data movement.

The DMA controller controls data transfers over the entire address range in XDATA memory space. Because most of the SFR registers are mapped into the DMA memory space, these flexible DMA channels can be used to unburden the CPU in innovative ways, for example, to feed a USART with data from memory or periodically to transfer samples between ADC and memory, and so forth. Use of the DMA can also reduce system power consumption by keeping the CPU in a low-power mode without having to wake up to move data to or from a peripheral unit (see [Section 4.1.1](#) for CPU low-power mode). Note that [Section 2.2.3](#) describes the SFR registers that are not mapped into XDATA memory space.

The main features of the DMA controller are as follows:

- Five independent DMA channels
- Three configurable levels of DMA channel priority
- 32 configurable transfer trigger events
- Independent control of source and destination address
- Single, block and repeated transfer modes
- Supports length field in transfer data, setting variable transfer length
- Can operate in either word-size or byte-size mode

Topic	Page
<b>8.1 DMA Operation .....</b>	<b>93</b>
<b>8.2 DMA Configuration Parameters.....</b>	<b>95</b>
<b>8.3 DMA Configuration Setup.....</b>	<b>97</b>
<b>8.4 Stopping DMA Transfers .....</b>	<b>98</b>
<b>8.5 DMA Interrupts.....</b>	<b>98</b>
<b>8.6 DMA Configuration-Data Structure .....</b>	<b>98</b>
<b>8.7 DMA Memory Access.....</b>	<b>98</b>
<b>8.8 DMA Registers .....</b>	<b>101</b>

## 8.1 DMA Operation

There are five DMA channels available in the DMA controller, numbered channel 0 through channel 4. Each DMA channel can move data from one place within the DMA memory space to another, that is, between XDATA locations.

In order to use a DMA channel, it must first be configured as described in [Section 8.2](#) and [Section 8.3](#). [Figure 8-1](#) shows the DMA state diagram.

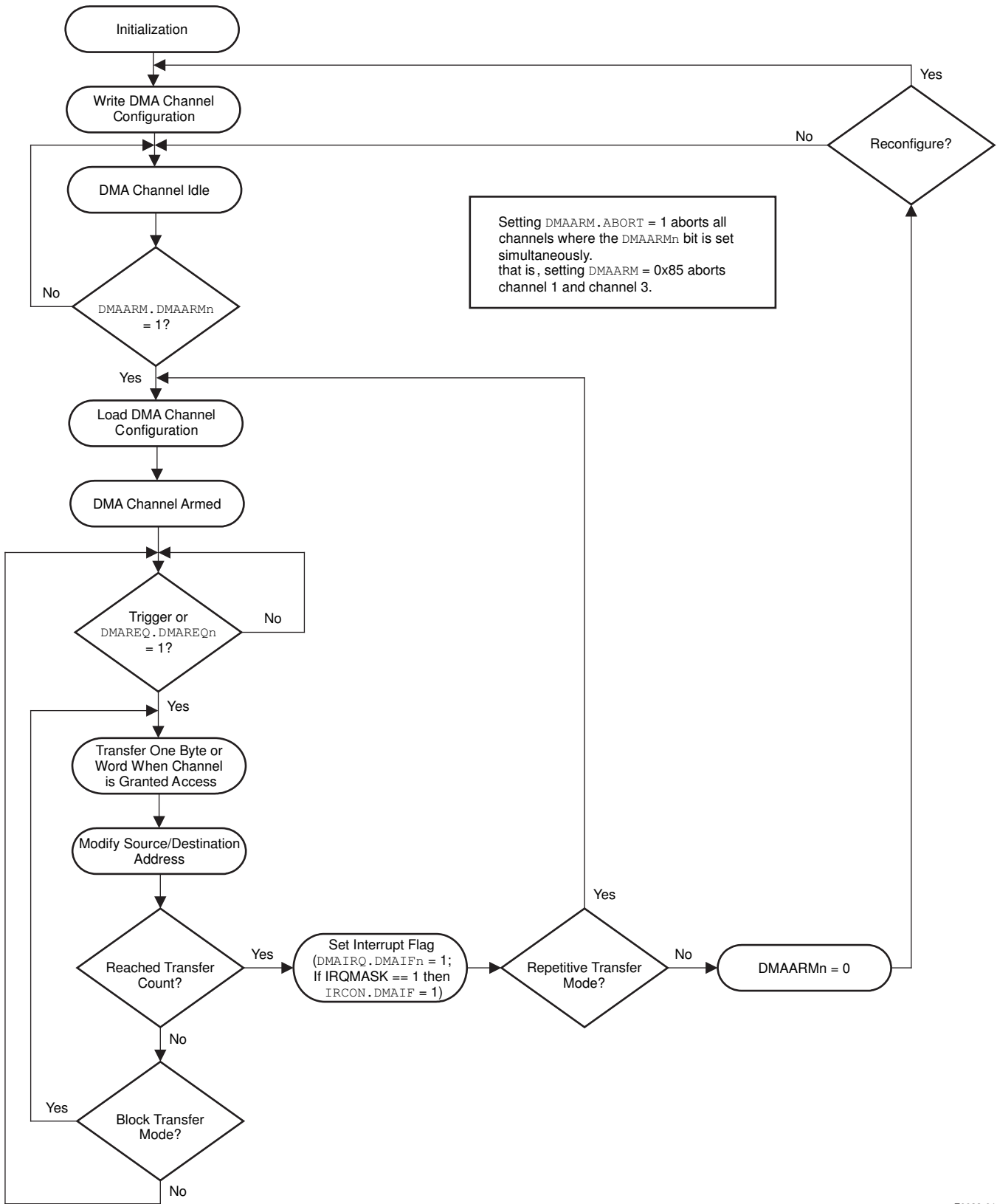
Once a DMA channel has been configured, it must be armed before any transfers are allowed to be initiated. A DMA channel is armed by setting the appropriate bit in the DMA channel-arm register `DMAARM`.

When a DMA channel is armed, a transfer begins when the configured DMA trigger event occurs. Note that the time to arm one channel (that is, get configuration data) takes nine system clocks; thus, if the corresponding `DMAARM` bit is set and a trigger appears within the time it takes to configure the channel, the wanted trigger is lost. If two or more DMA channels are armed simultaneously, the time for all channels to be configured is longer (sequential read from memory). If all five are armed, it takes 45 system clocks, and channel 1 is ready first, then channel 2, and lastly channel 0 (all within the last eight system clocks). There are 32 possible DMA trigger events (see [Table 8-1](#)), for example, UART transfer, timer overflow. The trigger event to be used by a DMA channel is set by the DMA channel configuration; thus, no knowledge of this is available until after the configuration has been read. The DMA trigger events are listed in [Table 8-1](#).

In addition to starting a DMA transfer through the DMA trigger events, the user software may force a DMA transfer to begin by setting the corresponding `DMAREQ` bit.

It should be noted that if the previously configured trigger source generates trigger events while DMA is being configured, these are counted as missed events, and as soon as the DMA channel is ready, the transfer is started. This occurs even though the new trigger source is not the same as the previous one. In some situations, this leads to errors in the transfer. In order to account for this, trigger source 0 should be the source between reconfigurations. This is achieved by setting up dummy source and destination addresses, using fixed length of one byte, block transfer, and trigger source 0. Enabling a software trigger (`DMAREQ`) clears missed-trigger counting, and no new triggers are generated while a new configuration is fetched from memory (unless software writes to `DMAREQ` for this channel).

A `DMAREQ` bit is cleared only when the corresponding DMA transfer occurs. The `DMAREQ` bit is not cleared when the channel is disarmed.



F0033-01

Figure 8-1. DMA Operation

## 8.2 DMA Configuration Parameters

Setup and control of the DMA operation is performed by the user software. This section describes the parameters which must be configured before a DMA channel can be used. [Section 8.3](#) describes how the parameters are set up in software and passed to the DMA controller.

The behavior of each of the five DMA channels is configured with the following parameters:

**Source address:** The first address from which the DMA channel should read data.

**Destination address:** The first address to which the DMA channel should write the data read from the source address. The user must ensure that the destination is writable.

**Transfer count:** The number of transfers to perform before rearming or disarming the DMA channel and alerting the CPU with an interrupt request. The length can be defined in the configuration or it can be defined as described next for the VLEN setting.

**VLEN setting:** The DMA channel is capable of variable-length transfers, using the first byte or word to set the transfer length. When doing this, various options are available regarding how to count the number of bytes to transfer.

**Priority:** The priority of the DMA transfers for the DMA channel with respect to the CPU and other DMA channels and access ports.

**Trigger event:** All DMA transfers are initiated by so-called DMA trigger events. This trigger either starts a DMA block transfer or a single DMA transfer. In addition to the configured trigger, a DMA channel can always be triggered by setting its designated `DMAREQ.DMAREQx` flag. The DMA trigger sources are described in [Table 8-1](#).

**Source and destination increment:** The source and destination addresses can be controlled to increment or decrement or not change.

**Transfer mode:** The transfer mode determines whether the transfer should be a single transfer or a block transfer, or repeated versions of these.

**Byte or word transfers:** Determines whether each DMA transfer should be 8-bit (byte) or 16-bit (word).

**Interrupt mask:** An interrupt request is generated on completion of the DMA transfer. The interrupt mask bit controls whether the interrupt generation is enabled or disabled.

**M8:** Decide whether to use seven or eight bits per byte for transfer length. This is only applicable when doing byte transfers.

A detailed description of all configuration parameters is given in [Section 8.2.1](#) through [Section 8.2.11](#).

### 8.2.1 Source Address

The address in XDATA memory where the DMA channel starts to read data. This can be any XDATA address – in RAM, in the mapped flash bank (see `MEMCTR.XBANK`), XREG, or XDATA addressed SFR.

### 8.2.2 Destination Address

The first address to which the DMA channel should write the data read from the source address. The user must ensure that the destination is writable. This can be any XDATA address – in RAM, XREG, or XDATA addressed SFR.

### 8.2.3 Transfer Count

The number of bytes/words that must be transferred for the DMA transfer to be complete. When the transfer count is reached, the DMA controller rearms or disarms the DMA channel and alerts the CPU with an interrupt request. The transfer count can be defined in the configuration or it can be defined as variable-length, as described in [Section 8.2.4](#).

## 8.2.4 VLEN Setting

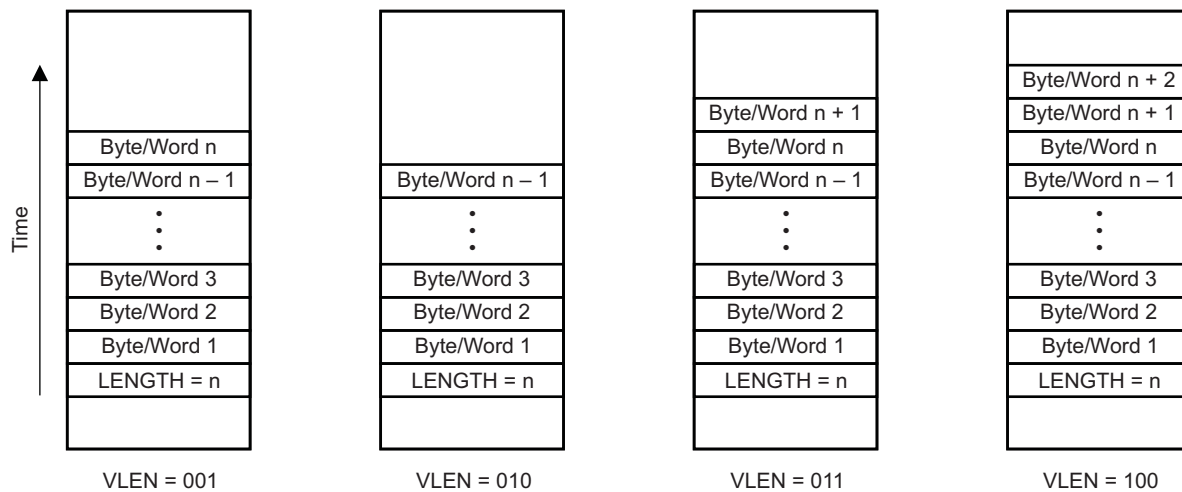
The DMA channel is capable of using the first byte or word (for word, bits 12:0 are used) in source data as the transfer length. This allows variable-length transfers. When using variable-length transfer, various options regarding how to count number of bytes to transfer are given. In any case, the transfer-count (LEN) setting is used as a maximum transfer count. If the transfer length specified by the first byte or word is greater than LEN, then LEN bytes or words are transferred. When using variable-length transfers, then LEN should be set to the largest allowed transfer length plus one.

Note that the M8 bit (Section 8.2.11) is only used when byte-size transfers are chosen.

Options which can be set with VLEN are the following:

1. Transfer number of bytes or words commanded by first byte/word + 1 (transfers the length byte/word, and then as many bytes/words as dictated by the length byte/word)
2. Transfer number of bytes or words commanded by first byte/word
3. Transfer number of bytes or words commanded by first byte/word + 2 (transfers the length byte/word, and then as many bytes/words as dictated by the length byte/word + 1)
4. Transfer number of bytes or words commanded by first byte/word + 3 (transfers the length byte/word, and then as many bytes/words as dictated by the length byte/word + 2)

Figure 8-2 shows the VLEN options.



M0103-02

**Figure 8-2. Variable Length (VLEN) Transfer Options**

## 8.2.5 Trigger Event

Each DMA channel can be set up to sense on a single trigger. This field determines which trigger the DMA channel senses.

## 8.2.6 Source and Destination Increment

When the DMA channel is armed or rearmed, the source and destination addresses are transferred to internal address pointers. The possibilities for address increment are:

- *Increment by zero.* The address pointer remains fixed after each transfer.
- *Increment by one.* The address pointer increments one count after each transfer.
- *Increment by two.* The address pointer increments two counts after each transfer.
- *Decrement by one.* The address pointer decrements one count after each transfer.

where a count equals 1 byte in byte mode and 2 bytes in word mode.



### 8.2.7 DMA Transfer Mode

The transfer mode determines how the DMA channel behaves when it starts transferring data. There are four transfer modes described as follows:

**Single:** On a trigger, a single DMA transfer occurs, and the DMA channel awaits the next trigger. After the number of transfers specified by the transfer count is completed, the CPU is notified, and the DMA channel is disarmed.

**Block:** On a trigger, the number of DMA transfers specified by the transfer count is performed as quickly as possible, after which the CPU is notified and the DMA channel is disarmed.

**Repeated single:** On a trigger, a single DMA transfer occurs, and the DMA channel awaits the next trigger. After the number of transfers specified by the transfer count is completed, the CPU is notified, and the DMA channel is rearmed.

**Repeated block:** On a trigger, the number of DMA transfers specified by the transfer count is performed as quickly as possible, after which the CPU is notified and the DMA channel is rearmed.

### 8.2.8 DMA Priority

A DMA priority is configurable for each DMA channel. The DMA priority is used to determine the winner in the case of multiple simultaneous internal memory requests, and whether the DMA memory access should have priority or not over a simultaneous CPU memory access. In case of an internal tie, a round-robin scheme is used to ensure access for all. There are three levels of DMA priority:

**High:** Highest internal priority. DMA access always prevails over CPU access.

**Normal:** Second-highest internal priority. DMA access prevails over the CPU on at least every second try.

**Low:** Lowest internal priority. DMA access always defers to a CPU access.

### 8.2.9 Byte or Word Transfers

Determines whether 8-bit (byte) or 16-bit (word) transfers are done.

### 8.2.10 Interrupt Mask

On completing a DMA transfer, the channel can generate an interrupt to the processor. This bit masks the interrupt.

### 8.2.11 Mode 8 Setting

This field determines whether to use 7 or 8 bits per byte for transfer length. Only applicable when doing byte transfers.

## 8.3 DMA Configuration Setup

The DMA channel parameters such as address mode, transfer mode, and priority, described in the previous section, must be configured before a DMA channel can be armed and activated. The parameters are not configured directly through SFR registers, but instead they are written in a special DMA configuration-data structure in memory. Each DMA channel in use requires its own DMA configuration-data structure. The DMA configuration-data structure consists of eight bytes and is described in [Section 8.6](#). A DMA configuration-data structure may reside at any location decided on by the user software, and the address location is passed to the DMA controller through a set of SFRs, `DMAxCFGH:DMAxCFGL`. Once a channel has been armed, the DMA controller reads the configuration data structure for that channel, given by the address in `DMAxCFGH:DMAxCFGL`.

It is important to note that the method for specifying the start address for the DMA configuration data structure differs between DMA channel 0 and DMA channels 1–4 as follows:

`DMA0CFGH:DMA0CFGL` gives the start address for the DMA channel 0 configuration data structure.

`DMA1CFGH:DMA1CFGL` gives the start address for the DMA channel 1 configuration data structure, followed by the channel 2–4 configuration-data structures.

Thus, the DMA controller expects the DMA configuration data structures for DMA channels 1–4 to lie in a contiguous area in memory starting at the address held in `DMA1CFGH:DMA1CFGL` and consisting of 32 bytes.

## 8.4 Stopping DMA Transfers

Ongoing DMA transfers or armed DMA channels are aborted using the `DMAARM` register to disarm the DMA channel.

One or more DMA channels are aborted by writing a 1 to the `DMAARM.ABORT` register bit, and at the same time selecting which DMA channels to abort by setting the corresponding `DMAARM.DMAARMx` bits to 1. When setting `DMAARM.ABORT` to 1, the `DMAARM.DMAARMx` bits for nonaborted channels must be written as 0.

No DMA interrupt is generated when aborting an ongoing DMA transfer (disarming a DMA channel).

## 8.5 DMA Interrupts

Each DMA channel can be configured to generate an interrupt to the CPU on completing a DMA transfer. This is accomplished with the `IRQMASK` bit in the channel configuration. The corresponding interrupt flag in the `DMAIRQ` SFR register is set when the interrupt is generated.

Regardless of the `IRQMASK` bit in the channel configuration, the corresponding interrupt flag in the `DMAIRQ` register is set on DMA channel completion. Thus, software should always check (and clear) this register when rearming a channel with a changed `IRQMASK` setting. Failure to do so could generate an interrupt based on the stored interrupt flag.

If a DMA transfer is aborted prior to its completion, the corresponding bit in the `DMAIRQ` register is not set, and an interrupt is not generated.

## 8.6 DMA Configuration-Data Structure

For each DMA channel, the DMA configuration-data structure consists of eight bytes. The configuration-data structure is described in [Table 8-2](#).

## 8.7 DMA Memory Access

The DMA data transfer is affected by endian convention. Note that the DMA descriptors follow big-endian convention while the other registers follow little-endian convention. This must be accounted for in compilers.

**Table 8-1. DMA Trigger Sources**

DMA Trigger		Functional Unit	Description
Number	Name		
0	NONE	DMA	No trigger, setting the <code>DMAREQ.DMAREQx</code> bit starts transfer.
1	PREV	DMA	DMA channel is triggered by completion of previous channel.
2	T1_CH0	Timer 1	Timer 1, compare, channel 0
3	T1_CH1	Timer 1	Timer 1, compare, channel 1
4	T1_CH2	Timer 1	Timer 1, compare, channel 2
5	T2_EVENT1	Timer 2	Timer 2, event pulse 1
6	T2_EVENT2	Timer 2	Timer 2, event pulse 2
7	T3_CH0	Timer 3	Timer 3, compare, channel 0
8	T3_CH1	Timer 3	Timer 3, compare, channel 1
9	T4_CH0	Timer 4	Timer 4, compare, channel 0
10	T4_CH1	Timer 4	Timer 4, compare, channel 1
11	ST	Sleep Timer (not in CC2540/41)	Sleep Timer compare
	RADIO1	Radio (CC2541)	Radio DMA trigger 1 (see <a href="#">Section 25.3.2</a> )

**Table 8-1. DMA Trigger Sources (continued)**

DMA Trigger		Functional Unit	Description
Number	Name		
12	IOC_0	I/O controller	Port 0 I/O pin input transition <sup>(1)</sup>
13	IOC_1	I/O controller	Port 1 I/O pin input transition <sup>(1)</sup>
14	URX0	USART 0	USART 0 RX complete
15	UTX0	USART 0	USART 0 TX complete
16	URX1	USART 1	USART 1 RX complete
17	UTX1	USART 1	USART 1 TX complete
18	FLASH	Flash controller	Flash data write complete
19	RADIO	Radio (not in CC2540)	CC253x: RF packet byte received (see <a href="#">Section 23.3</a> ) CC2541: Radio DMA trigger 0 (see <a href="#">Section 25.3.2</a> )
20	ADC_CHALL	ADC	ADC end of a conversion in a sequence, sample ready
21	ADC_CH11	ADC	ADC end of conversion channel 0 in sequence, sample ready
22	ADC_CH21	ADC	ADC end of conversion channel 1 in sequence, sample ready
23	ADC_CH32	ADC	ADC end of conversion channel 2 in sequence, sample ready
24	ADC_CH42	ADC	ADC end of conversion channel 3 in sequence, sample ready
25	ADC_CH53	ADC	ADC end of conversion channel 4 in sequence, sample ready
26	ADC_CH63	ADC	ADC end of conversion channel 5 in sequence, sample ready
27	ADC_CH74	ADC	ADC end of conversion channel 6 in sequence, sample ready
28	ADC_CH84	ADC	ADC end of conversion channel 7 in sequence, sample ready
29	ENC_DW	AES	AES encryption processor requests download of input data
30	ENC_UP	AES	AES encryption processor requests upload of output data
31	DBG_BW	Debug interface	Debug interface burst write

<sup>(1)</sup> Using this trigger source must be aligned with port interrupt-enable bits. Note that all interrupt-enabled port pins generate a trigger.

**Table 8-2. DMA Configuration-Data Structure**

Byte Offset	Bit	Name	Description
0	7:0	SRCADDR[15:8]	DMA channel source address, high
1	7:0	SRCADDR[7:0]	DMA channel source address, low
2	7:0	DESTADDR[15:8]	DMA channel destination address, high. Note that flash memory is not directly writable.
3	7:0	DESTADDR[7:0]	DMA channel destination address, low. Note that flash memory is not directly writable.
4	7:5	VLEN[2:0]	Variable-length transfer mode. In word mode, bits 12:0 of the first word are considered as the transfer length. 000: Use LEN for transfer count 001: Transfer the number of bytes or words specified by the first byte or word + 1 (up to a maximum specified by LEN). Thus, the transfer count excludes the length byte or word. 010: Transfer the number of bytes or words specified by the first byte or word (up to a maximum specified by LEN). Thus, the transfer count includes the length byte or word. 011: Transfer the number of bytes/words specified by the first byte/word + 2 (up to a maximum specified by LEN). 100: Transfer the number of bytes/words specified by the first byte/word + 3 (up to a maximum specified by LEN). 101: Reserved 110: Reserved 111: Alternative for using LEN as the transfer count
4	4:0	LEN[12:8]	The DMA channel transfer count

**Table 8-2. DMA Configuration-Data Structure (continued)**

Byte Offset	Bit	Name	Description
			Used as the maximum allowable length when VLEN differs from 000 and 111. The DMA channel counts in words when in WORDSIZE mode, and in bytes otherwise.
5	7:0	LEN[7:0]	The DMA channel transfer count Used as the maximum allowable length when VLEN differs from 000 and 111. The DMA channel counts in words when in WORDSIZE mode, and in bytes otherwise.
6	7	WORDSIZE	Selects whether each DMA transfer is 8-bit (0) or 16-bit (1).
6	6:5	TMODE[1:0]	The DMA channel transfer mode 00: Single 01: Block 10: Repeated single 11: Repeated block
6	4:0	TRIG[4:0]	Selects one of the triggers shown in <a href="#">Table 8-1</a>
7	7:6	SRCINC[1:0]	Source address increment mode (after each transfer): 00: 0 bytes or words 01: 1 byte or word 10: 2 bytes or word 11: -1 byte or word
7	5:4	DESTINC[1:0]	Destination address increment mode (after each transfer): 00: 0 bytes or words 01: 1 byte or word 10: 2 bytes or words 11: -1 byte or word
7	3	IRQMASK	Interrupt mask for this channel. 0: Disable interrupt generation 1: Enable interrupt generation on DMA channel done
7	2	M8	Mode of 8th bit for VLEN transfer length; only applicable when WORDSIZE = 0 and VLEN differs from 000 and 111. 0: Use all 8 bits for transfer count 1: Use 7 LSB for transfer count
7	1:0	PRIORITY[1:0]	The DMA channel priority: 00: Low, CPU has priority. 01: Assured, DMA at least every second try 10: High, DMA has priority 11: Reserved

## 8.8 DMA Registers

This section describes the SFR registers associated with the DMA controller.

### DMAARM (0xD6) – DMA Channel Arm

Bit	Name	Reset	R/W	Description
7	ABORT	0	R0/W	DMA abort. This bit is used to stop ongoing DMA transfers. Writing a 1 to this bit aborts all channels which are selected by setting the corresponding DMAARM bit to 1. 0: Normal operation 1: Abort all selected channels
6:5	–	00	R/W	Reserved
4	DMAARM4	0	R/W1	DMA arm channel 4 This bit must be set in order for any DMA transfers to occur on the channel. For nonrepetitive transfer modes, the bit is automatically cleared on completion.
3	DMAARM3	0	R/W1	DMA arm channel 3 This bit must be set in order for any DMA transfers to occur on the channel. For nonrepetitive transfer modes, the bit is automatically cleared on completion.
2	DMAARM2	0	R/W1	DMA arm channel 2 This bit must be set in order for any DMA transfers to occur on the channel. For nonrepetitive transfer modes, the bit is automatically cleared on completion.
1	DMAARM1	0	R/W1	DMA arm channel 1 This bit must be set in order for any DMA transfers to occur on the channel. For nonrepetitive transfer modes, the bit is automatically cleared on completion.
0	DMAARM0	0	R/W1	DMA arm channel 0 This bit must be set in order for any DMA transfers to occur on the channel. For nonrepetitive transfer modes, the bit is automatically cleared on completion.

### DMAREQ (0xD7) – DMA Channel Start Request and Status

Bit	Name	Reset	R/W	Description
7:5	–	000	R0	Reserved
4	DMAREQ4	0	R/W1 H0	DMA transfer request, channel 4 When set to 1, activate the DMA channel (has the same effect as a single trigger event). This bit is cleared when DMA transfer is started.
3	DMAREQ3	0	R/W1 H0	DMA transfer request, channel 3 When set to 1, activate the DMA channel (has the same effect as a single trigger event). This bit is cleared when DMA transfer is started.
2	DMAREQ2	0	R/W1 H0	DMA transfer request, channel 2 When set to 1, activate the DMA channel (has the same effect as a single trigger event). This bit is cleared when DMA transfer is started.
1	DMAREQ1	0	R/W1 H0	DMA transfer request, channel 1 When set to 1, activate the DMA channel (has the same effect as a single trigger event). This bit is cleared when DMA transfer is started.
0	DMAREQ0	0	R/W1 H0	DMA transfer request, channel 0 When set to 1, activate the DMA channel (has the same effect as a single trigger event). This bit is cleared when DMA transfer is started.

### DMA0CFGH (0xD5) – DMA Channel-0 Configuration Address High Byte

Bit	Name	Reset	R/W	Description
7:0	DMA0CFG[15:8]	0x00	R/W	The DMA channel-0 configuration address, high-order

### DMA0CFGL (0xD4) – DMA Channel-0 Configuration Address Low Byte

Bit	Name	Reset	R/W	Description
7:0	DMA0CFG[7:0]	0x00	R/W	The DMA channel 0 configuration address, low-order

**DMA1CFGH (0xD3) – DMA Channel 1–4 Configuration Address High Byte**

Bit	Name	Reset	R/W	Description
7:0	DMA1CFG[15:8]	0x00	R/W	The DMA channel 1–4 configuration address, high-order

**DMA1CFGH (0xD2) – DMA Channel 1–4 Configuration Address Low Byte**

Bit	Name	Reset	R/W	Description
7:0	DMA1CFG[7:0]	0x00	R/W	The DMA channel 1–4 configuration address, low-order

**DMAIRQ (0xD1) – DMA Interrupt Flag**

Bit	Name	Reset	R/W	Description
7:5	–	000	R0	Reserved
4	DMAIF4	0	R/W0	DMA channel-4 interrupt flag 0: DMA channel transfer not complete 1: DMA channel transfer complete or interrupt pending
3	DMAIF3	0	R/W0	DMA channel-3 interrupt flag 0: DMA channel transfer not complete 1: DMA channel transfer complete or interrupt pending
2	DMAIF2	0	R/W0	DMA channel-2 interrupt flag 0: DMA channel transfer not complete 1: DMA channel transfer complete or interrupt pending
1	DMAIF1	0	R/W0	DMA channel-1 interrupt flag 0: DMA channel transfer not complete 1: DMA channel transfer complete or interrupt pending
0	DMAIF0	0	R/W0	DMA channel-0 interrupt flag 0: DMA channel transfer not complete 1: DMA channel transfer complete or interrupt pending

## Timer 1 (16-Bit Timer)

Timer 1 is an independent 16-bit timer which supports typical timer and counter functions such as input capture, output compare, and PWM functions. The timer has five independent capture-or-compare channels. The timer uses one I/O pin per channel. The timer is used for a wide range of control and measurement applications, and the availability of up-and-down count mode with five channels allows, for example, implementation of motor-control applications.

The features of Timer 1 are as follows:

- Five capture-or-compare channels
- Rising-, falling-, or any-edge input capture
- Set, clear, or toggle output compare
- Free-running, modulo, or up-and down counter operation
- Clock prescaler for divide by 1, 8, 32, or 128
- Interrupt request generated on each capture or compare and terminal count
- DMA trigger function

Topic	Page
<b>9.1 16-Bit Counter</b> .....	<b>104</b>
<b>9.2 Timer 1 Operation</b> .....	<b>104</b>
<b>9.3 Free-Running Mode</b> .....	<b>104</b>
<b>9.4 Modulo Mode</b> .....	<b>105</b>
<b>9.5 Up-and-Down Mode</b> .....	<b>105</b>
<b>9.6 Channel-Mode Control</b> .....	<b>105</b>
<b>9.7 Input Capture Mode</b> .....	<b>106</b>
<b>9.8 Output Compare Mode</b> .....	<b>106</b>
<b>9.9 IR Signal Generation and Learning</b> .....	<b>111</b>
<b>9.10 Timer 1 Interrupts</b> .....	<b>113</b>
<b>9.11 Timer 1 DMA Triggers</b> .....	<b>113</b>
<b>9.12 Timer 1 Registers</b> .....	<b>114</b>
<b>9.13 Accessing Timer 1 Registers as Array</b> .....	<b>119</b>

## 9.1 16-Bit Counter

The timer consists of a 16-bit counter that increments or decrements at each active clock edge. The period of the active clock edges is defined by the register bits, `CLKCONCMD.TICKSPD`, which set the global division of the system clock, giving a variable clock-tick frequency from 0.25 MHz to 32 MHz (given the use of the 32-MHz XOSC as clock source). This frequency is further divided in Timer 1 by the prescaler value set by `T1CTL.DIV`. This prescaler value can be 1, 8, 32, or 128. Thus, the lowest clock frequency used by Timer 1 is 1953.125 Hz and the highest is 32 MHz when the 32 MHz XOSC is used as system clock source. When the 16-MHz RCOSC is used as system clock source, then the highest clock frequency used by Timer 1 is 16 MHz.

The counter operates as a free-running counter, a modulo counter, or an up-and-down counter for use in center-aligned PWM.

It is possible to read the 16-bit counter value through the two 8-bit SFRs, `T1CNTH` and `T1CNTL`, containing the high-order byte and low-order byte, respectively. When `T1CNTL` is read, the high-order byte of the counter at that instant is buffered in `T1CNTH` so that the high-order byte can be read from `T1CNTH`. Thus, `T1CNTL` must always be read first, before reading `T1CNTH`.

All write accesses to the `T1CNTL` register reset the 16-bit counter.

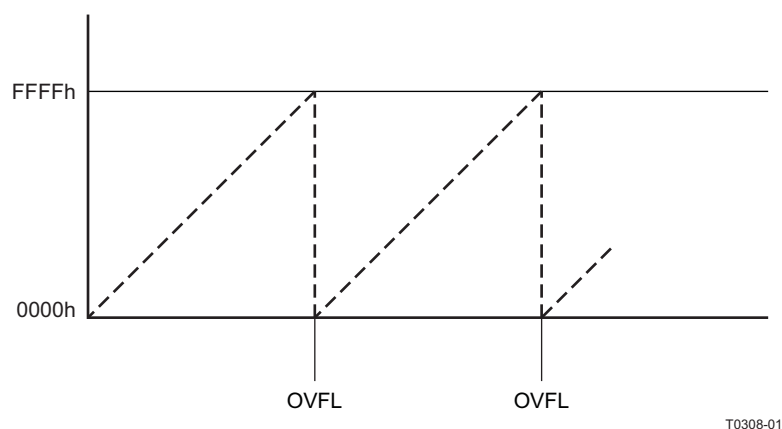
The counter produces an interrupt request when the terminal count value (overflow) is reached. It is possible to start and halt the counter with `T1CTL` control register settings. The counter is started when a value other than 00 is written to `T1CTL.MODE`. If 00 is written to `T1CTL.MODE`, the counter halts at its present value.

## 9.2 Timer 1 Operation

In general, control register `T1CTL` is used to control the timer operation. The status register `T1STAT` holds the interrupt flags. The various modes of operation are described as follows.

## 9.3 Free-Running Mode

In the free-running mode of operation, the counter starts from 0x0000 and increments at each active clock edge. When the counter reaches 0xFFFF (overflow), the counter is loaded with 0x0000 and continues incrementing its value as shown in [Figure 9-1](#). When the terminal count value 0xFFFF is reached, the interrupt flag `T1STAT.OVFIF` is set. An interrupt request is generated if enabled; see [Section 9.10](#) for details. The free-running mode can be used to generate independent time intervals and output-signal frequencies.



**Figure 9-1. Free-Running Mode**



### 9.4 Modulo Mode

When the timer operates in modulo mode, the 16-bit counter starts at 0x0000 and increments at each active clock edge. After the counter has reached the period value T1CC0, held in registers T1CC0H:T1CC0L, the counter is reset to 0x0000 and continues to increment. If the timer is started with a value above T1CC0, the interrupt flag T1STAT.OVFIF is set when the terminal count value (0xFFFF) is reached, after which the counter wraps to 0x0000. An interrupt request is generated if enabled; see Section 9.10 for details. If a periodic interrupt is wanted at the period value, this can be obtained by enabling an output compare interrupt on channel 0, as explained in Section 9.8. The modulo mode can be used for applications where a period other than 0xFFFF is required. The counter operation is shown in Figure 9-2.

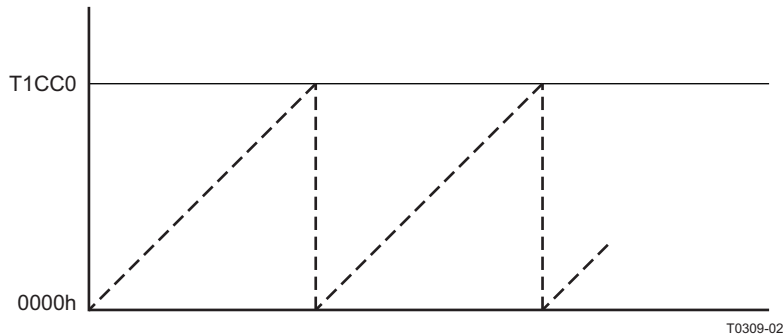


Figure 9-2. Modulo Mode

### 9.5 Up-and-Down Mode

In the up-and-down timer mode, the counter repeatedly starts from 0x0000 and counts up until the value held in T1CC0H:T1CC0L is reached, and then the counter counts down until 0x0000 is reached, as shown in Figure 9-3. This timer mode is used when symmetrical output pulses are required with a period other than 0xFFFF, and therefore allows implementation of center-aligned PWM output applications. The interrupt flag T1STAT.OVFIF is set when the counter value reaches 0x0000 in the up-and-down mode. An interrupt request is generated if enabled, see Section 9.10 for details.

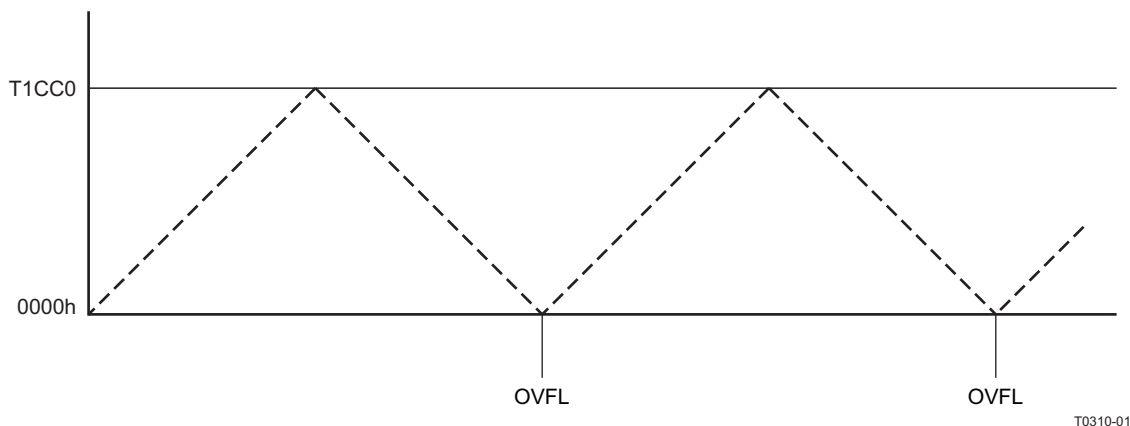


Figure 9-3. Up-and-Down Mode

### 9.6 Channel-Mode Control

The channel mode is set for each channel with its control and status register, T1CTLn. The settings include input capture and output compare modes.

## 9.7 Input Capture Mode

When a channel is configured as an input capture channel, the I/O pin associated with that channel is configured as an input. After the timer has been started, a rising edge, falling edge, or any edge on the input pin triggers a capture of the 16-bit counter contents into the associated capture register. Thus, the timer is able to capture the time when an external event takes place.

---

**NOTE:** Before an I/O pin can be used by the timer, the required I/O pin must be configured as a Timer 1 peripheral pin.

---

The channel input pin is synchronized to the internal system clock. Thus, pulses on the input pin must have a minimum duration greater than the system clock period.

The content of the 16-bit capture register is read out from registers T1CCnH:T1CCnL.

When the capture takes place, the interrupt flag for the channel, T1STAT.CHnIF (n is the channel number), is set. An interrupt request is generated if enabled; see [Section 9.10](#) for details.

## 9.8 Output Compare Mode

In output compare mode, the I/O pin associated with a channel is set as an output. After the timer has been started, the contents of the counter are compared with the contents of the channel compare register T1CCnH:T1CCnL. If the compare register equals the counter contents, the output pin is set, reset, or toggled, according to the compare output mode setting of T1CCTLn.CMP. Note that all edges on output pins are glitch-free when operating in a given output compare mode. Writing to the compare register T1CCnL is buffered, so that a value written to T1CCnL does not take effect until the corresponding high-order register, T1CCnH, is written. Writing to compare registers T1CCnH:T1CCnL does not take effect on the output compare value until the counter value is 0x00.

Note that channel 0 has fewer output compare modes because T1CC0H:T1CC0L has a special function in modes 6 and 7, meaning these modes would not be useful for channel 0.

When a compare occurs, the interrupt flag for the channel, T1STAT.CHnIF (n is the channel number), is set. An interrupt request is generated if enabled; see [Section 9.10](#) for details.

Examples of output compare modes in various timer modes are given in the following figures.

**Edge-aligned:** PWM output signals can be generated using the timer modulo mode and channels 1 and 2 in output compare mode 6 or 7 (defined by the T1CCTLn.CMP bits, where n is 1 or 2) as shown in [Figure 9-4](#). The period of the PWM signal is determined by the setting in T1CC0, and the duty cycle is determined by T1CCn, where n is the PWM channel, 1 or 2.

The timer free-running mode may also be used. In this case, CLKCONCMD.TICKSPD and the prescaler divider value in the T1CTL.DIV bits set the period of the PWM signal. The polarity of the PWM signal is determined by whether output compare mode 6 or 7 is used.

PWM output signals can also be generated using output compare modes 4 and 5 as shown in [Figure 9-4](#), or by using modulo mode as shown in [Figure 9-5](#). Using output compare mode 4 or 5 is preferred for simple PWM.

**Center-aligned:** PWM outputs can be generated when the timer up-and-down mode is selected. Channel output compare mode 4 or 5 (defined by T1CCTLn.CMP bits, where n is 1 or 2) is selected, depending on the required polarity of the PWM signal. The period of the PWM signal is determined by T1CC0, and the duty cycle for the channel output is determined by T1CCn, where n is the PWM channel, 1 or 2.

The center-aligned PWM mode is required by certain types of motor-drive applications, and typically less noise is produced than in the edge-aligned PWM mode, because the I/O pin transitions are not lined up on the same clock edge.

In some types of applications, a defined delay or dead time is required between outputs. Typically, this is required for outputs driving an H-bridge configuration to avoid uncontrolled cross-conduction in one side of the H-bridge. The delay or dead-time can be obtained in the PWM outputs by using T1CCn as shown in the following:

Assuming that channel 1 and channel 2 are used to drive the outputs using timer up-and-down mode and the channels use output compare modes 4 and 5, respectively, then the timer period (in Timer 1 clock periods) is:

$$t_p = T1CC0 \times 2$$

and the dead time, that is, the time when both outputs are low, (in Timer 1 clock periods) is given by:

$$t_d = T1CC1 - T1CC2$$

A compare output pin is initialized to the value listed in [Table 9-1](#) when:

- a value is written to T1CNTL (all Timer 1 channels)
- 0x7 is written to T1CCTLn.CMP (channel n)

**Table 9-1. Initial Compare Output Values (Compare Mode)**

Compare Mode (T1CCTLn.CMP)	Initial Compare Output
Set output on compare (000)	0
Clear output on compare (001)	1
Toggle output on compare (010)	0
Set output on compare-up, clear on compare down in up-and-down mode (011)	0
In other modes than up-and-down mode, set output on compare, clear on 0 (011)	0
Clear output on compare-up, set on compare down in up-and-down mode (100)	1
In other modes than up-and-down mode, clear output on compare, set on 0 (100)	1
Clear when equal T1CC0, set when equal T1CCn (101)	0
Set when equal T1CC0, clear when equal T1CCn (110)	1

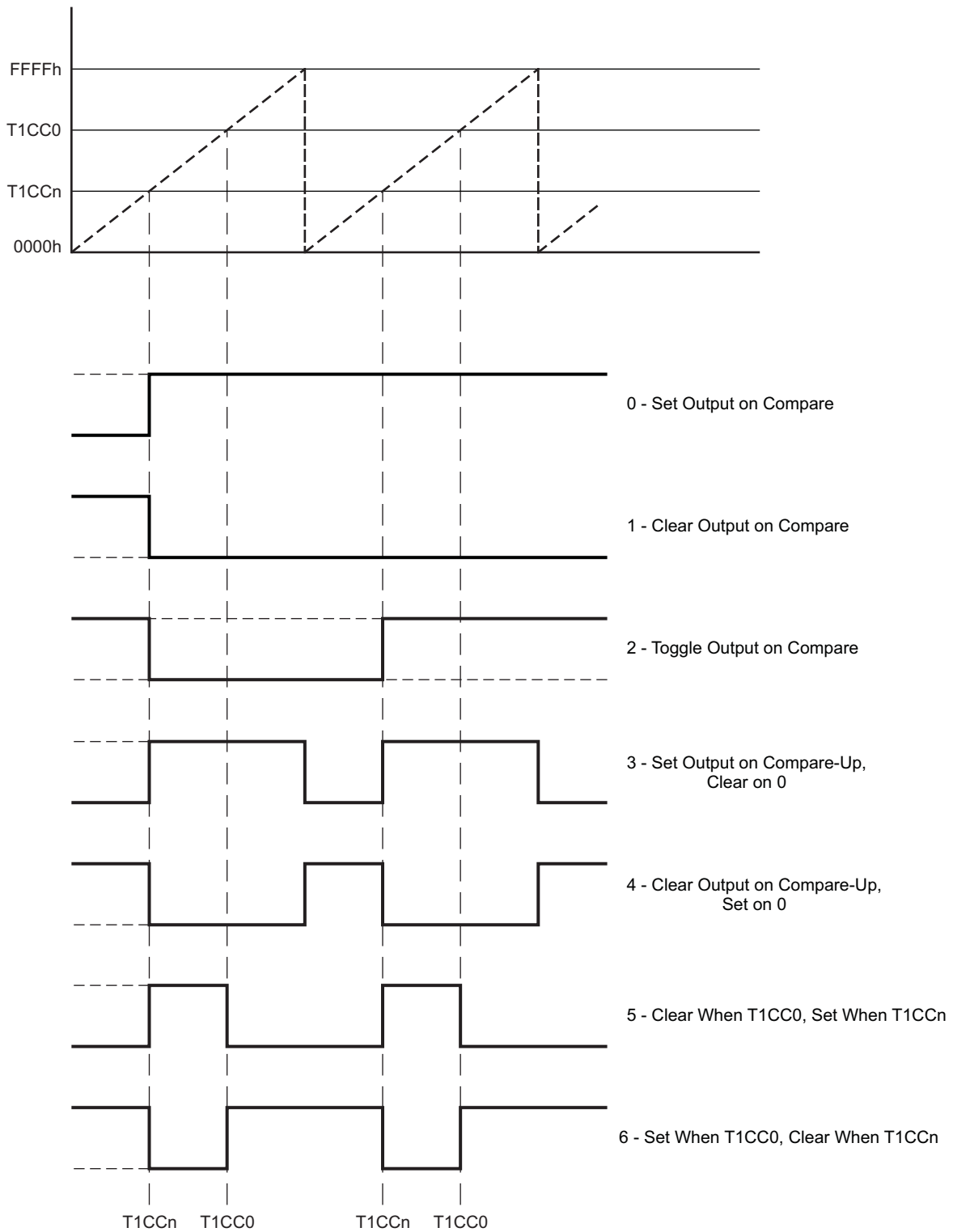
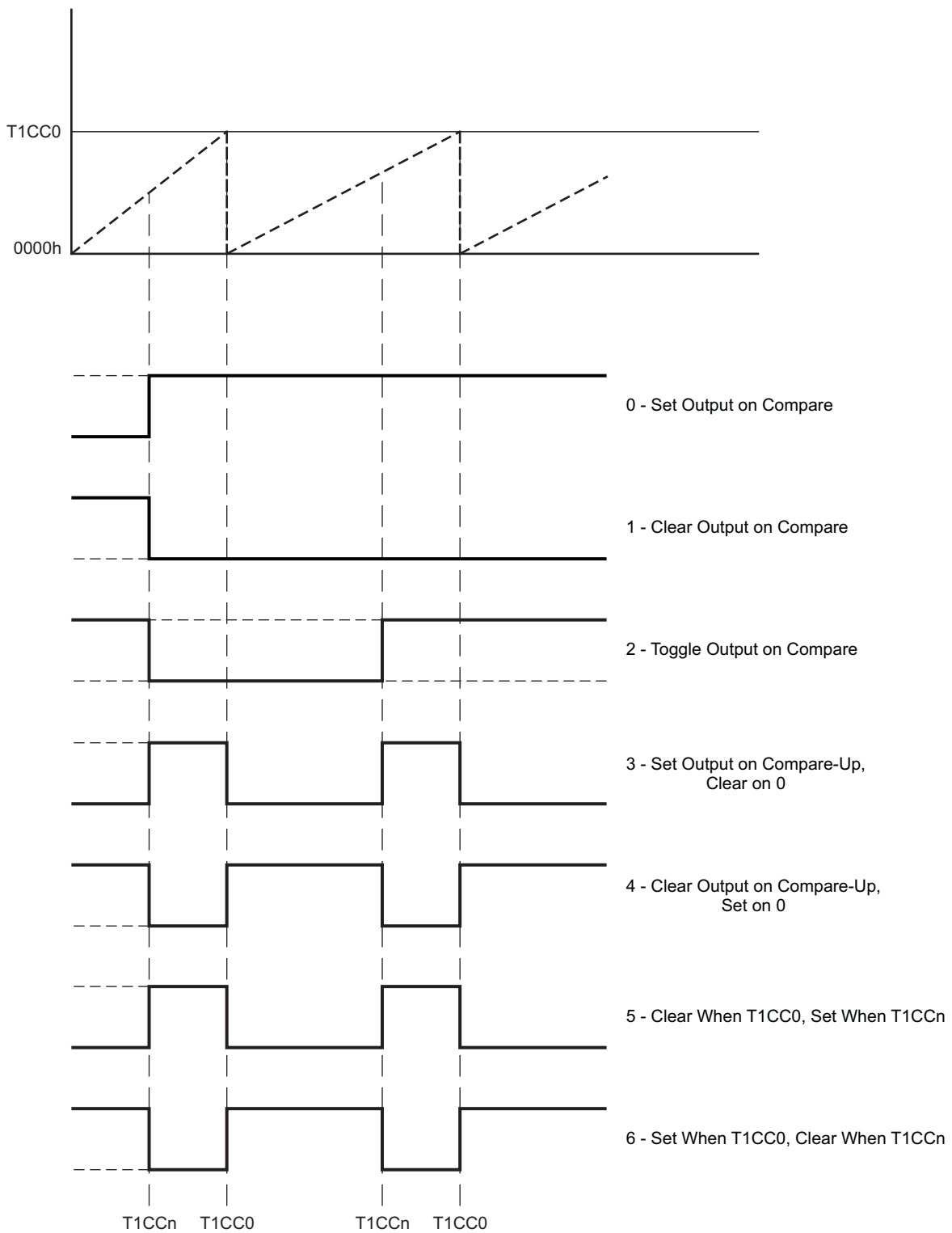


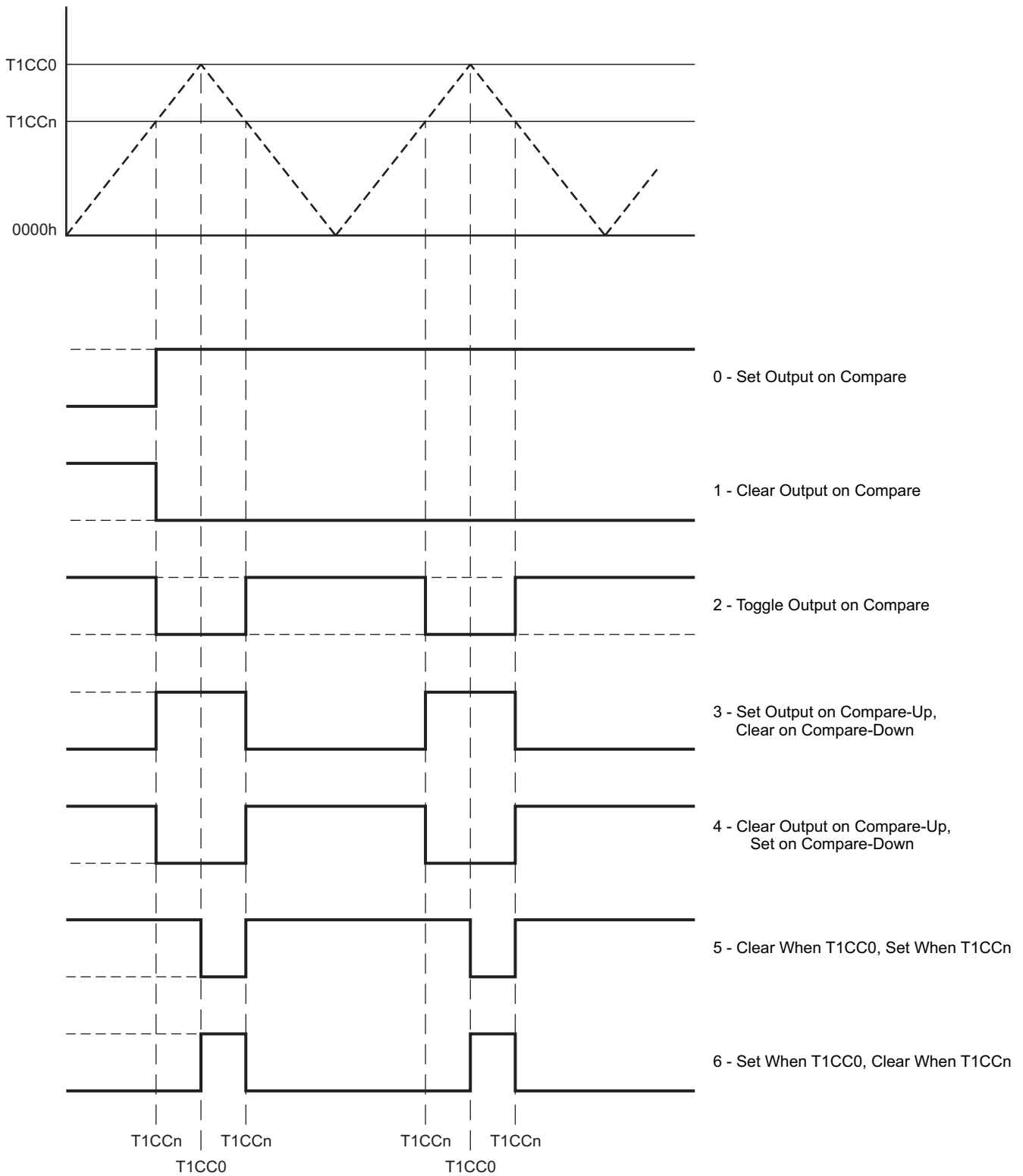
Figure 9-4. Output Compare Modes, Timer Free-Running Mode

T0311-01



T0312-01

Figure 9-5. Output Compare Modes, Timer Modulo Mode



T0313-01

**Figure 9-6. Output Compare Modes, Timer Up-and-Down Mode**

## 9.9 IR Signal Generation and Learning

This section describes how Timer 1 can be configured in IR generation mode, where it counts Timer 3 periods and the output is ANDed with the output of Timer 3 to generate modulated consumer IR signals with minimal CPU interaction.

### 9.9.1 Introduction

Generation of IR signals for remote control is generally done in one of two ways:

- Modulated codes
- Non-modulated codes (C-codes, flash codes)

The device includes flexible timer functionality to implement generation and learning of both types of IR signals with minimal CPU interaction. Most IR protocols can be implemented with only one CPU intervention per command.

### 9.9.2 Modulated Codes

Modulated codes can be generated using Timer 1 (16-bit) and Timer 3 (8-bit). Timer 3 in modulo mode is used to generate the carrier. Timer 3 has an individual prescaler for its input. Its period is set using `T3CC0`. Timer 3 channel 1 is used for PWM output. The duty cycle of the carrier is set using `T3CC1`. Channel 1 uses compare mode: *Clear output on compare, set on 0x00* (`T3CCTL1.CMP = 100`). [Table 9-2](#) shows the frequency error calculation for a 38-kHz carrier using Timer 3.

**Table 9-2. Frequency Error Calculation for 38-kHz Carrier**

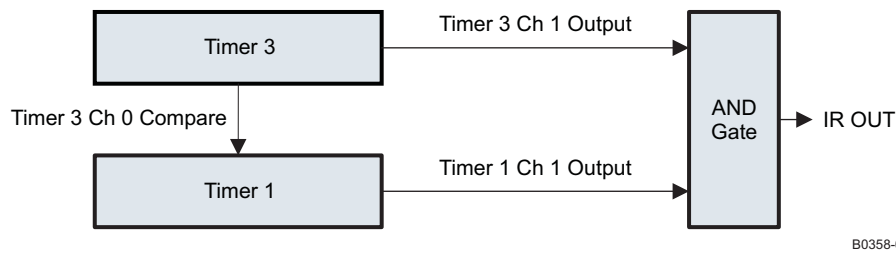
Description	Value
System clock frequency	32,000 kHz
IR carrier frequency	38 kHz
System clock period	0.00003125 ms
IR carrier period	0.026315789 ms
Timer prescaler	4
Timer period	0.000125 ms
Ideal timer value	210.5263158
True timer value	211
True timer period	0.026375 ms
True timer frequency	37.91469194 kHz
Period error	59.21052632 ns
Frequency error	85.30805687 Hz
Frequency error %	0.2245%

The `IRCTL.IRGEN` register bit enables IR generation mode in Timer 1. When the `IRGEN` bit is set, Timer 1 takes the output of the Timer 3 channel 1 compare signal as tick instead of the system tick. The Timer 1 period is set using `T1CC0` with Timer 1 in modulo mode (`T1CTL.MODE = 10`) and channel 0 in compare mode (`T1CCTL0.MODE = 1`). Channel 1 compare mode *Clear output on compare, set on 0x0000* (`T1CCTL1.CMP = 100`) is used for output of the gating signal.

The number of *mark* carrier periods is set by `T1CC1`. `T1CC1` must be updated every Timer 1 period by the DMA or CPU. Note that an update to `T1CC1` is buffered and does not take effect before Timer 1 reaches `0x0000`.

The number of *space* carrier periods is set by `T1CC0`. Its value should be set to the total number of *mark* and *space* carrier periods wanted. The compare values are buffered until the timer hits `0x0000`.

The output of Timer 1 channel 1 is ANDed with that of Timer 3 channel 1 to form the IR output as shown in [Figure 9-7](#)

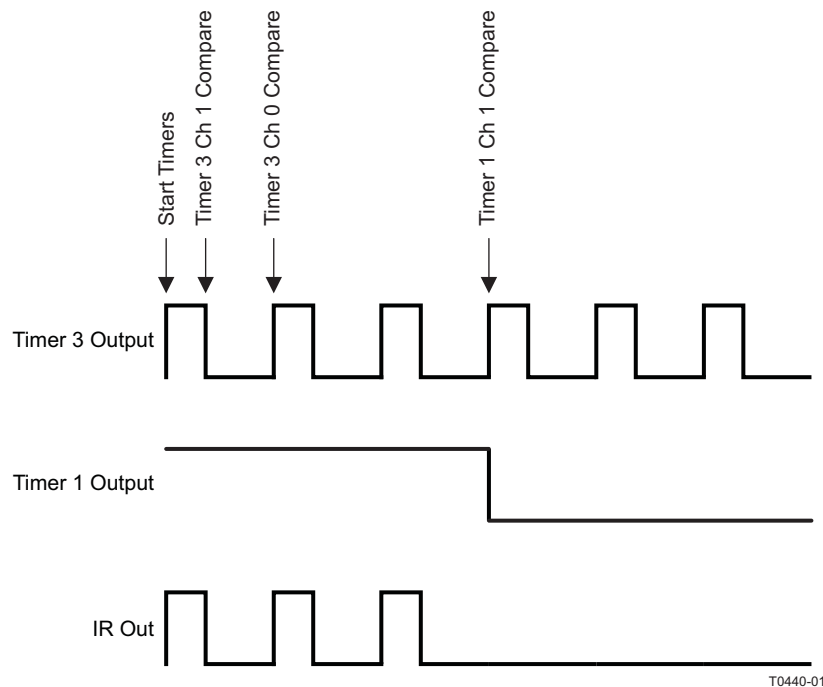


**Figure 9-7. Block Diagram of Timers in IR-Generation Mode**

The timing of the Timer 3 channel 1 output and Timer 1 channel 1 output signals is synchronized such that no glitches are produced on the IR Out signal.

When the `IRGEN` bit is set, the IR out signal is routed to pins instead of the normal Timer 1 channel 1 output (see also [Section 7.6.1](#)).

[Figure 9-8](#) shows the example of Timer 3 being initialized to a 33% duty cycle ( $T3CC0 = 3 \times T3CC1$ ). Timer 1 has been initialized to 3.



**Figure 9-8. Modulated Waveform Example**

To achieve a period of *space* only, `T1CC1` should be set to `0x00`.

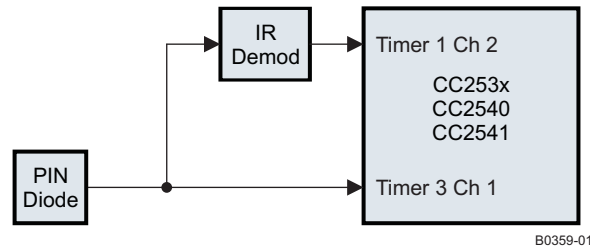
### 9.9.3 Non-Modulated Codes

To generate non-modulated IR codes, Timer 1 is used in modulo mode. The period of the signal is given by `T1CC0`, and the pulse duration is given by `T1CC1`. `T1CC1` gives the length of the *mark* period, and `T1CC0` gives the total number of *mark* and *space* periods. The compare values are buffered until the timer hits `0x0000`. The compare values must be updated once every period by the DMA or CPU if they are not to be kept the same.



### 9.9.4 Learning

Learning is done by using the capture function of Timer 1 (16-bit) and Timer 3 (8-bit). Timer 3 can handle the carrier frequency detection and Timer 1 can handle the code learning from the demodulated signal. The circuit could be set up as described in [Figure 9-9](#)



**Figure 9-9. IR Learning Board Diagram**

#### 9.9.4.1 Carrier Frequency Detection

Timer 3 is used to capture and detect the carrier frequency with input directly from the IR PIN diode. The timer should sample the carrier a limited number of times. If a carrier is detected, the frequency detected should contribute to the average number, which is what can be stored in the database.

#### 9.9.4.2 Demodulated Code Learning

The output from the IR PIN diode is demodulated by an appropriate circuit. The output from this circuit is used as input to one of the Timer 1 channels in capture mode.

### 9.9.5 Other Considerations

The IR output pin should be placed in the high-impedance state or pulled down during reset to avoid unnecessary power consumption from illuminating the IR LED. Note that only the P1.1 output for Timer 1 channel 1 is placed in the high-impedance state with no pullup during and after reset.

## 9.10 Timer 1 Interrupts

One interrupt vector is assigned to the timer. An interrupt request is generated when one of the following timer events occurs:

- Counter reaches terminal count value (overflow, or turns around zero).
- Input capture event
- Output compare event

The status register, `T1STAT`, contains the source interrupt flags for the terminal-count value event and the five channel compare or capture events. A source interrupt flag is set when the corresponding event occurs, regardless of interrupt mask bits. The CPU interrupt flag `IRCON.T1IF` is set when one of the events occurs if the corresponding interrupt mask bit is equal to 1. The interrupt mask bits are `T1CTLn.IM` for the five channels and `T1OVFIM.T1OVFIM` for the overflow event. The CPU interrupt flag `IRCON.T1IF` is also set when a Timer 1 source interrupt flag is being cleared and one or more other Timer 1 source interrupt flags are still set while the corresponding interrupt mask bit is set. An interrupt request is generated when `IRCON.T1IF` goes from 0 to 1 if `IEN1.T1IEN` and `IEN0.EA` are both equal to 1.

## 9.11 Timer 1 DMA Triggers

There are three DMA triggers associated with Timer 1. These are DMA triggers `T1_CH0`, `T1_CH1`, and `T1_CH2`, which are generated on timer compare events as follows:

- `T1_CH0` – Channel 0 compare
- `T1_CH1` – Channel 1 compare
- `T1_CH2` – Channel 2 compare

There are no triggers associated with channels 3 and 4.

## 9.12 Timer 1 Registers

This section describes the Timer 1 registers, which consist of the following registers:

- T1CNTH – Timer 1 count high
- T1CNTL – Timer 1 count low
- T1CTL – Timer 1 control
- T1STAT – Timer 1 status
- T1CTLn – Timer 1 channel n capture or compare control
- T1CCnH – Timer 1 channel n capture or compare value high
- T1CCnL – Timer 1 channel n capture or compare value low

The TIMIF.T1OVFIM register bit resides in the TIMIF register, which is described together with the Timer 3 and Timer 4 registers.

### T1CNTH (0xE3) – Timer 1 Counter High

Bit	Name	Reset	R/W	Description
7:0	CNT[15:8]	0x00	R	Timer count high-order byte. Contains the high byte of the 16-bit timer counter buffered at the time T1CNTL is read

### T1CNTL (0xE2) – Timer 1 Counter Low

Bit	Name	Reset	R/W	Description
7:0	CNT[7:0]	0x00	R/W	Timer count low-order byte. Contains the low byte of the 16-bit timer counter. Writing anything to this register results in the counter being cleared to 0x0000 and initializes all output pins of associated channels.

### T1CTL (0xE4) – Timer 1 Control

Bit	Name	Reset	R/W	Description
7:4	–	0000	R0	Reserved
3:2	DIV[1:0]	00	R/W	Prescaler divider value. Generates the active clock edge used to update the counter as follows: 00: Tick frequency / 1 01: Tick frequency / 8 10: Tick frequency / 32 11: Tick frequency / 128
1:0	MODE [1:0]	00	R/W	Timer 1 mode select. The timer operating mode is selected as follows: 00: Operation is suspended. 01: Free-running, repeatedly count from 0x0000 to 0xFFFF. 10: Modulo, repeatedly count from 0x0000 to T1CC0. 11: Up-and-down, repeatedly count from 0x0000 to T1CC0 and from T1CC0 down to 0x0000.

**T1STAT (0xAF) – Timer 1 Status**

Bit	Name	Reset	R/W	Description
7:6	–	00	R0	Reserved
5	OVFIF	0	R/W0	Timer 1 counter-overflow interrupt flag. Set when the counter reaches the terminal count value in free-running or modulo mode, and when zero is reached counting down in up-and-down mode. Writing a 1 has no effect.
4	CH4IF	0	R/W0	Timer 1 channel 4 interrupt flag. Set when the channel 4 interrupt condition occurs. Writing a 1 has no effect.
3	CH3IF	0	R/W0	Timer 1 channel 3 interrupt flag. Set when the channel 3 interrupt condition occurs. Writing a 1 has no effect.
2	CH2IF	0	R/W0	Timer 1 channel 2 interrupt flag. Set when the channel 2 interrupt condition occurs. Writing a 1 has no effect.
1	CH1IF	0	R/W0	Timer 1 channel 1 interrupt flag. Set when the channel 1 interrupt condition occurs. Writing a 1 has no effect.
0	CH0IF	0	R/W0	Timer 1 channel 0 interrupt flag. Set when the channel 0 interrupt condition occurs. Writing a 1 has no effect.

**T1CTL0 (0xE5) – Timer 1 Channel 0 Capture or Compare Control**

Bit	Name	Reset	R/W	Description
7	RFIRQ	0	R/W	When set, use RF interrupt for capture instead of regular capture input.
6	IM	1	R/W	Channel 0 interrupt mask. Enables interrupt request when set.
5:3	CMP[2:0]	000	R/W	Channel 0 compare-mode select. Selects action on output when timer value equals compare value in T1CC0 000: Set output on compare 001: Clear output on compare 010: Toggle output on compare 011: Set output on compare-up, clear on 0 100: Clear output on compare-up, set on 0 101: Reserved 110: Reserved 111: Initialize output pin. CMP[2:0] is not changed.
2	MODE	0	R/W	Mode. Select Timer 1 channel 0 capture or compare mode 0: Capture mode 1: Compare mode
1:0	CAP[1:0]	00	R/W	Channel 0 capture-mode select 00: No capture 01: Capture on rising edge 10: Capture on falling edge 11: Capture on all edges

**T1CC0H (0xDB) – Timer 1 Channel 0 Capture or Compare Value, High**

Bit	Name	Reset	R/W	Description
7:0	T1CC0[15:8]	0x00	R/W	Timer 1 channel 0 capture or compare value high-order byte. Writing to this register when T1CTL0.MODE = 1 (compare mode) causes the T1CC0[15:0] update to the written value to be delayed until T1CNT = 0x0000.

**T1CC0L (0xDA) – Timer 1 Channel 0 Capture or Compare Value, Low**

Bit	Name	Reset	R/W	Description
7:0	T1CC0[7:0]	0x00	R/W	Timer 1 channel 0 capture or compare value low-order byte. Data written to this register is stored in a buffer but not written to T1CC0[7:0] until, and at the same time as, a later write to T1CC0H takes effect.

**T1CCTL1 (0xE6) – Timer 1 Channel 1 Capture or Compare Control**

Bit	Name	Reset	R/W	Description
7	RFIRQ	0	R/W	When set, use RF interrupt for capture instead of regular capture input.
6	IM	1	R/W	Channel 1 interrupt mask. Enables interrupt request when set.
5:3	CMP[2:0]	000	R/W	Channel 1 compare-mode select. Selects action on output when timer value equals compare value in T1CC1. 000: Set output on compare 001: Clear output on compare 010: Toggle output on compare 011: Set output on compare-up, clear on compare-down in up-and-down mode. Otherwise set output on compare, clear on 0. 100: Clear output on compare-up, set on compare-down in up-and-down mode. Otherwise clear output on compare, set on 0. 101: Clear when equal T1CC0, set when equal T1CC1 110: Set when equal T1CC0, clear when equal T1CC1 111: Initialize output pin. CMP[2:0] is not changed.
2	MODE	0	R/W	Mode. Select Timer 1 channel 1 capture or compare mode 0: Capture mode 1: Compare mode
1:0	CAP[1:0]	00	R/W	Channel 1 capture-mode select 00: No capture 01: Capture on rising edge 10: Capture on falling edge 11: Capture on all edges

**T1CC1H (0xDD) – Timer 1 Channel 1 Capture or Compare Value, High**

Bit	Name	Reset	R/W	Description
7:0	T1CC1[15:8]	0x00	R/W	Timer 1 channel 1 capture or compare value high-order byte. Writing to this register when T1CCTL1.MODE = 1 (compare mode) causes the T1CC1[15:0] update to the written value to be delayed until T1CNT = 0x0000.

**T1CC1L (0xDC) – Timer 1 Channel 1 Capture or Compare Value, Low**

Bit	Name	Reset	R/W	Description
7:0	T1CC1[7:0]	0x00	R/W	Timer 1 channel 1 capture or compare value low-order byte. Data written to this register is stored in a buffer but not written to T1CC1[7:0] until, and at the same time as, a later write to T1CC1H takes effect.

**T1CCTL2 (0xE7) – Timer 1 Channel 2 Capture or Compare Control**

Bit	Name	Reset	R/W	Description
7	RFIRQ	0	R/W	When set, use RF interrupt for capture instead of regular capture input.
6	IM	1	R/W	Channel 2 interrupt mask. Enables interrupt request when set.
5:3	CMP[2:0]	000	R/W	Channel 2 compare mode select. Selects action on output when timer value equals compare value in T1CC2. 000: Set output on compare 001: Clear output on compare 010: Toggle output on compare 011: Set output on compare-up, clear on compare-down in up-and-down mode. Otherwise set output on compare, clear on 0. 100: Clear output on compare-up, set on compare-down in up-and-down mode. Otherwise clear output on compare, set on 0. 101: Clear when equal T1CC0, set when equal T1CC2 110: Set when equal T1CC0, clear when equal T1CC2 111: Initialize output pin. CMP[2:0] is not changed.
2	MODE	0	R/W	Mode. Select Timer 1 channel 2 capture or compare mode 0: Capture mode 1: Compare mode
1:0	CAP[1:0]	00	R/W	Channel 2 capture-mode select 00: No capture 01: Capture on rising edge 10: Capture on falling edge 11: Capture on all edges

**T1CC2H (0xDF) – Timer 1 Channel 2 Capture or Compare Value, High**

Bit	Name	Reset	R/W	Description
7:0	T1CC2[15:8]	0x00	R/W	Timer 1 channel 2 capture or compare value high-order byte. Writing to this register when T1CCTL2.MODE = 1 (compare mode) causes the T1CC2[15:0] update to the written value to be delayed until T1CNT = 0x0000.

**T1CC2L (0xDE) – Timer 1 Channel 2 Capture or Compare Value, Low**

Bit	Name	Reset	R/W	Description
7:0	T1CC2[7:0]	0x00	R/W	Timer 1 channel 2 capture or compare value low-order byte. Data written to this register is stored in a buffer but not written to T1CC2[7:0] until, and at the same time as, a later write to T1CC2H takes effect.

**T1CCTL3 (0x62A3) – Timer 1 Channel 3 Capture/Compare Control**

Bit	Name	Reset	R/W	Description
7	RFIRQ	0	R/W	When set, use RF interrupt for capture instead of regular capture input.
6	IM	1	R/W	Channel 3 interrupt mask. Enables interrupt request when set.
5:3	CMP[2:0]	000	R/W	Channel 3 compare mode select. Selects action on output when timer value equals compare value in T1CC3. 000: Set output on compare 001: Clear output on compare 010: Toggle output on compare 011: Set output on compare-up, clear on compare-down in up-and-down mode. Otherwise set output on compare, clear on 0. 100: Clear output on compare-up, set on compare down in up-and-down mode. Otherwise clear output on compare, set on 0. 101: Clear when equal T1CC0, set when equal T1CC3 110: Set when equal T1CC0, clear when equal T1CC3 111: Initialize output pin. CMP[2:0] is not changed.
2	MODE	0	R/W	Mode. Select Timer 1 channel 3 capture or compare mode 0: Capture mode 1: Compare mode
1:0	CAP[1:0]	00	R/W	Channel 3 capture-mode select 00: No capture 01: Capture on rising edge 10: Capture on falling edge 11: Capture on all edges

**T1CC3H (0x62AD) – Timer 1 Channel 3 Capture or Compare Value, High**

Bit	Name	Reset	R/W	Description
7:0	T1CC3[15:8]	0x00	R/W	Timer 1 channel 3 capture or compare value high-order byte. Writing to this register when T1CCTL3.MODE = 1 (compare mode) causes the T1CC3[15:0] update to the written value to be delayed until T1CNT = 0x0000.

**T1CC3L (0x62AC) – Timer 1 Channel 3 Capture or Compare Value, Low**

Bit	Name	Reset	R/W	Description
7:0	T1CC3[7:0]	0x00	R/W	Timer 1 channel 3 capture or compare value low-order byte. Data written to this register is stored in a buffer but not written to T1CC3[7:0] until, and at the same time as, a later write to T1CC3H takes effect.

**T1CCTL4 (0x62A4) – Timer 1 Channel 4 Capture or Compare Control**

Bit	Name	Reset	R/W	Description
7	RFIRQ	0	R/W	When set, use RF interrupt for capture instead of regular capture input.
6	IM	1	R/W	Channel 4 interrupt mask. Enables interrupt request when set.
5:3	CMP[2:0]	000	R/W	Channel 4 compare mode select. Selects action on output when timer value equals compare value in T1CC4. 000: Set output on compare 001: Clear output on compare 010: Toggle output on compare 011: Set output on compare-up, clear on compare down in up-and-down mode. Otherwise set output on compare, clear on 0. 100: Clear output on compare-up, set on compare down in up-and-down mode. Otherwise clear output on compare, set on 0. 101: Clear when equal T1CC0, set when equal T1CC4 110: Set when equal T1CC0, clear when equal T1CC4 111: Initialize output pin. CMP[2:0] is not changed.
2	MODE	0	R/W	Mode. Select Timer 1 channel 4 capture or compare mode 0: Capture mode 1: Compare mode
1:0	CAP[1:0]	00	R/W	Channel 4 capture-mode select 00: No capture 01: Capture on rising edge 10: Capture on falling edge 11: Capture on all edges

**T1CC4H (0x62AF) – Timer 1 Channel 4 Capture or Compare Value, High**

Bit	Name	Reset	R/W	Description
7:0	T1CC4[15:8]	0x00	R/W	Timer 1 channel 4 capture or compare value high-order byte. Writing to this register when T1CCTL4.MODE = 1 (compare mode) causes the T1CC4[15:0] update to the written value to be delayed until T1CNT = 0x0000.

**T1CC4L (0x62AE) – Timer 1 Channel 4 Capture or Compare Value, Low**

Bit	Name	Reset	R/W	Description
7:0	T1CC4[7:0]	0x00	R/W	Timer 1 channel 4 capture or compare value low-order byte. Data written to this register is stored in a buffer but not written to T1CC4[7:0] until, and at the same time as, a later write to T1CC4H takes effect.

**IRCTL (0x6281) – Timer 1 IR Generation Control**

Bit	Name	Reset	R/W	Description
7:1	–	0000 000	R/W	Reserved
0	IRGEN	0	R/W	When this bit is set, a connection between Timer 3 channel 1 and Timer 1 tick input is made so that the timers can be used to generate modulated IR codes (see also <a href="#">Section 9.9</a> ).

### 9.13 Accessing Timer 1 Registers as Array

The Timer 1 capture or compare channel registers can be accessed as a contiguous region in the XDATA memory space. This facilitates accessing the registers as a simple indexed structure. The five capture or compare control registers are mapped to 0x62A0–0x62A4. The 16-bit capture or compare values are mapped to 0x62A6–0x62AF; 0x62A5 is unused.

## **Timer 3 and Timer 4 (8-Bit Timers)**

Timer 3 and Timer 4 are two 8-bit timers. Each timer has two independent capture-or-compare channels, each using one I/O pin per channel.

Features of Timer 3 and Timer 4 are as follows:

- Two capture-or-compare channels
- Set, clear, or toggle output compare
- Clock prescaler for divide by 1, 2, 4, 8, 16, 32, 64, 128
- Interrupt request generated on each capture or compare and terminal-count event
- DMA trigger function

Topic	Page
<b>10.1 8-Bit Timer Counter .....</b>	<b>121</b>
<b>10.2 Timer 3 and Timer 4 Mode Control.....</b>	<b>121</b>
<b>10.3 Channel Mode Control .....</b>	<b>121</b>
<b>10.4 Input Capture Mode .....</b>	<b>122</b>
<b>10.5 Output Compare Mode .....</b>	<b>122</b>
<b>10.6 Timer 3 and Timer 4 Interrupts .....</b>	<b>122</b>
<b>10.7 Timer 3 and Timer 4 DMA Triggers .....</b>	<b>123</b>
<b>10.8 Timer 3 and Timer 4 Registers .....</b>	<b>123</b>



## 10.1 8-Bit Timer Counter

All timer functions are based on the main 8-bit counter found in Timer 3 and Timer 4. The counter increments or decrements at each active clock edge. The period of the active clock edges, as defined by the register bits `CLKCONCMD.TICKSPD[2:0]`, is further multiplied (the frequency is divided) by the prescaler value set by `TxCTL.DIV[2:0]` (where x refers to the timer number, 3 or 4). The counter operates as either a free-running counter, a down counter, a modulo counter, or an up-and-down counter.

It is possible to read the 8-bit counter value through the SFR register `TxCNT`, where x refers to the timer number, 3 or 4.

The possibility to clear and halt the counter is given with `TxCTL` control register settings. The counter is started when a 1 is written to `TxCTL.START`. If a 0 is written to `TxCTL.START`, the counter halts at its present value.

## 10.2 Timer 3 and Timer 4 Mode Control

In general, the control register `TxCTL` is used to control the timer operation.

### 10.2.1 Free-Running Mode

In the free-running mode of operation, the counter starts from 0x00 and increments at each active clock edge. When the counter reaches 0xFF, the counter is loaded with 0x00 and continues incrementing its value. When the terminal count value 0xFF is reached (that is, an overflow occurs), the interrupt flag `TIMIF.TxOVFIF` is set. An interrupt request is generated if enabled; see [Section 10.6](#) for details. The free-running mode can be used to generate independent time intervals and output-signal frequencies.

### 10.2.2 Down Mode

In the down mode, after the timer has been started, the counter is loaded with the contents in `TxCC0`. The counter then counts down to 0x00. The interrupt flag `TIMIF.TxOVFIF` is set when 0x00 is reached. An interrupt request is generated if enabled; see [Section 10.6](#) for details. The timer down mode can generally be used in applications where an event time-out interval is required.

### 10.2.3 Modulo Mode

When the timer operates in modulo mode, the 8-bit counter starts at 0x00 and increments at each active clock edge. After the count has reached the period value held in register `TxCC0`, the counter is reset to 0x00 and continues to increment. If the timer started with a value above `TxCC0`, the interrupt flag `TIMIF.TxOVFIF` is set when the terminal value (0xFF) is reached, after which the counter wraps to 0x00. An interrupt request is generated if enabled; see [Section 10.6](#) for details. If a periodic interrupt is wanted at the period value, this can be obtained by enabling an output compare interrupt on channel 0, as explained in [Section 10.5](#). The modulo mode can be used for applications where a period other than 0xFF is required.

### 10.2.4 Up-and-Down Mode

In the up-and-down timer mode, the counter repeatedly starts from 0x00 and counts up until the value held in `TxCC0` is reached, and then the counter counts down until 0x00 is reached. This timer mode is used when symmetrical output pulses are required with a period other than 0xFF, allowing implementation of center-aligned PWM output applications. The interrupt flag `TIMIF.TxOVFIF` is set when the counter value reaches 0x00 in the up-and-down mode. An interrupt request is generated if enabled; see [Section 10.6](#) for details.

Clearing the counter by writing to `TxCTL.CLR` also resets the count direction to the count-up-from-0x00 mode.

## 10.3 Channel Mode Control

The channel modes for each channel, 0 and 1, are set by the control and status registers `TxCCTLn`, where n is the channel number, 0 or 1. The settings include capture and compare modes.

## 10.4 Input Capture Mode

When a channel is configured as an input capture channel, the I/O pin associated with that channel is configured as an input. After the timer has been started, a rising edge, falling edge, or any edge on the input pin triggers a capture of the 8-bit counter contents into the associated capture register. Thus, the timer is able to capture the time when an external event takes place.

---

**NOTE:** Before an I/O pin can be used by the timer, the required I/O pin must be configured as a Timer 3 or Timer 4 peripheral pin.

---

The channel input pin is synchronized to the internal system clock. Thus, pulses on the input pin must have a minimum duration greater than the system clock period.

The content of the 8-bit capture register for channel n is read out from register `T3CCn` or `T4CCn`.

When the capture takes place, the interrupt flag for the channel, `TIMIF.TxCHnIF` (x is 3 or 4, n is the channel number), is set. An interrupt request is generated if enabled; see [Section 10.6](#) for details.

## 10.5 Output Compare Mode

In output-compare mode, the I/O pin associated with a channel must be set to an output. After the timer has been started, the content of the counter is compared with the contents of channel compare register `TxCCn`. If the compare register equals the counter contents, the output pin is set, reset, or toggled according to the compare output mode setting of `TxCCTL.CMP1:0`. Note that all edges on output pins are glitch-free when operating in a given compare output mode.

For simple PWM use, output compare modes 4 and 5 are preferred.

Writing to compare register `TxCC0` or `TxCC1` does not take effect on the output compare value until the counter value is 0x00.

When the capture takes place, the interrupt flag for the channel, `TIMIF.TxCHnIF` (x is 3 or 4, n is the channel number), is set. An interrupt request is generated if enabled; see [Section 10.6](#) for details.

A compare output pin is initialized to the value listed in [Table 10-1](#) when:

- a 1 is written to `TxCNTR.CLR` (All Timer x channels)
- 0x7 is written to `TxCCTLn.CMP` (Timer x, channel n)

**Table 10-1. Initial Compare Output Values (Compare Mode)**

Compare Mode ( <code>TxCCTLn.CMP</code> )	Initial Compare Output
Set output on compare (000)	0
Clear output on compare (001)	1
Toggle output on compare (010)	0
Set output on compare-up, clear on compare-down in up-and-down mode (011)	0
In other modes than up-and-down mode, set output on compare, clear on 0 (011)	0
Clear output on compare-up, set on compare-down in up-and-down mode (100)	1
In other modes than up-and-down mode, clear output on compare, set on 0 (100)	1
Set output on compare, clear on 0xFF (101)	0
Clear output on compare, set on 0x00 (110)	1

## 10.6 Timer 3 and Timer 4 Interrupts

One interrupt vector is assigned to each of the timers. These are T3 and T4. An interrupt request is generated when one of the following timer events occurs:

- Counter reaches terminal count value.
- Compare event

- Capture event

The SFR register `TIMIF` contains all interrupt flags for Timer 3 and Timer 4. The register bits `TIMIF.TxOVFIF` and `TIMIF.TxCHnIF` contain the source interrupt flags for the two terminal-count value events and the four channel-compare events, respectively. A source interrupt flag is set when the corresponding event occurs, regardless of interrupt mask bits. The CPU interrupt flag `IRCON.T3IF` or `IRCON.T4IF` is set when one of the events occurs if the corresponding interrupt mask bit is equal to 1. The interrupt mask bits are `TxCTLn.IM` for the four channels and `TxCTL.OVFIM` for the overflow events. The CPU interrupt flag `IRCON.T3IF` or `IRCON.T4IF` is also set when a Timer 3 or Timer 4 source interrupt flag is being cleared and one or more other source interrupt flags for the same timer are still set while the corresponding interrupt mask bit is set. An interrupt request is generated when `IRCON.TxIF` goes from 0 to 1 if `IEN1.TxIEN` and `IEN0.EA` are both equal to 1 (x is 3 or 4).

## 10.7 Timer 3 and Timer 4 DMA Triggers

Two DMA triggers are associated with Timer 3, and two DMA triggers are associated with Timer 4.

- T3\_CH0: Timer 3 channel 0 capture or compare
- T3\_CH1: Timer 3 channel 1 capture or compare
- T4\_CH0: Timer 4 channel 0 capture or compare
- T4\_CH1: Timer 4 channel 1 capture or compare

## 10.8 Timer 3 and Timer 4 Registers

### T3CNT (0xCA) – Timer 3 Counter

Bit	Name	Reset	R/W	Description
7:0	CNT[7:0]	0x00	R	Timer count byte. Contains the current value of the 8-bit counter

### T3CTL (0xCB) – Timer 3 Control

Bit	Name	Reset	R/W	Description
7:5	DIV[2:0]	000	R/W	Prescaler divider value. Generates the active clock edge used to clock the timer from <code>CLKCONCMD.TICKSPD</code> as follows: 000: Tick frequency / 1 001: Tick frequency / 2 010: Tick frequency / 4 011: Tick frequency / 8 100: Tick frequency / 16 101: Tick frequency / 32 110: Tick frequency / 64 111: Tick frequency / 128
4	START	0	R/W	Start timer. Normal operation when set, suspended when cleared
3	OVFIM	1	R/W	Overflow interrupt mask 0: Interrupt is disabled. 1: Interrupt is enabled.
2	CLR	0	R0/W1	Clear counter. Writing a 1 to CLR resets the counter to 0x00 and initializes all output pins of associated channels. Always read as 0.
1:0	MODE[1:0]	00	R/W	Timer 3 mode. Select the mode as follows: 00: Free-running, repeatedly count from 0x00 to 0xFF 01: Down, count from T3CC0 to 0x00 10: Modulo, repeatedly count from 0x00 to T3CC0 11: Up-and-down, repeatedly count from 0x00 to T3CC0 and down to 0x00

**T3CCTL0 (0xCC) – Timer 3 Channel 0 Capture or Compare Control**

Bit	Name	Reset	R/W	Description
7	–	0	R0	Reserved
6	IM	1	R/W	Channel 0 interrupt mask 0: Interrupt is disabled. 1: Interrupt is enabled.
5:3	CMP[2:0]	000	R/W	Channel 0 compare output mode select. Specified action occurs on output when timer value equals compare value in T3CC0. 000: Set output on compare 001: Clear output on compare 010: Toggle output on compare 011: Set output on compare-up, clear on 0 100: Clear output on compare-up, set on 0 101: Set output on compare, clear on 0xFF 110: Clear output on compare, set on 0x00 111: Initialize output pin. CMP[2:0] is not changed.
2	MODE	0	R/W	Mode. Select Timer 3 channel 0 mode 0: Capture mode 1: Compare mode
1:0	CAP[1:0]	00	R/W	Capture mode select 00: No capture 01: Capture on rising edge 10: Capture on falling edge 11: Capture on both edges

**T3CC0 (0xCD) – Timer 3 Channel 0 Capture or Compare Value**

Bit	Name	Reset	R/W	Description
7:0	VAL[7:0]	0x00	R/W	Timer capture or compare value, channel 0. Writing to this register when T3CCTL0.MODE = 1 (compare mode) causes the T3CC0.VAL[7:0] update to the written value to be delayed until T3CNT.CNT[7:0] = 0x00.

**T3CCTL1 (0xCE) – Timer 3 Channel 1 Capture or Compare Control**

Bit	Name	Reset	R/W	Description
7	–	0	R0	Reserved
6	IM	1	R/W	Channel 1 interrupt mask 0: Interrupt is disabled. 1: Interrupt is enabled.
5:3	CMP[2:0]	000	R/W	Channel 1 compare output-mode select. Specified action on output when timer value equals compare value in T3CC1 000: Set output on compare 001: Clear output on compare 010: Toggle output on compare 011: Set on compare-up, clear on compare-down in up-and-down mode. Otherwise, set output on compare, clear on 0. 100: Clear output on compare-up, set on compare-down in up-and-down mode. Otherwise clear output on compare, set on 0. 101: Set output on compare, clear on 0xFF 110: Clear output on compare, set on 0x00 111: Initialize output pin. CMP[2:0] is not changed
2	MODE	0	R/W	Mode. Select Timer 3 channel 1 mode 0: Capture mode 1: Compare mode
1:0	CAP[1:0]	00	R/W	Capture mode select 00: No capture 01: Capture on rising edge 10: Capture on falling edge 11: Capture on both edges

**T3CC1 (0xCF) – Timer 3 Channel 1 Capture or Compare Value**

Bit	Name	Reset	R/W	Description
7:0	VAL[7:0]	0x00	R/W	Timer 3 capture or compare value, channel 1. Writing to this register when T3CCTL1.MODE = 1 (compare mode) causes the T3CC1.VAL[7:0] update to the written value to be delayed until T3CNT.CNT[7:0] = 0x00.

**T4CNT (0xEA) – Timer 4 Counter**

Bit	Name	Reset	R/W	Description
7:0	CNT[7:0]	0x00	R	Timer count byte. Contains the current value of the 8-bit counter

**T4CTL (0xEB) – Timer 4 Control**

Bit	Name	Reset	R/W	Description
7:5	DIV[2:0]	000	R/W	Prescaler divider value. Generates the active clock edge used to clock the timer from CLKCONCMD.TICKSPD as follows: 000: Tick frequency / 1 001: Tick frequency / 2 010: Tick frequency / 4 011: Tick frequency / 8 100: Tick frequency / 16 101: Tick frequency / 32 110: Tick frequency / 64 111: Tick frequency / 128
4	START		R/W	Start timer. Normal operation when set, suspended when cleared
3	OVFIM	1	R/W	Overflow interrupt mask
2	CLR	0	R0/W1	Clear counter. Writing a 1 to CLR resets the counter to 0x00 and initializes all output pins of associated channels. Always read as 0.
1:0	MODE[1:0]	0	R/W	Timer 4 mode. Select the mode as follows: 00: Free running, repeatedly count from 0x00 to 0xFF 01: Down, count from T4CC0 to 0x00 10: Modulo, repeatedly count from 0x00 to T4CC0 11: Up-and-down, repeatedly count from 0x00 to T4CC0 and down to 0x00

**T4CTL0 (0xEC) – Timer 4 Channel 0 Capture/Compare Control**

Bit	Name	Reset	R/W	Description
7	–	0	R0	Reserved
6	IM	1	R/W	Channel 0 interrupt mask
5:3	CMP[2:0]	000	R/W	Channel 0 compare output-mode select. Specified action occurs on output when timer value equals compare value in T4CC0. 000: Set output on compare 001: Clear output on compare 010: Toggle output on compare 011: Set output on compare-up, clear on 0 100: Clear output on compare-up, set on 0 101: Set output on compare, clear on 0xFF 110: Clear output on compare, set on 0x00 111: Initialize output pin. CMP[2:0] is not changed.
2	MODE	0	R/W	Mode. Select Timer 4 channel 0 mode 0: Capture mode 1: Compare mode
1:0	CAP[1:0]	00	R/W	Capture mode select. 00 – No capture, 01 – Capture on rising edge, 10 – Capture on falling edge, 11 – Capture on both edges

**T4CC0 (0xED) – Timer 4 Channel 0 Capture or Compare Value**

Bit	Name	Reset	R/W	Description
7:0	VAL[7:0]	0x00	R/W	Timer 4 capture or compare value, channel 0. Writing to this register when T4CTL0.MODE = 1 (compare mode) causes the T4CC0.VAL[7:0] update to the written value to be delayed until T4CNT.CNT[7:0] = 0x00.

**T4CCTL1 (0xEE) – Timer 4 Channel 1 Capture or Compare Control**

Bit	Name	Reset	R/W	Description
7	–	0	R0	Reserved
6	IM	1	R/W	Channel 1 interrupt mask
5:3	CMP[2:0]	000	R/W	Channel 1 compare output-mode select. Specified action on output when timer value equals compare value in T4CC1 000: Set output on compare 001: Clear output on compare 010: Toggle output on compare 011: Set on compare-up, clear on compare-down in up-down mode. Otherwise, set output on compare, clear on 0. 100: Clear output on compare-up, set on compare-down in up-down mode. Otherwise clear output on compare, set on 0. 101: Set output on compare, clear on 0xFF 110: Clear output on compare, set on 0x00 111: Initialize output pin. CMP[2:0] is not changed.
2	MODE	0	R/W	Mode. Select Timer 4 channel 1 mode 0: Capture mode 1: Compare mode
1:0	CAP[1:0]	00	R/W	Capture mode select. 00 – No Capture, 01 – Capture on rising edge, 10 – Capture on falling edge, 11 – Capture on both edges

**T4CC1 (0xEF) – Timer 4 Channel 1 Capture or Compare Value**

Bit	Name	Reset	R/W	Description
7:0	VAL[7:0]	0x00	R/W	Timer capture/compare value, channel 1. Writing to this register when T4CCTL1.MODE = 1 (compare mode) causes the T4CC1.VAL[7:0] update to the written value to be delayed until T4CNT.CNT[7:0] = 0x00.

**TIMIF (0xD8) – Timer 1 Interrupt Mask, Timers 3 and 4 Interrupt Flags**

Bit	Name	Reset	R/W	Description
7	–	0	R0	Reserved
6	T1OVFIM	1	R/W	Timer 1 overflow interrupt mask
5	T4CH1IF	0	R/W0	Timer 4 channel 1 interrupt flag 0: No interrupt is pending. 1: Interrupt is pending.
4	T4CH0IF	0	R/W0	Timer 4 channel 0 interrupt flag 0: No interrupt is pending. 1: Interrupt is pending.
3	T4OVFIF	0	R/W0	Timer 4 overflow interrupt flag 0: No interrupt is pending. 1: Interrupt is pending.
2	T3CH1IF	0	R/W0	Timer 3 channel 1 interrupt flag 0: No interrupt is pending. 1: Interrupt is pending.
1	T3CH0IF	0	R/W0	Timer 3 channel 0 interrupt flag 0: No interrupt is pending. 1: Interrupt is pending.
0	T3OVFIF	0	R/W0	Timer 3 overflow interrupt flag 0: No interrupt is pending. 1: Interrupt is pending.

## Sleep Timer

---

---

The Sleep Timer is used to set the period during which the system enters and exits low-power modes PM1 and PM2. The Sleep Timer is also used to maintain timing in Timer 2 when entering power mode PM1 or PM2.

The main features of the Sleep Timer are the following:

- 24-bit timer up-counter operating at 32-kHz clock rate
- 24-bit compare with interrupt and DMA trigger
- 24-bit capture

Topic	Page
11.1 General .....	129
11.2 Timer Compare.....	129
11.3 Timer Capture .....	129
11.4 Sleep Timer Registers .....	130



## 11.1 General

The Sleep Timer is a 24-bit timer running on the 32-kHz clock (either RCOSC or XOSC). The timer starts running immediately after a reset and continues to run uninterrupted.

The current value of the timer can be read from SFR registers `ST2:ST1:ST0`. When `ST0` is read, the current value of the 24-bit counter is latched. Thus, the `ST0` register must be read before `ST1` and `ST2` to read a correct Sleep Timer count value.

The Sleep Timer is running when operating in all power modes except PM3. The value of the Sleep Timer is not preserved in PM3. When returning from PM1 or PM2 (where the system clock is shut down), the Sleep Timer value in `ST2:ST1:ST0` is not up-to-date until a positive edge on the 32-kHz clock has been detected after the system clock restarted. To ensure an updated value is read, wait for a positive transition on the 32-kHz clock by polling the `SLEEPSTA.CLK32K` bit, before reading the Sleep Timer value.

Note that if supply voltage drops below 2 V while in PM2, the sleep interval might be affected.

## 11.2 Timer Compare

A timer compare event occurs when the timer value is equal to the 24-bit compare value and there is a positive edge on the 32-kHz clock. The compare value is set by writing to registers `ST2:ST1:ST0`. Writing to `ST0` while `STLOAD.LDRDY` is 1 initiates loading of the new compare value, that is, the most-recent values written to the `ST2`, `ST1`, and `ST0` registers. This means that when writing a compare value, `ST2` and `ST1` must be written before `ST0`. `STLOAD.LDRDY` is 0 during the load, and software must not start a new load until `STLOAD.LDRDY` has flipped back to 1.

When setting a new compare value, the value should be at least 5 more than the current sleep timer value. Otherwise, the timer compare event may be lost.

The interrupt enable bit for the ST interrupt is `IEN0.STIE`, and the interrupt flag is `IRCON.STIF`. When a timer compare event occurs, the interrupt flag `IRCON.STIF` is asserted.

In PM1 and PM2, the Sleep Timer compare event may be used to wake up the device and return to active operation in active mode. The default value of the compare value after reset is 0xFF FFFF.

For all devices except the CC2540 and CC2541, the Sleep Timer compare event can also be used as a DMA trigger (DMA trigger 11 in [Table 8-1](#)).

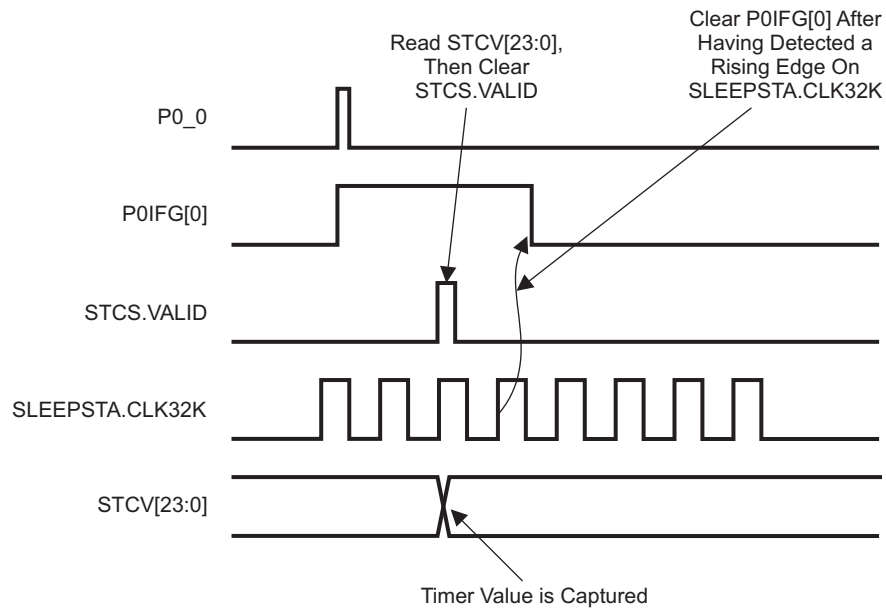
Note that if supply voltage drops below 2 V while in PM2, the sleep interval might be affected.

## 11.3 Timer Capture

The timer capture occurs when the interrupt flag for a selected I/O pin is set and this event has been detected by the 32-kHz clock. Sleep Timer capture is enabled by setting `STCC.PORT[1:0]` and `STCC.PIN[2:0]` to the I/O pin that is to be used to trigger the capture. When `STCS.VALID` goes high, the capture value in `STCV2:STCV1:STCV0` can be read. The captured value is 1 more than the value at the instant for the event on the I/O pin. Software should therefore subtract 1 from the captured value if absolute timing is required. To enable a new capture, follow these steps:

1. Clear `STCS.VALID`.
2. Wait until `SLEEPSTA.CLK32K` is low.
3. Wait until `SLEEPSTA.CLK32K` is high.
4. Clear the pin interrupt flag in the `P0IFG`,  
`P1IFG`, or `P2IFG` register.

This sequence, using the rising edge on P0.0 as an example, is shown in [Figure 11-1](#). Failure to follow the procedure may cause the capture functionality to stop working until a chip reset.



T0412-01

**Figure 11-1. Sleep Timer Capture (Example Using Rising Edge on P0\_0)**

It is not possible to switch the input-capture pin while capture is enabled. Capture must be disabled before a new input-capture pin can be selected. To disable capture, follow these steps (note that interrupts are disabled for up to half of a 32-kHz cycle, or 15.26  $\mu$ s):

1. Disable interrupts.
2. Wait until `SLEEPSTA.CLK32K` is high.
3. Set `STCC.PORT[1:0]` to 3. This disables capture.

## 11.4 Sleep Timer Registers

The registers used by the Sleep Timer are:

- `ST2` – Sleep Timer 2
- `ST1` – Sleep Timer 1
- `ST0` – Sleep Timer 0
- `STLOAD` – Sleep Timer load status
- `STCC` – Sleep Timer capture control
- `STCS` – Sleep Timer capture status
- `STCV0` – Sleep Timer capture value byte 0
- `STCV1` – Sleep Timer capture value byte 1
- `STCV2` – Sleep Timer capture value byte 2

### ST2 (0x97) – Sleep Timer 2

Bit	Name	Reset	R/W	Description
7:0	<code>ST2[7:0]</code>	0x00	R/W	Sleep Timer count or compare value. When read, this register returns the high bits [23:16] of the Sleep Timer count. When writing, this register sets the high bits [23:16] of the compare value. The value read is latched at the time of reading register <code>ST0</code> . The value written is latched when <code>ST0</code> is written.

**ST1 (0x96) – Sleep Timer 1**

Bit	Name	Reset	R/W	Description
7:0	ST1[7:0]	0x00	R/W	Sleep Timer count or compare value. When read, this register returns the middle bits [15:8] of the Sleep Timer count. When writing, this register sets the middle bits [15:8] of the compare value. The value read is latched at the time of reading register ST0. The value written is latched when ST0 is written.

**ST0 (0x95) – Sleep Timer 0**

Bit	Name	Reset	R/W	Description
7:0	ST0[7:0]	0x00	R/W	Sleep Timer count or compare value. When read, this register returns the low bits [7:0] of the Sleep Timer count. When writing, this register sets the low bits [7:0] of the compare value. Writes to this register are ignored unless STLOAD.LDRDY is 1.

**STLOAD (0xAD) – Sleep Timer Load Status**

Bit	Name	Reset	R/W	Description
7:1	–	0000 00 0	R0	Reserved
0	LDRDY	1	R	Load ready. This bit is 0 while the Sleep Timer loads the 24-bit compare value and 1 when the Sleep Timer is ready to start loading a new compare value.

**STCC (0x62B0) – Sleep Timer Capture Control**

Bit	Name	Reset	R/W	Description
7:5	–	000	R0	Reserved
4:3	PORT[1:0]	11	R	Port select. Valid settings are 0–2. Capture is disabled when set to 3, that is, an invalid setting is selected.
2:0	PIN[2:0]	111		Pin select. Valid settings are 0–7 when PORT[1:0] is 0 or 1, 0–5 when PORT[1:0] is 2. Capture is disabled when an invalid setting is selected.

**STCS (0x62B1) – Sleep Timer Capture Status**

Bit	Name	Reset	R/W	Description
7:1	–	0000 00 0	R0	Reserved
0	VALID	0	R/W0	Capture valid flag. Set to 1 when capture value in STCV has been updated. Clear explicitly to allow new capture.

**STCV0 (0x62B2) – Sleep Timer Capture Value Byte 0**

Bit	Name	Reset	R/W	Description
7:0	STCV[7:0]	0x00	R	Bits [7:0] of Sleep Timer capture value

**STCV1 (0x62B3) – Sleep Timer Capture Value Byte 1**

Bit	Name	Reset	R/W	Description
7:0	STCV[15:8]	0x00	R	Bits [15:8] of Sleep Timer capture value

**STCV2 (0x62B4) – Sleep Timer Capture Value Byte 2**

Bit	Name	Reset	R/W	Description
7:0	STCV[23:16]	0x00	R	Bits [23:16] of Sleep Timer capture value

**ADC**

The ADC (in the CC2530, CC2531, CC2540, and CC2541) supports 14-bit analog-to-digital conversion with up to 12 effective number of bits (ENOB). It includes an analog multiplexer with up to eight individually configurable channels and a reference voltage generator. Conversion results can be written to memory through DMA. Several modes of operation are available.

Topic	Page
<b>12.1 ADC Introduction</b> .....	<b>133</b>
<b>12.2 ADC Operation</b> .....	<b>133</b>

## 12.1 ADC Introduction

The ADC supports up to 14-bit analog-to-digital conversion with up to 12 bits ENOB (Effective Number Of Bits). It includes an analog multiplexer with up to eight individually configurable channels and a reference voltage generator. Conversion results can be written to memory through DMA. Several modes of operation are available.

The main features of the ADC are as follows:

- Selectable decimation rates which also set the effective resolution (7 to 12 bits).
- Eight individual input channels, single-ended or differential
- Reference voltage selectable as internal, external single-ended, external differential, or AVDD5
- Interrupt request generation
- DMA triggers at end of conversions
- Temperature sensor input
- Battery measurement capability

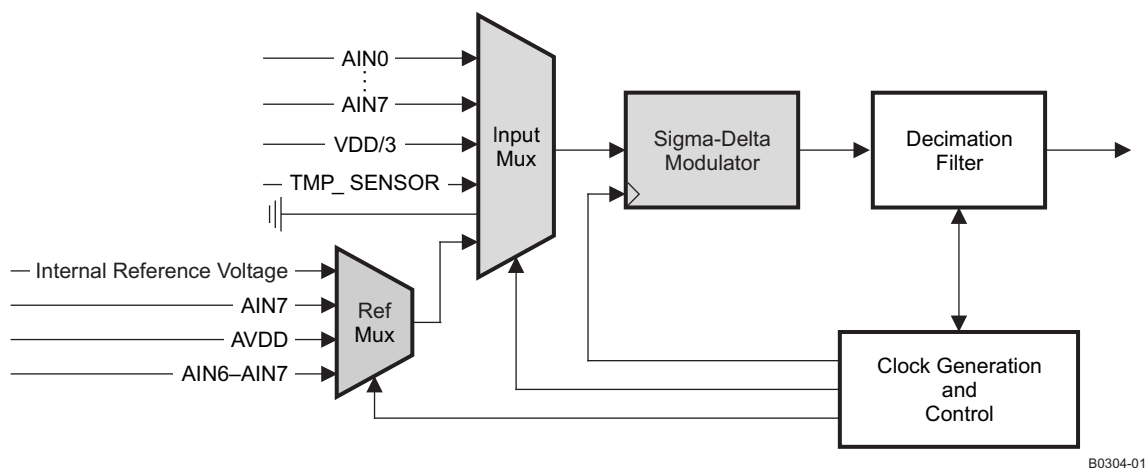


Figure 12-1. ADC Block Diagram

## 12.2 ADC Operation

This section describes the general setup and operation of the ADC and describes the use of the ADC control and status registers accessed by the CPU.

### 12.2.1 ADC Inputs

The signals on the port-0 pins can be used as ADC inputs. In the following, these port pins are referred to as the AIN0–AIN7 pins. The input pins AIN0–AIN7 are connected to the ADC.

It is possible to configure the inputs as single-ended or differential inputs. In the case where differential inputs are selected, the differential inputs consist of the input pairs AIN0–AIN1, AIN2–AIN3, AIN4–AIN5, and AIN6–AIN7. Note that no negative supply can be applied to these pins, nor a supply higher than VDD (unregulated power). It is the difference between the pins of each pair that is converted in differential mode.

In addition to the input pins AIN0–AIN7, the output of an on-chip temperature sensor can be selected as an input to the ADC for temperature measurements. In order to do so, the registers `TR0.ADCTM` and `ATEST.ATESTCTRL` must be set as described in [Section 12.2.10](#) and [Section 23.15.3](#) (CC253x) or [Section 24.1](#) (CC2540), or [Section 25.12](#) (CC2541).

It is also possible to select a voltage corresponding to AVDD5 divided by 3 as an ADC input. This input allows the implementation of, for example, a battery monitor in applications where this feature is required. Note that the reference in this case must not be dependent on the battery voltage; for instance, the AVDD5 voltage must not be used as a reference.

The single-ended inputs AIN0 through AIN7 are represented by channel numbers 0 to 7. Channel numbers 8 through 11 represent the differential inputs consisting of AIN0–AIN1, AIN2–AIN3, AIN4–AIN5, and AIN6–AIN7. Channel numbers 12 through 15 represent GND (12), temperature sensor (14), and AVDD5/3 (15), with channel 13 being reserved. These values are used in the `ADCCON2.SCH` and `ADCCON3.SCH` fields.

The ADC input is a switched capacitance stage which draws current during the conversion. As an example, the equivalent input impedance of a typical device was found to be 176 k $\Omega$  when used with an input voltage of 3 V, a 512 $\times$  decimation rate, and the internal reference.

To enable the temperature sensor as an input to the ADC, the `TR0.ADCTM` bit must be set to 1 before setting the `ATEST.ATESTCTRL` bit to 1. When disabling the temperature sensor as an input, the `ATEST.ATESTCTRL` bit must be set to 0 before clearing the `TR0.ADCTM` bit. The `TR0` register does not have any retention in PM2 or PM3, so `ATEST` and `TR0` must be cleared in the correct manner before entering these power modes.

### 12.2.2 ADC Conversion Sequences

The ADC can perform a sequence of conversions and move the results to memory (through DMA) without any interaction from the CPU.

The conversion sequence can be influenced with the `APCFG` register (see [Section 7.6.6](#)), in that the eight analog inputs to the ADC come from I/O pins that are not necessarily programmed to be analog inputs. If a channel should normally be part of a sequence, but the corresponding analog input is disabled in the `APCFG` register, then that channel is skipped. When using differential inputs, both pins in a differential pair must set as analog input pins in the `APCFG` register.

The `ADCCON2.SCH` register bits are used to define an ADC conversion sequence from the ADC inputs. If `ADCCON2.SCH` is set to a value less than 8, the conversion sequence contains a conversion from each channel from 0 up to and including the channel number programmed in `ADCCON2.SCH`. When `ADCCON2.SCH` is set to a value between 8 and 12, the sequence consists of differential inputs, starting at channel 8 and ending at the programmed channel. For `ADCCON2.SCH` greater than or equal to 12, the sequence consists of the selected channel only.

### 12.2.3 Single ADC Conversion

In addition to this sequence of conversions, the ADC can be programmed to perform a single conversion from any channel. Such a conversion is triggered by writing to the `ADCCON3` register. The conversion starts immediately unless a conversion sequence is already ongoing, in which case the single conversion is performed as soon as that sequence is finished.

### 12.2.4 ADC Operating Modes

This section describes the operating modes and initialization of conversions.

The ADC has three control registers: `ADCCON1`, `ADCCON2`, and `ADCCON3`. These registers are used to configure the ADC and to report status.

The `ADCCON1.EOC` bit is a status bit that is set high when a conversion ends and cleared when `ADCH` is read.

The `ADCCON1.ST` bit is used to start a sequence of conversions. A sequence starts when this bit is set high, `ADCCON1.STSEL` is 11, and no conversion is currently running. When the sequence is completed, this bit is automatically cleared.

The `ADCCON1.STSEL` bits select the event that starts a new sequence of conversions. The options which can be selected are rising edge on external pin P2.0, end of previous sequence, a Timer 1 channel 0 compare event, or `ADCCON1.ST` is 1.

The `ADCCON2` register controls how the sequence of conversions is performed.

`ADCCON2.SREF` is used to select the reference voltage. The reference voltage should only be changed when no conversion is running.

The `ADCCON2.SDIV` bits select the decimation rate, thereby also the resolution and time required to complete a conversion, and hence the sample rate. The decimation rate should only be changed when no conversion is running.

The last channel of a sequence is selected with the `ADCCON2.SCH` bits as described previously.

The `ADCCON3` register controls the channel number, reference voltage, and decimation rate for a single conversion. The single conversion takes place immediately after the `ADCCON3` register is written to, or if a conversion sequence is ongoing, immediately after the sequence has ended. The coding of the register bits is exactly as for `ADCCON2`.

### 12.2.5 ADC Conversion Results

The digital conversion result is represented in 2s-complement form. For single-ended configurations, the result can be expected to be positive. This is because the result is the difference between the input signal and ground, which is always positively signed ( $V_{conv} = V_{in} - V_{in0}$ , where  $V_{in0} = 0\text{ V}$ ). The maximum value is reached when the input signal is equal to  $V_{REF}$ , the selected voltage reference. For differential configurations, the difference between two pins is converted, and this difference can be negatively signed. For example, with a decimation rate of 512 using only the 12 MSBs of the digital conversion result register, the maximum value of 2047 is reached when the analog input ( $V_{conv}$ ) is equal to  $V_{REF}$ , and minimum value of  $-2048$  is reached when the analog input is equal to  $-V_{REF}$ .

The digital conversion result is available in `ADCH` and `ADCL` when `ADCCON1.EOC` is set to 1. Note that the conversion result always resides in the MSB section of the combined `ADCH` and `ADCL` registers.

When the `ADCCON2.SCH` bits are read, they indicate the channel on which conversion is ongoing. The results in `ADCL` and `ADCH` normally apply to the previous conversion. If the conversion sequence has ended, `ADCCON2.SCH` has a value of one more than the last channel number, but if the channel number last written to `ADCCON2.SCH` was 12 or more, the same value is read back.

### 12.2.6 ADC Reference Voltage

The positive reference voltage for analog-to-digital conversions is selectable as either an internally generated voltage, the `AVDD5` pin, an external voltage applied to the `AIN7` input pin, or a differential voltage applied to the `AIN6`–`AIN7` inputs.

The accuracy of the conversion results depend on the stability and noise properties of the reference voltage. Offset from the wanted voltage introduces a gain error in the ADC proportional to the ratio of the wanted voltage and the actual voltage. Noise on the reference must be lower than quantization noise of the ADC to ensure the specified SNR is achieved.

### 12.2.7 ADC Conversion Timing

The ADC should only be used with the 32-MHz `XOSC`, and no system clock division should be implemented by the user. The actual ADC sampling frequency of 4 MHz is generated by fixed internal division. The time required to perform a conversion depends on the selected decimation rate. In general, the conversion time is given by:

$$T_{conv} = (\text{decimation rate} + 16) \times 0.25 \mu\text{s}$$

### 12.2.8 ADC Interrupts

The ADC generates an interrupt when a single conversion triggered by writing to `ADCCON3` has completed. No interrupt is generated when a conversion from the sequence is completed.

### 12.2.9 ADC DMA Triggers

The ADC generates a DMA trigger every time a conversion from the sequence has completed. When a single conversion completes, no DMA trigger is generated.

There is one DMA trigger for each of the eight channels defined by the first eight possible settings for `ADCCON2.SCH`. The DMA trigger is active when a new sample is ready from the conversion for the channel. The DMA triggers are named `ADC_CHsd` in [Table 8-1](#), where **s** is single-ended channel and **d** is differential channel.

In addition, one DMA trigger, ADC\_CHALL, is active when new data is ready from any of the channels in the ADC conversion sequence.

### 12.2.10 ADC Registers

This section describes the ADC registers.

#### ADCL (0xBA) – ADC Data, Low

Bit	Name	Reset	R/W	Description
7:2	ADC[5:0]	0000 00	R	Least-significant part of ADC conversion result
1:0	–	00	R0	Reserved. Always read as 0

#### ADCH (0xBB) – ADC Data, High

Bit	Name	Reset	R/W	Description
7:0	ADC[13:6]	0x00	R	Most-significant part of ADC conversion result

#### ADCCON1 (0xB4) – ADC Control 1

Bit	Name	Reset	R/W	Description
7	EOC	0	R/H0	End of conversion. Cleared when ADCH has been read. If a new conversion is completed before the previous data has been read, the EOC bit remains high. 0: Conversion not complete 1: Conversion completed
6	ST	0	R/W1/H0	Start conversion. Read as 1 until conversion has completed 0: No conversion in progress 1: Start a conversion sequence if ADCCON1.STSEL = 11 and no sequence is running.
5:4	STSEL[1:0]	11	R/W	Start select. Selects the event that starts a new conversion sequence 00: External trigger on P2.0 pin 01: Full speed. Do not wait for triggers 10: Timer 1 channel 0 compare event 11: ADCCON1.ST = 1
3:2	–	00	R/W	Controls the 16-bit random-number generator. See <b>ADCCON1 (0xB4) – ADC Control 1</b> description in <a href="#">Section 14.3</a> .
1:0	–	11	R/W	Reserved. Always set to 11



**ADCCON2 (0xB5) – ADC Control 2**

Bit	Name	Reset	R/W	Description
7:6	SREF[1:0]	00	R/W	<p>Selects reference voltage used for the sequence of conversions</p> <p>00: Internal reference</p> <p>01: External reference on AIN7 pin</p> <p>10: AVDD5 pin</p> <p>11: External reference on AIN6–AIN7 differential input</p>
5:4	SDIV[1:0]	01	R/W	<p>Sets the decimation rate for channels included in the sequence of conversions. The decimation rate also determines the resolution and time required to complete a conversion.</p> <p>00: 64 decimation rate (7 bits ENOB setting)</p> <p>01: 128 decimation rate (9 bits ENOB setting)</p> <p>10: 256 decimation rate (10 bits ENOB setting)</p> <p>11: 512 decimation rate (12 bits ENOB setting)</p>
3:0	SCH[3:0]	0000	R/W	<p>Sequence channel select. Selects the end of the sequence. A sequence can either be from AIN0 to AIN7 (<math>SCH \leq 7</math>) or from differential input AIN0–AIN1 to AIN6–AIN7 (<math>8 \leq SCH \leq 11</math>). For other settings, only one conversion is performed.</p> <p>When read, these bits indicate the channel number on which a conversion is ongoing.</p> <p>0000: AIN0</p> <p>0001: AIN1</p> <p>0010: AIN2</p> <p>0011: AIN3</p> <p>0100: AIN4</p> <p>0101: AIN5</p> <p>0110: AIN6</p> <p>0111: AIN7</p> <p>1000: AIN0–AIN1</p> <p>1001: AIN2–AIN3</p> <p>1010: AIN4–AIN5</p> <p>1011: AIN6–AIN7</p> <p>1100: GND</p> <p>1101: Reserved</p> <p>1110: Temperature sensor</p> <p>1111: VDD / 3</p>

**ADCCON3 (0xB6) – ADC Control 3**

Bit	Name	Reset	R/W	Description
7:6	EREF[1:0]	00	R/W	Selects reference voltage used for the extra conversion 00: Internal reference 01: External reference on AIN7 pin 10: AVDD5 pin 11: External reference on AIN6–AIN7 differential input
5:4	EDIV[1:0]	00	R/W	Sets the decimation rate used for the extra conversion. The decimation rate also determines the resolution and the time required to complete the conversion. 00: 64 decimation rate (7 bits ENOB) 01: 128 decimation rate (9 bits ENOB) 10: 256 decimation rate (10 bits ENOB) 11: 512 decimation rate (12 bits ENOB)
3:0	ECH[3:0]	0000	R/W	Single channel select. Selects the channel number of the single conversion that is triggered by writing to ADCCON3. 0000: AIN0 0001: AIN1 0010: AIN2 0011: AIN3 0100: AIN4 0101: AIN5 0110: AIN6 0111: AIN7 1000: AIN0–AIN1 1001: AIN2–AIN3 1010: AIN4–AIN5 1011: AIN6–AIN7 1100: GND 1101: Reserved 1110: Temperature sensor 1111: VDD / 3

**TR0 (0x624B) – Test Register 0**

Bit	Name	Reset	R/W	Description
7:1	–	0000 000	R0	Reserved. Write as 0.
0	ADCTM	0	R/W	Set to 1 to connect the temperature sensor to the SOC_ADC. See also ATEST register description to enable the temperature sensor in <a href="#">Section 23.15.3</a> (CC253x) or <a href="#">Section 24.1</a> (CC2540) or <a href="#">Section 25.12.3</a> (CC2541).

## Battery Monitor

---

---

---

The battery monitor (in the CC2533 only) enables simple voltage monitoring in the devices that do not include an ADC. It is designed such that it is accurate in the voltage areas around 2 V, with lower resolution at higher voltages. The registers `BATTMON` and `MONMUX` are used to access and control the functionality of the battery monitor.

The battery monitor can also be used to do simple temperature monitoring by connecting it to the chip internal temperature sensor instead of the supply voltage. The input is controlled using the `MONMUX` register.

Topic	Page
<b>13.1 Functionality and Usage of the Battery Monitor</b> .....	<b>140</b>
<b>13.2 Using the Battery Monitor for Temperature Monitoring</b> .....	<b>140</b>
<b>13.3 Battery Monitor Registers</b> .....	<b>141</b>

### 13.1 Functionality and Usage of the Battery Monitor

The battery monitor makes it possible to check whether the supply voltage (AVDD5) is above or below a certain programmable level. Its usage is controlled by the BATTMON register in the following manner:

BATTMON\_VOLTAGE is used to set the trigger point for the battery monitor. Note the fact that the step size is different for different voltage ranges (see the register description in [Section 13.3](#) for details). This is done to achieve good accuracy in the voltage areas around 2 V, with lower resolution at higher voltages.

BATTMON\_PD is used to enable or disable the battery monitor.

After enabling the battery monitor by setting BATTMON\_PD = 0 and waiting for at least 2  $\mu$ s, the value of BATTMON\_OUT indicates whether the voltage is above or below the trigger point (set by BATTMON\_VOLTAGE).

---

**NOTE:** One should turn the battery monitor off (BATTMON\_PD = 1) after reading the measurement BATTMON\_OUT in order to save power, as the battery monitor consumes power when enabled (= 0).

---

Recommended usage of the battery monitor can be summarized in the following way:

1. Set BATTMON\_VOLTAGE to the value to be monitored.
2. Enable the battery monitor by setting BATTMON\_PD = 0.
3. Wait for at least 2  $\mu$ s.
4. Read the BATTMON\_OUT result to see whether the voltage level is above or below the value set in BATTMON\_VOLTAGE.
5. Disable the battery monitor (BATTMON\_PD = 1) to avoid unnecessary current consumption.

### 13.2 Using the Battery Monitor for Temperature Monitoring

The battery monitor can also be used to do some simple temperature monitoring. When the battery monitor is connected to the internal temperature sensor instead of the supply voltage AVDD5 (see the description of MONMUX in [Section 13.3](#)), it can indicate whether the temperature is above or below a certain level. This is done by comparing the voltage coming from the temperature sensor to the voltage trigger point of the battery monitor. The controls for this measurement are the same as for the normal use of the battery monitor (see the description of BATTMON in [Section 13.3](#)).

It is important to understand that due to the nature of the battery monitor (optimized for voltages around 2 V) and the output voltage range of the temperature sensor, there are only about 8 temperature trigger values in the temperature range of  $-40^{\circ}\text{C}$  to  $125^{\circ}\text{C}$  (see [Table 13-1](#)). As a result, the battery monitor gives only a rough indication of the temperature range, but this is useful for doing temperature compensation on analog components in a system. See the data sheet of the device ([Appendix C](#)) for details of performance characteristics.

**Table 13-1. Values Showing How Different Temperatures Relate to BATTMON\_VOLTAGE for a Typical Device**

Temperature	BATTMON_VOLTAGE
$-40^{\circ}\text{C}$	22
$-26^{\circ}\text{C}$	21
$-11^{\circ}\text{C}$	20
$7^{\circ}\text{C}$	19
$25^{\circ}\text{C}$	18
$47^{\circ}\text{C}$	17
$70^{\circ}\text{C}$	16
$97^{\circ}\text{C}$	15
$128^{\circ}\text{C}$	14

The temperature sensor is inversely proportional to `BATTMON_VOLTAGE`. The temperature (in °C) corresponding to a given `BATTMON_VOLTAGE` is given by:

$$\text{Temp} = \frac{A}{\text{BATTMON\_VOLTAGE}\langle 4:0 \rangle} - B \quad (1)$$

Assuming `BATTMON_VOLTAGE` < 27, and only valid for  $-40^{\circ}\text{C} < \text{Temp} < 125^{\circ}\text{C}$ , A and B for a typical device are given in [Table 13-2](#).

**Table 13-2. Values for A and B (for a Typical Device)  
When Using the Battery monitor for Temperature  
Monitoring**

Constant	Typ
A	6470
B	334

Note that A should be relatively constant for all devices, but B is not. Information that can be used to calculate B for a given chip is included in the chip's information page (see [Section 2.2.3](#) for information about the information page).

Example:

Find the `BATTMON_VOLTAGE` setting that tells whether the temperature is above or below  $75^{\circ}\text{C}$ .

$$\text{BATTMON\_VOLTAGE}\langle 4:0 \rangle = \frac{6470}{75 + 334} = 15.82 \quad (2)$$

The closest setting is 16, which corresponds to approximately  $70^{\circ}\text{C}$  (see [Table 13-1](#)). By writing 16 to `BATTMON_VOLTAGE`, an output of `BATTMON_OUT` = 1 tells that the temperature is above  $70^{\circ}\text{C}$ , whereas `BATTMON_OUT` = 0 tells that it is below  $70^{\circ}\text{C}$ .

### 13.3 Battery Monitor Registers

This section describes the battery monitor registers.

**BATTMON (0x6264) – Battery Monitor**

Bit	Name	Reset	R/W	Description
7	–	0	R0	Reserved. Always read 0
6	BATTMON_OUT	0	R	Result from the battery monitor. 1: Voltage is above value set in BATTMON_VOLTAGE . 0: Voltage is below the value set in BATTMON_VOLTAGE . Note that the value of BATTMON_OUT is undefined except when BATTMON_PD is 0 and has been 0 for 2 $\mu$ s.
5:1	BATTMON_VOLTAGE	11 100	R/W	Controls the trigger point for the battery monitor. The step size is 24 mV for the first 23 settings, and then 169 mV (unless temperature-sense mode is enabled; see <a href="#">Section 13.2</a> for details). Range to be used: 3–31 3: 1.93 V 4: $1.93\text{ V} + (4 - 3) \times 0.024\text{ V} = 1.954\text{ V}$ 5: $1.93\text{ V} + (5 - 3) \times 0.024\text{ V} = 1.978\text{ V}$ 6: $1.93\text{ V} + (6 - 3) \times 0.024\text{ V} = 2.002\text{ V}$ 7: $1.93\text{ V} + (7 - 3) \times 0.024\text{ V} = 2.026\text{ V}$ 8: $1.93\text{ V} + (8 - 3) \times 0.024\text{ V} = 2.050\text{ V}$ 9: $1.93\text{ V} + (9 - 3) \times 0.024\text{ V} = 2.074\text{ V}$ 10: $1.93\text{ V} + (10 - 3) \times 0.024\text{ V} = 2.098\text{ V}$ 11: $1.93\text{ V} + (11 - 3) \times 0.024\text{ V} = 2.122\text{ V}$ 12: $1.93\text{ V} + (12 - 3) \times 0.024\text{ V} = 2.146\text{ V}$ 13: $1.93\text{ V} + (13 - 3) \times 0.024\text{ V} = 2.170\text{ V}$ 14: $1.93\text{ V} + (14 - 3) \times 0.024\text{ V} = 2.194\text{ V}$ 15: $1.93\text{ V} + (15 - 3) \times 0.024\text{ V} = 2.218\text{ V}$ 16: $1.93\text{ V} + (16 - 3) \times 0.024\text{ V} = 2.242\text{ V}$ 17: $1.93\text{ V} + (17 - 3) \times 0.024\text{ V} = 2.266\text{ V}$ 18: $1.93\text{ V} + (18 - 3) \times 0.024\text{ V} = 2.290\text{ V}$ 19: $1.93\text{ V} + (19 - 3) \times 0.024\text{ V} = 2.314\text{ V}$ 20: $1.93\text{ V} + (20 - 3) \times 0.024\text{ V} = 2.338\text{ V}$ 21: $1.93\text{ V} + (21 - 3) \times 0.024\text{ V} = 2.362\text{ V}$ 22: $1.93\text{ V} + (22 - 3) \times 0.024\text{ V} = 2.386\text{ V}$ 23: $1.93\text{ V} + (23 - 3) \times 0.024\text{ V} = 2.410\text{ V}$ 24: $1.93\text{ V} + (24 - 3) \times 0.024\text{ V} = 2.434\text{ V}$ 25: $1.93\text{ V} + (25 - 3) \times 0.024\text{ V} = 2.458\text{ V}$ 26: $1.93\text{ V} + (26 - 3) \times 0.024\text{ V} = 2.482\text{ V}$ 27: $2.482\text{ V} + (27 - 26) \times 0.169\text{ V} = 2.651\text{ V}$ 28: $2.482\text{ V} + (28 - 26) \times 0.169\text{ V} = 2.820\text{ V}$ 29: $2.482\text{ V} + (29 - 26) \times 0.169\text{ V} = 2.989\text{ V}$ 30: $2.482\text{ V} + (30 - 26) \times 0.169\text{ V} = 3.158\text{ V}$ 31: $2.482\text{ V} + (31 - 26) \times 0.169\text{ V} = 3.327\text{ V}$
0	BATTMON_PD	1	R/W	Turns on the battery monitor. Wait at least 2 $\mu$ s before reading BATTMON_OUT. 0: Enable the battery monitor. 1: Disable the battery monitor. One should turn the battery monitor off (BATTMON_PD = 1) after reading out the measurement BATTMON_OUT in order to save power, as the battery monitor consumes power when enabled (BATTMON_PD = 0).

**MONMUX (0x61A6) – Monitor MUX**

Bit	Name	Reset	R/W	Description
7:1	–	–	–	Reserved
0	BATTMON_INPUT	0	R/W	Determines the input to the battery monitor: 0: Supply voltage (AVDD5) 1: Voltage from the temperature sensor, which must be enabled using the ATEST.ATEST_CTRL register; described in <a href="#">Section 23.15.3</a>

## Random-Number Generator

---

---

---

This chapter provides information about the random-number generator and its usage.

Topic	Page
14.1 Introduction .....	144
14.2 Random-Number-Generator Operation .....	144
14.3 Random-Number-Generator Registers .....	145

## 14.1 Introduction

The random-number generator has the following features.

- Generates pseudorandom bytes which can be read by the CPU or used directly by the command strobe processor (see [Section 23.14](#))
- Calculates CRC16 of bytes that are written to RNDH
- Seeded by value written to RNDL

The random-number generator is a 16-bit linear-feedback shift register (LFSR) with polynomial  $X^{16} + X^{15} + X^2 + 1$  (that is, CRC16). It uses different levels of unrolling depending on the operation it performs. The basic version (no unrolling) is shown in [Figure 14-1](#).

The random-number generator is turned off when `ADCCON1.RCTRL = 11`.

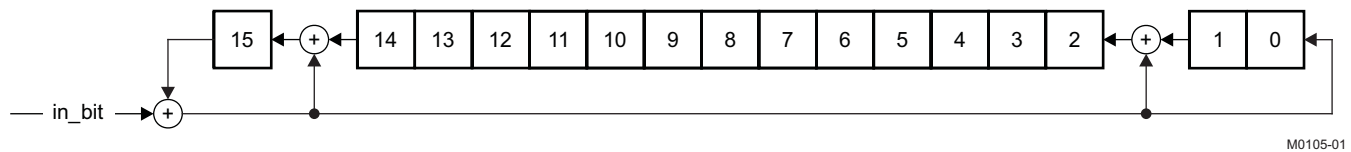


Figure 14-1. Basic Structure of the Random-Number Generator

## 14.2 Random-Number-Generator Operation

The operation of the random-number generator is controlled by the `ADCCON1.RCTRL` bits (see also [Section 14.3](#)). The current value of the 16-bit shift register in the LFSR can be read from the `RNDH` and `RNDL` registers.

### 14.2.1 Pseudorandom Sequence Generation

The default operation (`ADCCON1.RCTRL` is 00) is to clock the LFSR once (13x unrolling; where clocking with 13x unrolling means performing an operation equivalent to doing 13 shifts with feedback) each time the command strobe processor ([Section 23.14](#)) reads the random value. This leads to the availability of a fresh pseudorandom byte from the LSB end of the LFSR.

Another way to update the LFSR is to set `ADCCON1.RCTRL` to 01. This clocks the LFSR once (13x unrolling), and the `ADCCON1.RCTRL` bits are automatically cleared when the operation has completed.

### 14.2.2 Seeding

The LFSR can be seeded by writing to the `RNDL` register twice. Each time the `RNDL` register is written, the 8 LSBs of the LFSR are copied to the 8 MSBs and the 8 LSBs are replaced with the new data byte that was written to `RNDL`.

For the CC253x, when a random value is required, the LFSR should be seeded by writing `RNDL` with random bits from the `IF_ADC` in the RF receive path. To use this seeding method, the radio must first be powered on. The radio should be placed in the infinite RX state to avoid possible sync detect in the RX state. The random bits from the `IF_ADC` are read from the least-significant bit position of the RF register `RFRND`. These bits should be concatenated over time to form the bytes needed for the random-number-generator seed. See [Section 23.12](#) for a description of the randomness of these numbers. Note that this cannot be done while the radio is in use for normal tasks.

Note that a seed value of 0x0000 or 0x8003 always leads to an unchanged value in the LFSR after clocking, as no values are pushed in via `in_bit` (see [Figure 14-1](#)); hence, neither of these seed values should be used for random-number generation.

### 14.2.3 CRC16

The LFSR can also be used to calculate the CRC value of a sequence of bytes. Writing to the `RNDH` register triggers a CRC calculation. The new byte is processed from the MSB end and an 8x unrolling is used, so that a new byte can be written to `RNDH` every clock cycle.



Note that the LFSR must be properly seeded by writing to RNDL before the CRC calculations start. Usually, the seed value for CRC calculations should be 0x0000 or 0xFFFF.

### 14.3 Random-Number-Generator Registers

This section describes the random-number-generator registers.

#### RNDL (0xBC) – Random-Number-Generator Data, Low Byte

Bit	Name	Reset	R/W	Description
7:0	RNDL[7:0]	0xFF	R/W	<p>Random value, random seed, or CRC result, low byte</p> <p>When used for random-number generation, writing to this register twice seeds the random-number generator. Writing to this register copies the 8 LSBs of the LFSR to the 8 MSBs and replaces the 8 LSBs with the data value written.</p> <p>The value returned when reading from this register is the 8 LSBs of the LFSR.</p> <p>When used for random-number generation, reading this register returns the 8 LSBs of the random number. When used for CRC calculations, reading this register returns the 8 LSBs of the CRC result.</p>

#### RNDH (0xBD) – Random-Number-Generator Data, High Byte

Bit	Name	Reset	R/W	Description
7:0	RNDH[7:0]	0xFF	R/W	<p>Random value, CRC result, or input data, high byte</p> <p>When written, a CRC16 calculation is triggered, and the data value written is processed starting with the MSB.</p> <p>The value returned when reading from this register is the 8 MSBs of the LFSR.</p> <p>When used for random-number generation, reading this register returns the 8 MSBs of the random number. When used for CRC calculations, reading this register returns the 8 MSBs of the CRC result.</p>

#### ADCCON1 (0xB4) – ADC Control 1 (see also [Section 12.2.10](#))

Bit	Name	Reset	R/W	Description
7:4	–	0011	–	For CC2533, these bits are reserved. For the other devices, see the <b>ADCCON1 (0xB4) – ADC Control 1</b> description in <a href="#">Section 12.2.10</a> .
3:2	RCTRL[1:0]	00	R/W	<p>Controls the 16-bit random-number generator (<a href="#">Chapter 14</a>). When 01 is written, the setting automatically returns to 00 when the operation has completed.</p> <p>00: Normal operation. (13× unrolling)</p> <p>01: Clock the LFSR once (13× unrolling)</p> <p>10: Reserved</p> <p>11: Stopped. Random-number generator is turned off.</p>
1:0	–	11	R/W	Reserved. Always set to 11

## AES Coprocessor

The Advanced Encryption Standard (AES) coprocessor allows encryption or decryption to be performed with minimal CPU usage.

The coprocessor has the following features:

- Supports all security suites in IEEE 802.15.4
- ECB, CBC, CFB, OFB, CTR, and CBC-MAC modes
- Hardware support for CCM mode
- 128-bit key and IV/nonce
- DMA transfer trigger capability

Topic	Page
15.1 AES Operation .....	147
15.2 Key and IV .....	147
15.3 Padding of Input Data .....	147
15.4 Interface to CPU .....	147
15.5 Modes of Operation .....	147
15.6 CBC-MAC .....	147
15.7 CCM Mode .....	148
15.8 AES Interrupts.....	150
15.9 AES DMA Triggers.....	150
15.10 AES Registers.....	150

## 15.1 AES Operation

To encrypt a message, the following procedure must be followed (ECB, CBC):

- Load key
- Load initialization vector (IV)
- Download and upload data for encryption or decryption.

The AES coprocessor works on blocks of 128 bits. A block of data is loaded into the coprocessor, encryption is performed, and the result must be read out before the next block can be processed. Before each block is loaded, a dedicated start command must be sent to the coprocessor.

## 15.2 Key and IV

Before a key or IV/nonce load starts, an appropriate load key or IV/nonce command must be issued to the coprocessor. When loading the IV, it is important also to set the correct mode.

A key load or IV load operation aborts any processing that could be running. The key, once loaded, stays valid until a key reload takes place.

The IV must be downloaded before the beginning of each message (not each block).

Both the key and IV values are cleared by a reset of the device and when PM2 or PM3 is entered.

## 15.3 Padding of Input Data

The AES coprocessor works on blocks of 128 bits. If the last block contains less than 128 bits, it must be padded with zeros when written to the coprocessor.

## 15.4 Interface to CPU

The CPU communicates with the coprocessor using three SFR registers:

- ENCCS, encryption control and status register
- ENCDI, encryption input register
- ENCDO, encryption output register

Read or write to the status register is done directly by the CPU, whereas access to the input/output registers should be performed using direct memory access (DMA).

When using DMA with the AES coprocessor, two DMA channels must be used, one for input data and one for output data. The DMA channels must be initialized before a start command is written to ENCCS. Writing a start command generates a DMA trigger, and the transfer is started. After each block is processed, an interrupt is generated. The interrupt is used to issue a new start command to ENCCS.

## 15.5 Modes of Operation

When using CFB, OFB, or CTR mode, the 128-bit blocks are divided into four 32-bit blocks. The 32 bits are loaded into the AES coprocessor, and the resulting 32 bits are read out. This continues until all 128 bits have been encrypted. The only time one must consider this is if data is loaded or read directly using the CPU. When using DMA, this is handled automatically by the DMA triggers generated by the AES coprocessor; thus, DMA is preferred.

Both encryption and decryption are performed similarly.

The CBC-MAC mode is a variant of the CBC mode. See [Section 15.6](#) for an explanation.

CCM is a combination of CBC-MAC and CTR. Parts of the CCM must therefore be done in software. The following section gives a short explanation of the necessary steps to be done.

## 15.6 CBC-MAC

When performing CBC-MAC encryption, data is downloaded to the coprocessor in CBC-MAC mode one block at a time, except for the last block. Before the last block is loaded, the mode is changed to CBC. The last block is downloaded and the block uploaded is the message MAC.

CBC-MAC decryption is similar to encryption. The message MAC uploaded must be compared with the MAC to be verified.

## 15.7 CCM Mode

To encrypt a message in CCM mode, the following sequence can be conducted (key is already loaded):

### Message Authentication Phase

This phase takes place during the following steps 1–6.

1. The software loads the IV with zeros.
2. The software creates block B0. The layout of block B0 is shown in [Figure 15-1](#).

Name		Designation														
B0		First Block for Authentication in CCM Mode														
Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Name	Flag	Nonce									L_M					

**Figure 15-1. Message Authentication Phase Block B0**

There is no restriction on the nonce value. L\_M is the message length in bytes.

For 802.15.4, nonce is 13 bytes and L\_M is 2 bytes.

The content of the authentication flag byte is described in [Figure 15-2](#).

L is set to 6 in this example. So, L – 1 is set to 5. M and A\_Data can be set to any value.

Name		Designation						
FLAG-B0		Authentication Flag Field for CCM mode						
Bit	7	6	5	4	3	2	1	0
Name	Reserved	A_Data	(M – 2) / 2			L – 1		
Value	0	x	x	x	x	1	0	1

**Figure 15-2. Authentication Flag Byte**

3. If some additional authentication data (denoted a, following) is needed (that is, A\_Data = 1), the software creates the A\_Data length field, called L(a) by:
  - (a) If  $l(a) = 0$ , (that is, A\_Data = 0), then L(a) is the empty string. Note that  $l(a)$  is the length of a in octets.
  - (b) If  $0 < l(a) < 2^{16} - 2^8$ , then L(a) is the 2-octet encoding of  $l(a)$ .

The additional authentication data is appended to the A\_Data length field L(a). The additional authentication blocks are padded with zeros until the last additional authentication block is full. There is no restriction on the length of a.

AUTH-DATA = L(a) + Authentication Data + (zero padding)

4. The last block of the message is padded with zeros until full (that is, if its length is not an integral multiple of 128 bits).
5. The software concatenates block B0, the additional authentication blocks if any, and the message; Input message = B0 + AUTH-DATA + Message + (zero padding of message)
6. Once the input message authentication by CBC-MAC is finished, the software leaves the uploaded buffer contents unchanged (M = 16), or keeps only the higher-M bytes of the buffer unchanged, while setting the lower bits to 0 (M != 16).

The result is called T.

### Message Encryption

7. The software creates the key stream block A0. Note that L = 6, with the current example of the CTR generation. The content is shown in [Figure 15-3](#).

Note that when encrypting authentication data T to generate U in OFB mode, the CTR value must be zero. When encrypting message blocks using CTR mode, the CTR value must be any value but zero.

The content of the encryption-flag byte is described in [Figure 15-4](#).

Byte	Name A0					Designation First CTR Value for CCM Mode										
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Name	Flag	Nonce										CTR				

**Figure 15-3. Message Encryption Phase Block**

Bit	Name FLAG-A0		Designation Encryption Flag Field for CCM Mode					
	7	6	5	4	3	2	1	0
Name	Reserved		—			L – 1		
Value	0	0	0	0	0	1	0	1

**Figure 15-4. Encryption Flag Byte**

8. The software loads A0 by selecting a Load IV/nonce command. To do so, it sets the mode to CFB or OFB at the same time it selects the Load IV/nonce command.
9. The software calls a CFB or an OFB encryption on the authenticated data T. The uploaded buffer contents stay unchanged ( $M = 16$ ), or only its first M bytes stay unchanged, the others being set to 0 ( $M - 16$ ). The result is U, which is used later.
10. The software calls a CTR-mode encryption immediately on the still-padded message blocks. It must reload the IV when the CTR value is any value but zero.
11. The encrypted authentication data U is appended to the encrypted message. This gives the final result, C.

$$\text{Result C} = \text{encrypted message(m)} + \text{U}$$

### **Message Decryption**

#### CCM Mode Decryption

In the coprocessor, the automatic generation of CTR works on 32 bits; therefore, the maximum length of a message is  $128 \times 2^{32}$  bits, that is  $2^{36}$  bytes, which can be written in a 6-bit word. So, the value L is set to 6. To decrypt a CCM-mode processed message, the following sequence can be conducted (key is already loaded).

#### **Message Parsing Phase**

1. The software parses the message by separating the M rightmost octets, namely U, and the other octets, namely string C.
2. C is padded with zeros until it can fill an integral number of 128-bit blocks.
3. U is padded with zeros until it can fill a 128-bit block.
4. The software creates the key stream block A0. It is done the same way as for CCM encryption.
5. The software loads A0 by selecting a Load IV/nonce command. To do so, it sets the mode to CFB or OFB at the same time as it selects the IV load.
6. The software calls a CFB or an OFB encryption on the encrypted authenticated data U. The uploaded buffer contents stay unchanged (M = 16), or only its first M bytes stay unchanged, the others being set to 0 (M != 16). The result is T.
7. The software calls a CTR-mode decryption immediately on the encrypted message blocks C. Reloading the IV/CTR is not necessary.

#### **Reference Authentication Tag Generation**

This phase is identical to the authentication phase of CCM encryption. The only difference is that the result is named MACTag (instead of T).

#### **Message Authentication Checking Phase**

The software compares T with MACTag.

## **15.8 AES Interrupts**

The AES interrupt, ENC, is produced when encryption or decryption of a block is completed. The interrupt enable bit is `IEN0.ENCIE`, and the interrupt flag is `SOCON.ENCIF`.

## **15.9 AES DMA Triggers**

Two DMA triggers are associated with the AES coprocessor. These are ENC\_DW, which is active when input data must be downloaded to the ENCDI register, and ENC\_UP, which is active when output data must be uploaded from the ENCDO register.

The ENCDI and ENCDO registers should be set as destination and source locations for DMA channels used to transfer data to or from the AES coprocessor.

## **15.10 AES Registers**

The AES coprocessor registers have the layout shown in this section.

The registers return to their reset value when the chip enters PM2 or PM3.

**ENCCS (0xB3) – Encryption Control and Status**

Bit	Name	Reset	R/W	Description
7	–	0	R0	Reserved, always read as 0
6:4	MODE[2:0]	000	R/W	Encryption/decryption mode 000: CBC 001: CFB 010: OFB 011: CTR 100: ECB 101: CBC MAC 110: Reserved 111: Reserved
3	RDY	1	R	Encryption or decryption ready status 0: Encryption or decryption in progress 1: Encryption or decryption is completed.
2:1	CMD[1:0]	0	R/W	Command to be performed when a 1 is written to ST 00: Encrypt block 01: Decrypt block 10: Load key 11: Load IV/nonce
0	ST	0	R/W1 H0	Start processing command set by CMD. Must be issued for each command or 128-bit block of data. Cleared by hardware.

**ENCDI (0xB1) – Encryption Input Data**

Bit	Name	Reset	R/W	Description
7:0	DIN[7:0]	0x00	R/W	Encryption input data

**ENCDO (0xB2) – Encryption Output Data**

Bit	Name	Reset	R/W	Description
7:0	DOU[7:0]	0x00	R/W	Encryption output data

## Watchdog Timer

The Watchdog Timer (WDT) is intended as a recovery method in situations where the CPU may be subjected to a software upset. The WDT resets the system when software fails to clear the WDT within the selected time interval. The watchdog can be used in applications that are subject to electrical noise, power glitches, electrostatic discharge, and so forth, or where high reliability is required. If the watchdog function is not needed in an application, it is possible to configure the Watchdog Timer to be used as an interval timer that can be used to generate interrupts at selected time intervals.

The features of the Watchdog Timer are as follows:

- Four selectable timer intervals
- Watchdog mode
- Timer mode
- Interrupt request generation in timer mode

The WDT is configured as either a Watchdog Timer or as a timer for general-purpose use. The operation of the WDT module is controlled by the `WDCTL` register. The Watchdog Timer consists of a 15-bit counter clocked by the 32-kHz clock source. Note that the contents of the 15-bit counter are not user-accessible. The contents of the 15-bit counter are retained during all power modes, and the Watchdog Timer continues counting when entering active mode again.

Topic	Page
<b>16.1 Watchdog Mode</b> .....	<b>153</b>
<b>16.2 Timer Mode</b> .....	<b>153</b>
<b>16.3 Watchdog Timer Register</b> .....	<b>153</b>



## 16.1 Watchdog Mode

The WDT is disabled after a system reset. To start the WDT in watchdog mode, the `WDCTL.MODE[1:0]` bits must be set to 10. The Watchdog Timer counter then starts incrementing from 0. When the timer is enabled in watchdog mode, it is not possible to disable the timer. Therefore, writing 00 or 01 to `WDCTL.MODE[1:0]` has no effect if the WDT is already operating in Watchdog mode.

The WDT operates with a Watchdog Timer clock frequency of 32.768 kHz (when the 32-kHz XOSC is used). This clock frequency gives time-out periods equal to 1.9 ms, 15.625 ms, 0.25 s, and 1 s, corresponding to the count value settings 64, 512, 8192, and 32,768, respectively.

If the counter reaches the selected timer interval value, the Watchdog Timer generates a reset signal for the system. If a watchdog clear sequence is performed before the counter reaches the selected timer interval value, the counter is reset to 0 and continues incrementing its value. The watchdog clear sequence consists of writing 0xA to `WDCTL.CLR[3:0]`, followed by writing 0x5 to the same register bits within one watchdog clock period. If this complete sequence is not performed before the end of the watchdog period, the Watchdog Timer generates a reset signal for the system.

When the WDT has been enabled in watchdog mode, it is not possible to change the mode by writing to the `WDCTL.MODE[1:0]` bits, and the timer interval value cannot be changed.

In watchdog mode, the WDT does not produce interrupt requests.

## 16.2 Timer Mode

To start the WDT in timer mode, the `WDCTL.MODE[1:0]` bits must be set to 11. The timer is started and the counter starts incrementing from 0. When the counter reaches the selected interval value, the timer produces an interrupt request (`IRCON2.WDTIF` or `IEN2.WDTIE`). When the counter reaches the selected interval value, the timer produces an interrupt request, using `IRCON2.WDTIF` as the interrupt flag and `IEN2.WDTIE` as the interrupt mask.

In timer mode, it is possible to clear the timer contents by writing a 1 to `WDCTL.CLR[0]`. When the timer is cleared, the content of the counter is set to 0. Writing 00 to `WDCTL.MODE[1:0]` stops the timer and clears it to 0.

The timer interval is set by the `WDCTL.INT[1:0]` bits. The interval cannot be changed during timer operation, and should be set when the timer is started. In timer mode, a reset is not produced when the timer interval has been reached.

Note that if the watchdog mode is selected, the timer mode cannot be selected before the chip is reset.

## 16.3 Watchdog Timer Register

This section describes the register, `WDCTL`, for the Watchdog Timer.

**WDCTL (0xC9) – Watchdog Timer Control**

Bit	Name	Reset	R/W	Description
7:4	CLR[3:0]	0000	R0/W	Clear timer. In watchdog mode, when 0xA followed by 0x5 is written to these bits, the timer is cleared (that is, loaded with 0). Note that the timer is only cleared when 0x5 is written within one watchdog clock period after 0xA was written. Writing these bits when the Watchdog Timer is IDLE has no effect. When operating in timer mode, the timer can be cleared to 0x0000 (but not stopped) by writing 1 to CLR[0] (the other 3 bits are don't care).
3:2	MODE[1:0]	00	R/W	Mode select. These bits are used to start the WDT in watchdog mode or timer mode. Setting these bits to IDLE stops the timer when in timer mode. Note: to switch to watchdog mode when operating in timer mode, first stop the WDT - then start the WDT in Watchdog mode. When operating in Watchdog mode, writing these bits has no effect.  00: IDLE 01: Reserved 10: Watchdog mode 11: Timer mode
1:0	INT[1:0]	00	R/W	Timer interval select. These bits select the timer interval, which is defined as a given number of 32-kHz oscillator periods. Note that the interval can only be changed when the WDT is IDLE, so the interval must be set at the same time as the timer is started.  00: Clock period × 32,768 (approximately 1 s) when running the 32-kHz XOSC 01: Clock period × 8192 (approximately 0.25 s) 10: Clock period × 512 (approximately 15.625 ms) 11: Clock period × 64 (approximately 1.9 ms)  For CC253x and CC2540, when clock division is enabled through CLKCONCMD.CLKSPD, the length of the watchdog timer interval is reduced by a factor equal to the current oscillator clock frequency divided by the set clock speed. For example, if the 32-MHz crystal is selected and the clock speed is set to 4 MHz, then the watchdog timeout is reduced by a factor of 32 MHz / 4 MHz = 8. If the watchdog interval set by WDCTL.INT was 1 s, nominally it is 1 / 8 s with this clock division factor. For CC2541, the watchdog timer interval is independent of the clock division rate.

# USART

USART 0 and USART 1 are serial communications interfaces that can be operated separately in either asynchronous UART mode or in synchronous SPI mode. The two USARTs have identical functions, and are assigned to separate I/O pins. See [Section 7.6](#) for I/O configuration.

<b>Topic</b>	<b>Page</b>
17.1 <b>UART Mode</b> .....	<b>156</b>
17.2 <b>SPI Mode</b> .....	<b>157</b>
17.3 <b>SSN Slave-Select Pin</b> .....	<b>158</b>
17.4 <b>Baud-Rate Generation</b> .....	<b>158</b>
17.5 <b>USART Flushing</b> .....	<b>159</b>
17.6 <b>USART Interrupts</b> .....	<b>159</b>
17.7 <b>USART DMA Triggers</b> .....	<b>159</b>
17.8 <b>USART Registers</b> .....	<b>159</b>

## 17.1 UART Mode

For asynchronous serial interfaces, the UART mode is provided. In the UART mode, the interface uses a two-wire or four-wire interface consisting of the pins RXD and TXD, and optionally RTS and CTS. The UART mode of operation includes the following features:

- 8 or 9 payload bits
- Odd, even, or no parity
- Configurable start- and stop-bit levels
- Configurable LSB- or MSB-first transfer
- Independent receive and transmit interrupts
- Independent receive and transmit DMA triggers
- Parity and framing error status

The UART mode provides full-duplex asynchronous transfers, and the synchronization of bits in the receiver does not interfere with the transmit function. A UART byte transfer consists of a start bit, eight data bits, an optional ninth data or parity bit, and one or two stop bits. Note that the data transferred is referred to as a byte, although the data can actually consist of eight or nine bits.

The UART operation is controlled by the USART control and status registers, `UxCSR`, and the UART control registers, `UxUCR`, where `x` is the USART number, 0 or 1.

The UART mode is selected when `UxCSR.MODE` is set to 1.

### 17.1.1 UART Transmit

A UART transmission is initiated when the USART receive- and transmit-data buffers, `UxDBUF`, are written. The byte is transmitted on the TXD<sub>x</sub> output pins. The `UxDBUF` registers are double-buffered.

The `UxCSR.ACTIVE` bit goes high when the byte transmission starts and low when it ends. When the transmission ends, the `UxCSR.TX_BYTE` bit is set to 1. An interrupt request is generated when the `UxDBUF` register is ready to accept new transmit data. This happens immediately after the transmission has been started; hence, a new data byte value can be loaded into the data buffer while the byte is being transmitted.

### 17.1.2 UART Receive

Data reception on the UART is initiated when a 1 is written to the `UxCSR.RE` bit. The UART then searches for a valid start bit on the RXD<sub>x</sub> input pin and sets the `UxCSR.ACTIVE` bit high. When a valid start bit has been detected, the received byte is shifted into the receive register. The `UxCSR.RX_BYTE` bit is set and a receive interrupt is generated when the operation has completed. At the same time, `UxCSR.ACTIVE` goes low.

The received data byte is available through the `UxDBUF` register. When `UxDBUF` is read, `UxCSR.RX_BYTE` is cleared by hardware.

---

**NOTE:** When the application has read `UxDBUF`, it is important that it does not clear `UxCSR.RX_BYTE`. Clearing `UxCSR.RX_BYTE` implicitly makes the UART believe that the UART RX shift register is empty, even though it might hold pending data (typically due to back-to-back transmission). Consequently, the UART asserts (TTL low) the RT/RTS line, which allows flow into the UART, leading to potential overflow. Hence, the `UxCSR.RX_BYTE` flag integrates closely with the automatic RT/RTS function and must therefore be controlled solely by the SoC UART itself. Otherwise, the application could typically experience that the RT/RTS line remains asserted (TTL low), even though a back-to-back transmission clearly suggests it ought to intermittently pause the flow.

---

### 17.1.3 UART Hardware Flow Control

Hardware flow control is enabled when the `UxUCR.FLOW` bit is set to 1. The RTS output is driven low when the receive register is empty and reception is enabled. Transmission of a byte does not occur before the CTS input goes low.

### 17.1.4 UART Character Format

If the `BIT9` and `PARITY` bits in register `UxUCR` are set high, parity generation and detection is enabled. The parity is computed and transmitted as the ninth bit, and during reception, the parity is computed and compared to the received ninth bit. If there is a parity error, the `UxCSR.ERR` bit is set high. This bit is cleared when `UxCSR` is read.

The number of stop bits to be transmitted is set to one or two bits, as determined by the register bit `UxUCR.SPB`. The receiver always checks for one stop bit. If the first stop bit received during reception is not at the expected stop bit level, a framing error is signaled by setting register bit `UxCSR.FE` high. `UxCSR.FE` is cleared when `UxCSR` is read. The receiver checks both stop bits when `UxUCR.SPB` is set. Note that the RX interrupt is set when the first stop bit is checked OK. If second stop bit is not OK, there is a delay in setting the framing error bit, `UxCSR.FE`. This delay is baud-rate dependent (bit duration).

## 17.2 SPI Mode

This section describes the SPI mode of operation for synchronous communication. In SPI mode, the USART communicates with an external system through a three-wire or four-wire interface. The interface consists of the pins `MOSI`, `MISO`, `SCK`, and `SS_N`. See [Section 7.6](#) for a description of how the USART pins are assigned to the I/O pins.

The SPI mode includes the following features:

- Three-wire (master) and four-wire SPI interface
- Master and slave modes
- Configurable `SCK` polarity and phase
- Configurable LSB- or MSB-first transfer

The SPI mode is selected when `UxCSR.MODE` is set to 0.

In SPI mode, the USART can be configured to operate either as a SPI master or as a SPI slave by writing the `UxCSR.SLAVE` bit.

### 17.2.1 SPI Master Operation

A SPI byte transfer in master mode is initiated when the `UxDBUF` register is written. The USART generates the `SCK` serial clock using the baud-rate generator (see [Section 17.4](#)) and shifts the provided byte from the transmit register onto the `MOSI` output. At the same time, the receive register shifts in the received byte from the `MISO` input pin.

The `UxCSR.ACTIVE` bit goes high when the transfer starts and low when the transfer ends. When the transfer ends, the `UxCSR.TX_BYTE` bit is set to 1.

The polarity and clock phase of the serial clock `SCK` is selected by `UxGCR.CPOL` and `UxGCR.CPHA`. The order of the byte transfer is selected by the `UxGCR.ORDER` bit.

At the end of the transfer, the received data byte is available for reading from the `UxDBUF`. A receive interrupt is generated when this new data is ready in the `UxDBUF` USART receive- and transmit-data register.

A transmit interrupt is generated when the unit is ready to accept another data byte for transmission. Because `UxDBUF` is double-buffered, this happens just after the transmission has been initiated. Note that data should not be written to `UxDBUF` until `UxCSR.TX_BYTE` is 1. For DMA transfers, this is handled automatically. For back-to-back transmits using DMA, the `UxGCR.CPHA` bit must be set to zero; if not, transmitted bytes can become corrupted. For systems requiring setting of `UxGCR.CPHA`, polling `UxCSR.TX_BYTE` is needed.

Also, note the difference between transmit interrupt and receive interrupt, as the former arrives approximately eight bit-periods prior to the latter.

SPI master-mode operation as described previously is a three-wire interface. No select input is used to enable the master. If the external slave requires a slave-select signal, this can be implemented through software using a general-purpose I/O pin.

### 17.2.2 SPI Slave Operation

A SPI byte transfer in slave mode is controlled by the external system. The data on the MOSI input is shifted into the receive register controlled by the serial clock, SCK, which is an input in slave mode. At the same time, the byte in the transmit register is shifted out onto the MISO output.

The `UxCSR.ACTIVE` bit goes high when the transfer starts and low when the transfer ends. Then the `UxCSR.RX_BYTE` bit is set and a receive interrupt is generated.

The expected polarity and clock phase of SCK is selected by `UxGCR.CPOL` and `UxGCR.CPHA`. The expected order of the byte transfer is selected by the `UxGCR.ORDER` bit.

At the end of the transfer, the received data byte is available for reading from `UxDBUF`.

The transmit interrupt is generated at the start of the operation.

### 17.3 SSN Slave-Select Pin

When the USART is operating in SPI mode, configured as a SPI slave, a four-wire interface is used with the slave-select (SSN) pin as an input to the SPI. When SSN is low, the SPI slave is active, receives data on the MOSI input, and outputs data on the MISO output. When SSN is high, the SPI slave is inactive and does not receive data. The MISO output is in the high-impedance state when SSN is high. Also note that the release of SSN (SSN going high) must be aligned to the end of the byte received or sent. If released during a byte, the next received byte is not received properly, as information about the previous byte is present in the SPI system. A USART flush can be used to remove this information.

In SPI master mode, the SSN pin is not used. When the USART operates as a SPI master and a slave-select signal is required by an external SPI slave device, then a general-purpose I/O pin should be used to implement the slave-select function in software.

### 17.4 Baud-Rate Generation

An internal baud-rate generator sets the UART baud rate when operating in UART mode and the SPI master clock frequency when operating in SPI mode.

The `UxBAUD.BAUD_M[7:0]` and `UxGCR.BAUD_E[4:0]` registers define the baud rate used for UART transfers and the rate of the serial clock for SPI transfers. The baud rate is given by the following equation:

$$\text{Baud Rate} = \frac{(256 + \text{BAUD\_M}) \times 2^{\text{BAUD\_E}}}{2^{28}} \times f \quad (3)$$

where  $f$  is the system clock frequency, 16 MHz for the RCOSC or 32 MHz for the XOSC.

The register values required for standard baud rates are shown in [Table 17-1](#) for a typical system clock set to 32 MHz. The table also gives the difference in actual baud rate to standard baud rate value as a percentage error.

The maximum baud rate for the UART mode is  $f / 16$  when `BAUD_E` is 16 and `BAUD_M` is 0, and where  $f$  is the system clock frequency.

See the device data sheet for the maximum baud rate in SPI mode.

Note that the baud rate must be set through the `UxBAUD` and `UxGCR` registers before any other UART or SPI operations take place. If the baud rate is changed while in UART mode, it may take up to one bit period of the old baud rate before the change takes effect.

**Table 17-1. Commonly Used Baud-Rate Settings for 32 MHz System Clock**

Baud Rate (bps)	UxBAUD.BAUD_M	UxGCR.BAUD_E	Error (%)
2400	59	6	0.14
4800	59	7	0.14
9600	59	8	0.14
14,400	216	8	0.03
19,200	59	9	0.14

**Table 17-1. Commonly Used Baud-Rate Settings for 32 MHz System Clock (continued)**

Baud Rate (bps)	UxBAUD.BAUD_M	UxGCR.BAUD_E	Error (%)
28,800	216	9	0.03
38,400	59	10	0.14
57,600	216	10	0.03
76,800	59	11	0.14
115,200	216	11	0.03
230,400	216	12	0.03

## 17.5 USART Flushing

The current operation can be aborted by setting the `UxUCR.FLUSH` register bit. This event stops the current operation and clears all data buffers. Note that when setting the flush bit during the TX or RX of a bit, the flushing does not take place until TX or RX of this bit has ended (buffers are cleared immediately, but timers keeping knowledge of bit duration are not). Thus, using the flush bit should either be aligned with USART interrupts or use a wait time of one bit duration at the current baud rate before updated data or configuration can be received by the USART.

## 17.6 USART Interrupts

Each USART has two interrupts. These are the RX complete interrupt (URXx) and the TX interrupt (UTXx). The TX interrupt is triggered when transmission starts and the data buffer is offloaded.

The USART interrupt enable bits are found in the `IEN0` and `IEN2` registers. The interrupt flags are located in the `TCON` and `IRCON2` registers. See [Section 2.5](#) for details of these registers. The interrupt enables and flags are summarized as follows.

Interrupt enables:

- USART0 RX: `IEN0.URX0IE`
- USART1 RX: `IEN0.URX1IE`
- USART0 TX: `IEN2.UTX0IE`
- USART1 TX: `IEN2.UTX1IE`

Interrupt flags:

- USART0 RX: `TCON.URX0IF`
- USART1 RX: `TCON.URX1IF`
- USART0 TX: `IRCON2.UTX0IF`
- USART1 TX: `IRCON2.UTX1IF`

## 17.7 USART DMA Triggers

There are two DMA triggers associated with each USART. The DMA triggers are activated by RX complete and TX complete events, that is, the same events as the USART interrupt requests. A DMA channel can be configured using a USART receive-and-transmit buffer, `UxDBUF`, as source or destination address.

See [Table 8-1](#) for an overview of the DMA triggers.

## 17.8 USART Registers

The registers for the USART are described in this section. For each USART, there are five registers consisting of the following (x refers to the USART number, that is, 0 or 1):

- `UxCSR`, USART x control and status
- `UxUCR`, USART x UART control

- U<sub>x</sub>GCR, USART x generic control
- U<sub>x</sub>DBUF, USART x receive- and transmit-data buffer
- U<sub>x</sub>BAUD, USART x baud-rate control

**U0CSR (0x86) – USART 0 Control and Status**

Bit	Name	Reset	R/W	Description
7	MODE	0	R/W	USART mode select 0: SPI mode 1: UART mode
6	RE	0	R/W	UART receiver enable. Note: Do not enable receive before UART is fully configured. 0: Receiver disabled 1: Receiver enabled
5	SLAVE	0	R/W	SPI master or slave mode select 0: SPI master 1: SPI slave
4	FE	0	R/W0	UART framing error status. This bit is automatically cleared on a read of the U0CSR register or bits in the U0CSR register. 0: No framing error detected 1: Byte received with incorrect stop-bit level
3	ERR	0	R/W0	UART parity error status. This bit is automatically cleared on a read of the U0CSR register or bits in the U0CSR register. 0: No parity error detected 1: Byte received with parity error
2	RX_BYTE	0	R/W0	Receive byte status. UART mode and SPI slave mode. This bit is automatically cleared when reading U0DBUF; clearing this bit by writing 0 to it effectively discards the data in U0DBUF. 0: No byte received 1: Received byte ready
1	TX_BYTE	0	R/W0	Transmit byte status. UART mode and SPI master mode 0: Byte not transmitted 1: Last byte written to data-buffer register has been transmitted
0	ACTIVE	0	R	USART transmit or receive active status. In SPI slave mode, this bit equals slave select. 0: USART idle 1: USART busy in transmit or receive mode



**U0UCR (0xC4) – USART 0 UART Control**

Bit	Name	Reset	R/W	Description
7	FLUSH	0	R0/W1	Flush unit. When set, this event stops the current operation and returns the unit to the idle state.
6	FLOW	0	R/W	UART hardware flow enable. Selects use of hardware flow control with RTS and CTS pins 0: Flow control disabled 1: Flow control enabled
5	D9	0	R/W	If parity is enabled (see <b>PARITY</b> , bit 3 in this register), then this bit sets the parity level as follows: 0: Odd parity 1: Even parity
4	BIT9	0	R/W	Set this bit to 1 in order to enable the parity bit transfer (as 9th bit). The content of this 9th bit is given by <b>D9</b> , if parity is enabled by <b>PARITY</b> . 0: 8-bit transfer 1: 9-bit transfer
3	PARITY	0	R/W	UART parity enable. One must set <b>BIT9</b> in addition to setting this bit for parity to be calculated. 0: Parity disabled 1: Parity enabled
2	SPB	0	R/W	UART number of stop bits. Selects the number of stop bits to transmit 0: 1 stop bit 1: 2 stop bits
1	STOP	1	R/W	UART stop-bit level must be different from the start-bit level 0: Low stop bit 1: High stop bit
0	START	0	R/W	UART start-bit level. Ensure that the polarity of the start bit is opposite the level of the idle line. 0: Low start bit 1: High start bit

**U0GCR (0xC5) – USART 0 Generic Control**

Bit	Name	Reset	R/W	Description
7	CPOL	0	R/W	SPI clock polarity 0: Negative clock polarity 1: Positive clock polarity
6	CPHA	0	R/W	SPI clock phase 0: Data is output on <i>MOSI</i> when <i>SCK</i> goes from <b>CPOL</b> inverted to <b>CPOL</b> , and data input is sampled on <i>MISO</i> when <i>SCK</i> goes from <b>CPOL</b> to <b>CPOL</b> inverted. 1: Data is output on <i>MOSI</i> when <i>SCK</i> goes from <b>CPOL</b> to <b>CPOL</b> inverted, and data input is sampled on <i>MISO</i> when <i>SCK</i> goes from <b>CPOL</b> inverted to <b>CPOL</b> .
5	ORDER	0	R/W	Bit order for transfers 0: LSB first 1: MSB first
4:0	BAUD_E[4:0]	0 0000	R/W	Baud rate exponent value. <b>BAUD_E</b> along with <b>BAUD_M</b> determines the UART baud rate and the SPI master <i>SCK</i> clock frequency.

**U0DBUF (0xC1) – USART 0 Receive- and Transmit-Data Buffer**

Bit	Name	Reset	R/W	Description
7:0	DATA[7:0]	0x00	R/W	USART receive and transmit data. When writing this register, the data written is written to the internal transmit-data register. When reading this register, the data from the internal read-data register is read.

**U0BAUD (0xC2) – USART 0 Baud-Rate Control**

Bit	Name	Reset	R/W	Description
7:0	BAUD_M[7:0]	0x00	R/W	Baud-rate mantissa value. <b>BAUD_E</b> along with <b>BAUD_M</b> decides the UART baud rate and the SPI master <i>SCK</i> clock frequency.

**U1CSR (0xF8) – USART 1 Control and Status**

Bit	Name	Reset	R/W	Description
7	MODE	0	R/W	USART mode select 0: SPI mode 1: UART mode
6	RE	0	R/W	UART receiver enable. Note: Do not enable receive before UART is fully configured. 0: Receiver disabled 1: Receiver enabled
5	SLAVE	0	R/W	SPI master- or slave-mode select 0: SPI master 1: SPI slave
4	FE	0	R/W0	UART framing error status. This bit is automatically cleared on a read of the U1CSR register or bits in the U1CSR register. 0: No framing error detected 1: Byte received with incorrect stop-bit level
3	ERR	0	R/W0	UART parity error status. This bit is automatically cleared on a read of the U1CSR register or bits in the U1CSR register. 0: No parity error detected 1: Byte received with parity error
2	RX_BYTE	0	R/W0	Receive byte status. UART mode and SPI slave mode. This bit is automatically cleared when reading U1DBUF; clearing this bit by writing 0 to it effectively discards the data in U1DBUF. 0: No byte received 1: Received byte ready
1	TX_BYTE	0	R/W0	Transmit byte status. UART mode and SPI master mode 0: Byte not transmitted 1: Last byte written to data buffer register has been transmitted
0	ACTIVE	0	R	USART transmit or receive active status. In SPI slave mode, this bit equals slave select. 0: USART idle 1: USART busy in transmit or receive mode

**U1UCR (0xFB) – USART 1 UART Control**

Bit	Name	Reset	R/W	Description
7	FLUSH	0	R0/W1	Flush unit. When set, this event stops the current operation and returns the unit to the idle state.
6	FLOW	0	R/W	UART hardware flow enable. Selects use of hardware flow control with RTS and CTS pins 0: Flow control disabled 1: Flow control enabled
5	D9	0	R/W	If parity is enabled (see PARITY, bit 3 in this register), then this bit sets the parity level as follows. 0: Odd parity 1: Even parity
4	BIT9	0	R/W	Set this bit to 1 in order to enable the parity bit transfer (as 9th bit). The content of this 9th bit is given by D9, if parity is enabled by PARITY. 0: 8-bit transfer 1: 9-bit transfer
3	PARITY	0	R/W	UART parity enable. One must set BIT9 in addition to setting this bit for parity to be calculated. 0: Parity disabled 1: Parity enabled
2	SPB	0	R/W	UART number of stop bits. Selects the number of stop bits to transmit 0: 1 stop bit 1: 2 stop bits
1	STOP	1	R/W	UART stop-bit level must be different from start-bit level. 0: Low stop bit 1: High stop bit
0	START	0	R/W	UART start-bit level. Ensure that the polarity of the start bit is opposite the level of the idle line. 0: Low start bit 1: High start bit

**U1GCR (0xFC) – USART 1 Generic Control**

Bit	Name	Reset	R/W	Description
7	CPOL	0	R/W	SPI clock polarity 0: Negative clock polarity 1: Positive clock polarity
6	CPHA	0	R/W	SPI clock phase 0: Data is output on <i>MOSI</i> when <i>SCK</i> goes from <b>CPOL</b> inverted to <b>CPOL</b> , and data input is sampled on <i>MISO</i> when <i>SCK</i> goes from <b>CPOL</b> to <b>CPOL</b> inverted. 1: Data is output on <i>MOSI</i> when <i>SCK</i> goes from <b>CPOL</b> to <b>CPOL</b> inverted, and data input is sampled on <i>MISO</i> when <i>SCK</i> goes from <b>CPOL</b> inverted to <b>CPOL</b> .
5	ORDER	0	R/W	Bit order for transfers 0: LSB first 1: MSB first
4:0	BAUD_E[4:0]	0 0000	R/W	Baud rate exponent value. BAUD_E along with BAUD_M determines the UART baud rate and the SPI master SCK clock frequency.

**U1DBUF (0xF9) – USART 1 Receive- and Transmit-Data Buffer**

Bit	Name	Reset	R/W	Description
7:0	DATA[7:0]	0x00	R/W	USART receive and transmit data. When writing this register, the data written is written to the internal transmit-data register. When reading this register, the data from the internal read-data register is read.

**U1BAUD (0xFA) – USART 1 Baud-Rate Control**

Bit	Name	Reset	R/W	Description
7:0	BAUD_M[7:0]	0x00	R/W	Baud rate mantissa value. BAUD_E along with BAUD_M determines the UART baud rate and the SPI master SCK clock frequency.

## Operational Amplifier

---

---

The operational amplifier (in the CC2530, CC2531, and CC2540) has the following features:

- Low offset
- Ideal for use in combination with the onboard ADC in sensor applications

Topic	Page
<b>18.1 Description</b> .....	<b>165</b>
<b>18.2 Calibration</b> .....	<b>165</b>
<b>18.3 Clock Source</b> .....	<b>165</b>
<b>18.4 Registers</b> .....	<b>165</b>

## 18.1 Description

The operational amplifier is connected to the I/O pins as follows:

- The positive input pin is connected to P0\_0.
- The negative input pin is connected to P0\_1.
- The output is connected to P0\_2.

The pins used by the operational amplifier must be configured as analog pins, by setting bits `APCFG[2:0]` to 1. The `OPAMPC.EN` bit is used to enable or disable the operational amplifier. When power mode PM2 or PM3 is entered, the operational amplifier is shut down automatically and must be restarted when entering PM0 again.

## 18.2 Calibration

The operational amplifier must be calibrated. A calibration is started by writing 1 to `OPAMPC.CAL`. During calibration, `OPAMPS.CAL_BUSY` is 1. A new calibration is not accepted before `OPAMPS.CAL_BUSY` goes low. Every time after enabling the operational amplifier, calibration must be performed.

## 18.3 Clock Source

The operational amplifier uses a divided version of the system clock. The division factor depends on which clock source is used, 16-MHz RCOSC or XOSC. While the operational amplifier is enabled, the clock source should not be changed.

## 18.4 Registers

This section describes the registers for the operational amplifier.

### OPAMPMC (CC2530, CC2531: 0x61A6. CC2540: 0x61AD) – Operational Amplifier Mode Control

Bit	Name	Reset	R/W	Description
7:2	–	0000 00	R/W	Reserved. Always write 0000 00.
1:0	MODE	00	R/W	Operational amplifier mode 00 and 01: Non-chop mode – Higher offset (approximately 500 $\mu$ V), but no chopper ripple. Use in conjunction with Mode 10 if offset cancellation is required. Offset for these two modes is the opposite of the offset seen in Mode 10. 10: Non-chop mode – Higher offset (approximately 500 $\mu$ V), but no chopper ripple. Use in conjunction with Mode 00 or Mode 01 to double sample and correct for the offset by averaging the two samples. 11: Chop mode – Very low offset (approximately 50 $\mu$ V), and very low noise (1 / f noise shifted to 1 MHz due to chopping), and 1 MHz ripple

### OPAMPC (0x62C0) – Operational Amplifier Control

Bit	Name	Reset	R/W	Description
7:2	–	0000 00	R0	Reserved
1	CAL	0	W1/R0	Start calibration. Calibration only starts if <code>OPAMPC.EN</code> is 1.
0	EN	0	R/W	Operational amplifier enable

### OPAMPS (0x62C1) – Operational Amplifier Status

Bit	Name	Reset	R/W	Description
7:1	–	0000 000	R0	Reserved
0	CAL_BUSY	0	R	Calibration in progress

## Analog Comparator

---

---

The analog comparator (in the CC2530, CC2531, CC240 and CC2541) has the following features:

- Low-power operation
- Wake-up source

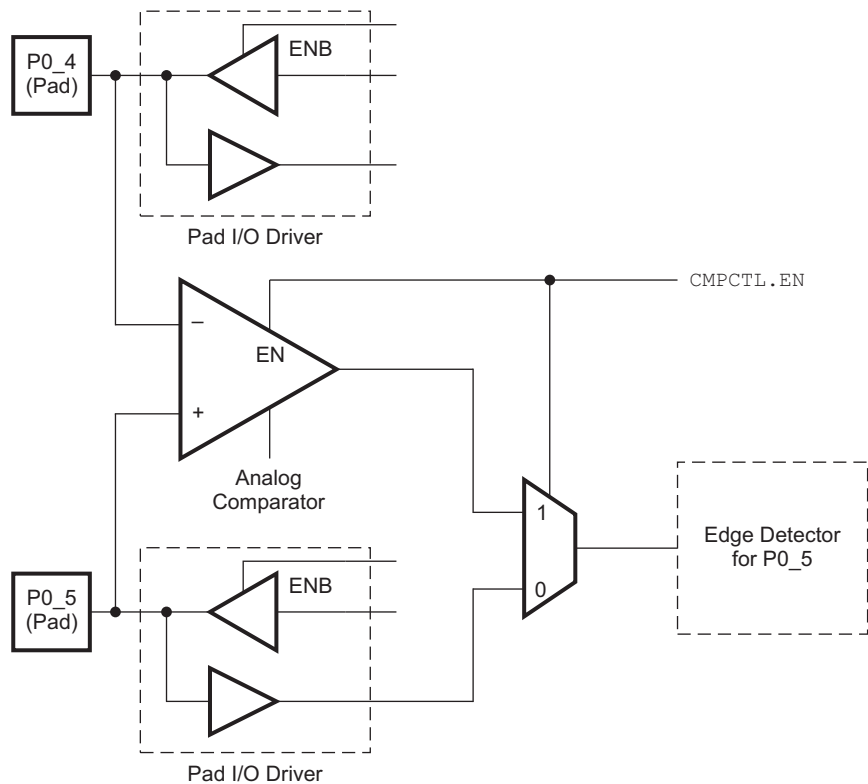
Topic	Page
<b>19.1 Description</b> .....	<b>167</b>
<b>19.2 Register</b> .....	<b>167</b>

## 19.1 Description

The analog comparator is connected to the I/O pins as follows:

- The positive input pin is connected to P0\_5.
- The negative input pin is connected to P0\_4.
- The output can be read from `CMPCTL.OUTPUT`.

The comparator pins must be configured as analog pins by setting bits `APCFG[5:4]` to 1. The `CMPCTL.EN` bit is used to enable or disable the comparator. The output from the comparator is connected internally to the edge detector that controls `P0IFG[5]`. This makes it possible to associate an I/O interrupt with a rising or falling edge on the comparator output. When enabled, the comparator remains active while in power mode 2 or 3. Thus, it is possible to wake up from power mode 2 or 3 on a rising or falling edge on the comparator output.



S0385-01

Figure 19-1. Analog Comparator

## 19.2 Register

This section describes the registers for the analog comparator.

### CMPCTL (0x62D0) – Analog Comparator Control and Status

Bit	Name	Reset	R/W	Description
7:2	–	0000 00	R0	Reserved
1	EN	0	R/W	Comparator enable
0	OUTPUT	0	R	Comparator output

The I<sup>2</sup>C module (in the CC2533 and CC2541) provides an interface between the device and I<sup>2</sup>C-compatible devices connected by the two-wire I<sup>2</sup>C serial bus. External components attached to the I<sup>2</sup>C bus serially transmit and/or receive serial data to or from the I<sup>2</sup>C module through the two-wire I<sup>2</sup>C interface.

The I<sup>2</sup>C module features include:

- Compliance with the I<sup>2</sup>C specification v2.1 (published by NXP)
- 7-bit device addressing modes
- General call
- START, RESTART, and STOP
- Multi-master transmitter and receiver modes
- Slave receiver and transmitter modes
- Standard mode up to 100-kbps and fast mode up to 400-kbps support

Figure 20-1 shows the block diagram of the I<sup>2</sup>C module.

On the CC2533 and CC2541, the I<sup>2</sup>C module is connected to pins 2 and 3 on the chip and uses the P2 interrupt to the CPU. Pins 2 and 3 can alternatively be controlled as two GPIO pins if they are not used by the I<sup>2</sup>C module.

The I<sup>2</sup>C pins cannot be used to wake the device from PM2 or PM3. To wake up on activity on the I<sup>2</sup>C, the I<sup>2</sup>C bus must be connected to a normal GPIO in parallel.

Topic	Page
<b>20.1 Operation</b> .....	<b>169</b>
<b>20.2 I<sup>2</sup>C Registers</b> .....	<b>178</b>



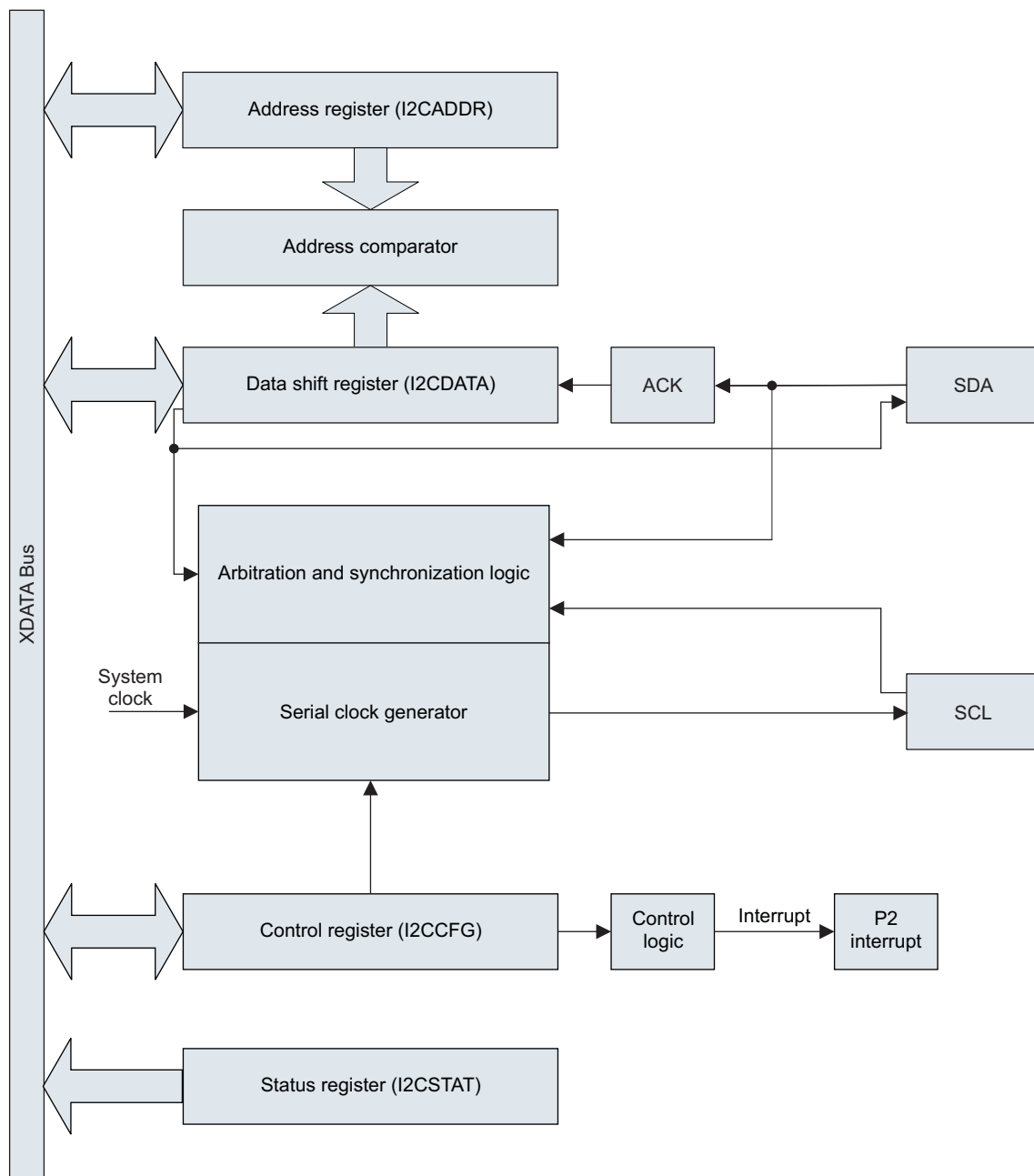


Figure 20-1. Block Diagram of the I<sup>2</sup>C Module

## 20.1 Operation

The I<sup>2</sup>C module supports any slave or master I<sup>2</sup>C-compatible device. Figure 20-2 shows an example of an I<sup>2</sup>C bus. Each I<sup>2</sup>C device is recognized by a unique address and can operate as either a transmitter or a receiver. A device connected to the I<sup>2</sup>C bus can be considered as the master or the slave when performing data transfers. A master initiates a data transfer and generates the clock signal, SCL. Any device addressed by a master is considered a slave.

I<sup>2</sup>C data is communicated using the serial data (SDA) pin and the serial clock (SCL) pin. Both SDA and SCL are bidirectional and must be connected to a positive supply voltage using a pullup resistor.

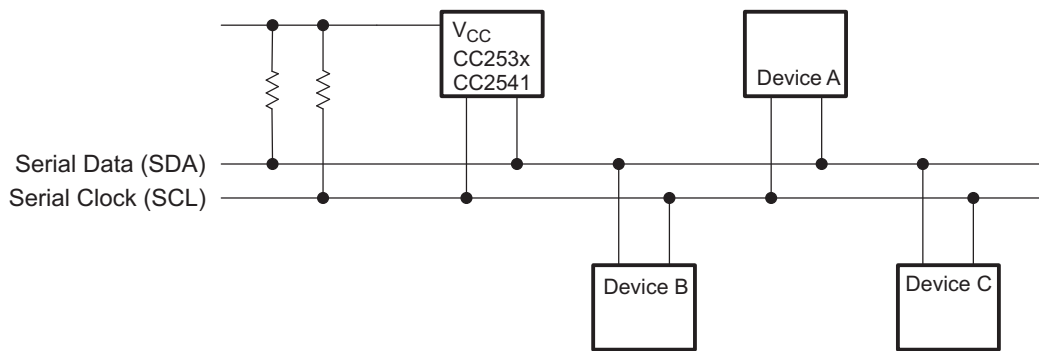


Figure 20-2. I<sup>2</sup>C Bus Connection Diagram

### 20.1.1 I<sup>2</sup>C Initialization and Reset

The I<sup>2</sup>C module is enabled by setting the I2CCFG.ENS1 bit. It is then in the not-addressed slave state.

The I<sup>2</sup>C configuration and state is not retained in power modes PM2 and PM3. It must be reconfigured after coming out of sleep mode.

The I<sup>2</sup>C module is not reset when disabled, and retains its internal state until the next time I2CCFG.ENS1 is set.

### 20.1.2 I<sup>2</sup>C Serial Data

One clock pulse is generated by the master device for each data bit transferred. The I<sup>2</sup>C module operates with byte data. Data is transferred MSB first as shown in Figure 20-3.

The first byte after a START condition consists of a 7-bit slave address and the R/W bit. When  $R/\overline{W} = 0$ , the master transmits data to a slave. When  $R/\overline{W} = 1$ , the master receives data from a slave. The ACK bit is sent from the receiver after each byte on the ninth SCL clock.

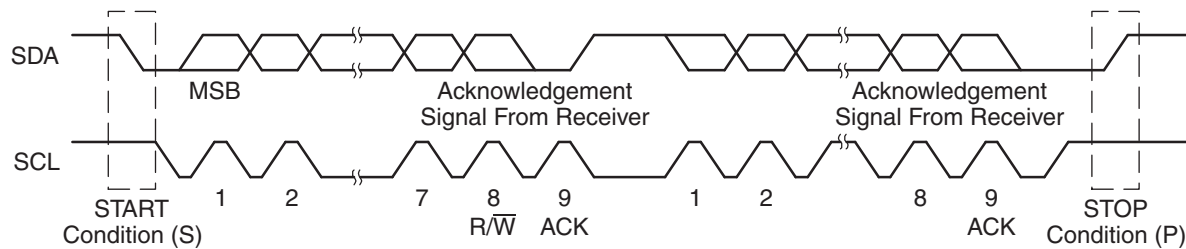


Figure 20-3. I<sup>2</sup>C Module Data Transfer

START and STOP conditions are generated by the master and are shown in Figure 20-3. A START condition is a high-to-low transition on the SDA line while SCL is high. A STOP condition is a low-to-high transition on the SDA line while SCL is high.

Data on SDA must be stable during the high period of SCL (see Figure 20-4). The state of SDA can only change when SCL is low, otherwise a START or STOP condition is generated.

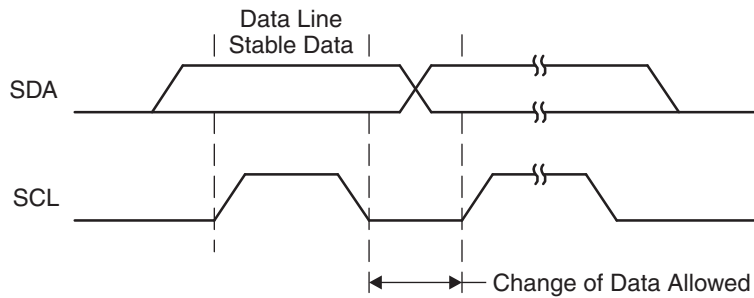


Figure 20-4. Bit Transfer on I<sup>2</sup>C Bus

### 20.1.3 I<sup>2</sup>C Addressing Modes

The I<sup>2</sup>C module supports 7-bit addressing mode.

#### 20.1.3.1 7-Bit Addressing

In the 7-bit addressing format (see Figure 20-5), the first byte is the 7-bit slave address and the R/W bit. The ACK bit is sent from the receiver after each byte.

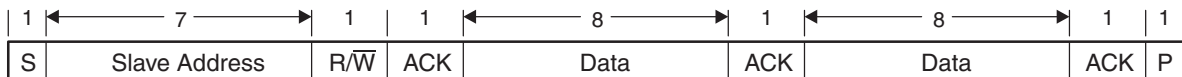


Figure 20-5. I<sup>2</sup>C Module 7-Bit Addressing Format

#### 20.1.3.2 Repeated Start Conditions

The direction of data flow on SDA can be changed by the master, without first stopping a transfer, by issuing a repeated START condition. This is called a RESTART. After a RESTART is issued, the slave address is again sent out with the new data direction specified by the R/W bit. The RESTART condition is shown in Figure 20-6.

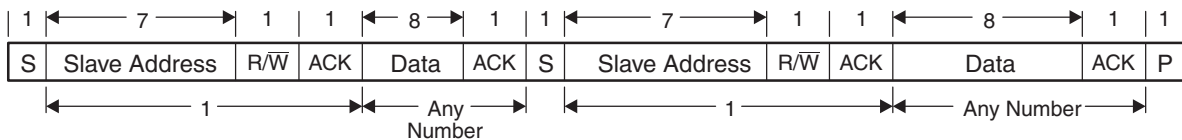


Figure 20-6. I<sup>2</sup>C Module Addressing Format With Repeated START Condition

### 20.1.4 I<sup>2</sup>C Module Operating Modes

The I<sup>2</sup>C module can operate in master transmitter, master receiver, slave transmitter, or slave receiver mode. The modes are discussed in the following sections.

#### 20.1.4.1 Slave Mode

Initially, the I<sup>2</sup>C module is configured in receiver mode by setting the I2CCFG.ENS1 bit to receive the I<sup>2</sup>C address. Afterwards, transmit and receive operations are controlled automatically, depending on the R/W bit received, together with the slave address.

The I<sup>2</sup>C slave address is programmed with the I2CADDR.ADDR bits. The value of the I2CADDR.GC bit determines whether the slave responds to a general call.

When a START condition is detected on the bus, the I<sup>2</sup>C module receives the transmitted address and compares it against its own address stored in I2CADDR.ADDR. If the compare is successful, an interrupt is generated and the I2CCFG.SI bit is set. The same is done for a general call address match if the I2CADDR.GC bit is set.

#### 20.1.4.1.1 I<sup>2</sup>C Slave Transmitter Mode

Slave transmitter mode is entered when the slave address transmitted by the master is identical to the address of this device with a set R/W bit. The slave transmitter shifts the serial data out on SDA with the clock pulses that are generated by the master device. The slave device does not generate the clock, but it does hold SCL low. A CPU intervention is required after a byte has been transmitted by the slave device.

If the master requests data from the slave, the I<sup>2</sup>C module is automatically configured as a transmitter, and I2CCFG.SI is set. The SCL line is held low until the first data to be sent is written into the data buffer, I2CDATA. Then the address is acknowledged and the data is transmitted. After the data is acknowledged by the master, the bus is stalled during the acknowledge cycle by holding SCL low until new data is written into I2CDATA. If the master sends a NACK, the I<sup>2</sup>C module returns to the not-addressed slave state.

Table 20-1 provides more details regarding the slave transmitter operation.

**Table 20-1. Slave Transmitter Mode**

Status Code (Value of I2CSTAT)	Status of the I <sup>2</sup> C	Application Software Response					Next Action Taken by I <sup>2</sup> C Hardware
		To or From I2CDATA	To I2CCFG				
			STA	STO	SI	AA	
0xA8	Own SLA+R has been received; ACK has been returned.	Load data byte	X	0	0	0	Last data byte is transmitted and ACK is received. Data byte is transmitted; ACK is received.
		or load data byte	X	0	0	1	
0xB0	Arbitration lost in SLA+R/W as master; own SLA+R has been received; ACK has been returned.	Load data byte	X	0	0	0	Last data byte is transmitted and ACK is received. Data byte is transmitted; ACK is received.
		or load data byte	X	0	0	1	
0xB8	Data byte has been transmitted; ACK has been received.	Load data byte	X	0	0	0	Last data byte is transmitted and ACK is received Data byte is transmitted; ACK is received.
		or load data byte	X	0	0	1	
0xC0	Data byte has been transmitted; not-ACK has been received.	No action	0	0	0	0	Switched to not-addressed SLV mode; no recognition of own SLA or general call address
		or no action	0	0	0	1	Switched to not-addressed SLV mode; own SLA or general-call address is recognized.
		or no action	1	0	0	0	Switched to not-addressed SLV mode; no recognition of own SLA or general call address; START condition is transmitted when the bus becomes free.
		or no action	1	0	0	1	Switched to not-addressed SLV mode; own SLA or general-call address is recognized; START condition is transmitted when the bus becomes free.

**Table 20-1. Slave Transmitter Mode (continued)**

Status Code (Value of I2CSTAT)	Status of the I <sup>2</sup> C	Application Software Response				Next Action Taken by I <sup>2</sup> C Hardware	
		To or From I2CDATA	To I2CCFG				
			STA	STO	SI		AA
0xC8	Last data byte has been transmitted; ACK has been received.	No action	0	0	0	0	Switched to not-addressed SLV mode; no recognition of own SLA or general-call address
		or no action	0	0	0	1	Switched to not-addressed SLV mode; own SLA or general call address is recognized.
		or no action	1	0	0	0	Switched to not-addressed SLV mode; no recognition of own SLA or general call address; START condition is transmitted when the bus becomes free.
		or no action	1	0	0	1	Switched to not-addressed SLV mode; own SLA or general call address is recognized; START condition is transmitted when the bus becomes free.

### 20.1.4.1.2 I<sup>2</sup>C Slave Receiver Mode

Slave receiver mode is entered when the slave address transmitted by the master is identical to its own address and a cleared R/W bit is received. In slave receiver mode, serial data bits received on SDA are shifted in with the clock pulses that are generated by the master device. The slave device does not generate the clock, but it can hold SCL low if intervention of the CPU is required after a byte has been received.

If the slave interrupt is triggered from the master, the I<sup>2</sup>C module is automatically configured as a receiver and I2CCFG.SI is set. After the first data byte is received, the interrupt flag I2CCFG.SI is set again. The I<sup>2</sup>C module automatically acknowledges the received data.

While the I2CCFG.SI flag is set, the bus is stalled by holding SCL low.

When the master generates a STOP condition, the I2CCFG.STO flag is set.

If the master generates a repeated START condition, the I<sup>2</sup>C state machine returns to its address reception state.

Table 20-2 provides more details regarding slave receiver operation.

**Table 20-2. Slave Receiver Mode**

Status Code (Value of I2CSTAT)	Status of the I <sup>2</sup> C	Application Software Response				Next Action Taken by I <sup>2</sup> C Hardware	
		To or From I2CDATA	To I2CCFG				
			STA	STO	SI		AA
0x60	Own SLA+W has been received; ACK has been returned.	No action	X	0	0	0	Data byte is received and not-ACK is returned
		or no action	X	0	0	1	Data byte is received and ACK is returned.
0x68	Arbitration lost in SLA+R/W as master; own SLA+W has been received, ACK returned.	No action	X	0	0	0	Data byte is received and not-ACK is returned.
		or no action	X	0	0	1	Data byte is received and ACK is returned.
0x70	General-call address (0x00) has been received; ACK has been returned	No action	X	0	0	0	Data byte is received and not-ACK is returned.
		or no action	X	0	0	1	Data byte is received and ACK is returned.

**Table 20-2. Slave Receiver Mode (continued)**

Status Code (Value of I2CSTAT)	Status of the I <sup>2</sup> C	Application Software Response				Next Action Taken by I <sup>2</sup> C Hardware	
		To or From I2CDATA	To I2CCFG				
			STA	STO	SI		AA
0x78	Arbitration lost in SLA+R/W as master; general-call address has been received, ACK returned.	No action	X	0	0	0	Data byte is received and not-ACK is returned.
		or no action	X	0	0	1	Data byte is received and ACK is returned.
0x80	Previously addressed with own SLV address; DATA has been received, ACK returned.	Read data byte	X	0	0	0	Data byte is received and not-ACK is returned.
		or read data byte	X	0	0	1	Data byte is received and ACK is returned.
0x88	Previously addressed with own SLA; DATA byte has been received, not-ACK returned.	Read data byte	0	0	0	0	Switched to not-addressed SLV mode; no recognition of own SLA or general-call address
		or read data byte	0	0	0	1	Switched to not-addressed SLV mode; own SLA or general call address is recognized.
		or read data byte	1	0	0	0	Switched to not-addressed SLV mode; no recognition of own SLA or general-call address; START condition is transmitted when the bus becomes free.
		or read data byte	1	0	0	1	Switched to not-addressed SLV mode; own SLA or general-call address is recognized; START condition is transmitted when the bus becomes free.
0x90	Previously addressed with general-call address; DATA has been received, ACK returned.	Read data byte	X	0	0	0	Data byte is received and not-ACK is returned.
		or read data byte	X	0	0	1	Data byte is received and ACK is returned.
0x98	Previously addressed with own SLA; DATA byte has been received, not-ACK returned.	Read data byte	0	0	0	0	Switched to not-addressed SLV mode; no recognition of own SLA or general-call address
		or read data byte	0	0	0	1	Switched to not-addressed SLV mode; own SLA or general-call address is recognized.
		or read data byte	1	0	0	0	Switched to not-addressed SLV mode; no recognition of own SLA or general-call address; START condition is transmitted when the bus becomes free.
		or read data byte	1	0	0	1	Switched to not-addressed SLV mode; own SLA or general-call address is recognized; START condition is transmitted when the bus becomes free.
0xA0	A STOP condition or repeated START condition has been received while still addressed as SLV-REC or SLV-TRX.	No action	0	0	0	0	Switched to not-addressed SLV mode; no recognition of own SLA or general-call address
		or no action	0	0	0	1	Switched to not-addressed SLV mode; own SLA or general-call address is recognized.
		or no action	1	0	0	0	Switched to not-addressed SLV mode; no recognition of own SLA or general-call address; START condition is transmitted when the bus becomes free.
		or no action	1	0	0	1	Switched to not-addressed SLV mode; own SLA or general-call address is recognized; START condition is transmitted when the bus becomes free.

### 20.1.4.2 Master Mode

The I<sup>2</sup>C module is configured as an I<sup>2</sup>C master by setting the I2CCFG.ENS1 and I2CCFG.STA bits. When the master is part of a multi-master system, its own address must be programmed into the I2CADDR.ADDR register. The value of the I2CADDR.GC bit determines whether the I<sup>2</sup>C module responds to a general call.

#### 20.1.4.2.1 I<sup>2</sup>C Master Transmitter Mode

To enable master transmit mode, set the I2CCFG.ENS1 and I2CCFG.STA bits. The I<sup>2</sup>C module then waits until the I<sup>2</sup>C bus is free. When the I<sup>2</sup>C bus is free, it generates a START condition, sends the slave address, and transfers a transmit direction bit. It then generates an interrupt, and the first byte of data can be written to the I2CDATA register. The I<sup>2</sup>C core sends I2CDATA content if arbitration is not lost, and then generates another interrupt. The I2CSTAT register contains a value of 0x18 or 0x20, depending on the received ACK bit (see Table 20-3). If a not-ACK is received from the slave, the master must react with either a repeated START condition or a STOP condition. Setting I2CCFG.STA during transmission causes a repeated START condition to be transmitted. Setting I2CCFG.STO during transmission causes a STOP condition to be transmitted and the I2CCFG.STO bit to be reset.

Table 20-3 provides more details regarding the master transmitter operation.

**Table 20-3. Master Transmitter Mode**

Status Code (Value of I2CSTAT)	Status of the I <sup>2</sup> C	Application Software Response					Next Action Taken by I <sup>2</sup> C Hardware
		To or From I2CDATA	To I2CCFG				
			STA	STO	SI	AA	
0x08	A START condition has been transmitted.	Load SLA+W	X	0	0	X	SLA+W is transmitted. ACK is received.
0x10	A repeated START condition has been transmitted.	Load SLA+W or load SLA+R	X X	0 0	0 0	X X	Same as for START condition (0x08) SLA+W is transmitted; I <sup>2</sup> C is switched to MST/REC mode.
0x18	SLA+W has been transmitted; ACK has been received.	Load data byte	0	0	0	X	Data byte is transmitted; ACK is received.
		or no action	1	0	0	X	Repeated START is transmitted.
		or no action	0	1	0	X	STOP condition is transmitted; STO flag is reset.
		or no action	1	1	0	X	STOP condition followed by a START condition is transmitted; STO flag is reset.
0x20	SLA+W has been transmitted; not-ACK has been received.	Load data byte	0	0	0	X	Data byte is transmitted; ACK is received.
		or no action	1	0	0	X	Repeated START is transmitted.
		or no action	0	1	0	X	STOP condition is transmitted; STO flag is reset.
		or no action	1	1	0	X	STOP condition followed by a START condition is transmitted; STO flag is reset.
0x28	Data byte is transmitted; ACK is received.	Load data byte	0	0	0	X	Data byte is transmitted; ACK is received.
		or no action	1	0	0	X	Repeated START is transmitted.
		or no action	0	1	0	X	STOP condition is transmitted; STO flag is reset.
		or no action	1	1	0	X	STOP condition followed by a START condition is transmitted; STO flag is reset.

**Table 20-3. Master Transmitter Mode (continued)**

Status Code (Value of I2CSTAT)	Status of the I <sup>2</sup> C	Application Software Response					Next Action Taken by I <sup>2</sup> C Hardware
		To or From I2CDATA	To I2CCFG				
			STA	STO	SI	AA	
0x30	Data byte in I2CDATA has been transmitted.	Data byte	0	0	0	X	Data byte is transmitted; ACK is received.
		or no action	1	0	0	X	Repeated START is transmitted.
		or no action	0	1	0	X	STOP condition is transmitted; STO flag is reset.
		or no action	1	1	0	X	STOP condition followed by a START condition is transmitted; STO flag is reset.
0x38	Arbitration lost in SLA+R/W or data bytes	No action	0	0	0	X	I <sup>2</sup> C bus is released; not-addressed slave is entered.
		or no action	1	0	0	X	A START condition is transmitted when the bus becomes free.

#### 20.1.4.2.2 I<sup>2</sup>C Master Receiver Mode

To enable master receive mode, set the I2CCFG.ENS1 and the I2CCFG.STA bits. The I<sup>2</sup>C module then waits until the I<sup>2</sup>C bus is free. When the I<sup>2</sup>C bus is free, it generates a START condition, sends the slave address, and transfers a receive direction bit. It then generates an interrupt, and the first byte is received.

Table 20-4 provides more details regarding the master receiver operation.

**Table 20-4. Master Receiver Mode**

Status Code (Value of I2CSTAT)	Status of the I <sup>2</sup> C	Application Software Response					Next Action Taken by I <sup>2</sup> C Hardware
		To or From I2CDATA	To I2CCFG				
			STA	STO	SI	AA	
0x08	A START condition has been transmitted.	Load SLA+R	X	0	0	X	SLA+R is transmitted. ACK is received.
0x10	A repeated START condition has been transmitted.	Load SLA+R	X	0	0	X	As above
		or load SLA+W	X	0	0	X	SLA+W is transmitted; I <sup>2</sup> C is switched to MST-TRX mode.
0x38	Arbitration lost in not-ACK bit.	No action	0	0	0	X	I <sup>2</sup> C bus is released; I <sup>2</sup> C enters slave mode.
		or no action	1	0	0	X	A start condition is transmitted when the bus becomes free.
0x40	SLA+R has been transmitted; ACK has been received.	No action	0	0	0	0	Data byte is received; not-ACK is returned.
		or no action	0	0	0	1	Data byte is received; ACK is returned.
0x48	SLA+R has been transmitted; not-ACK has been received.	No action	1	0	0	X	Repeated START condition is transmitted.
		or no action	0	1	0	X	STOP condition is transmitted; STO flag is reset.
		or no action	1	1	0	X	STOP condition followed by a START condition is transmitted; STO flag is reset.
0x50	Data byte has been received; ACK has been returned.	Read data byte	0	0	0	0	Data byte is received; not-ACK is returned.
		or read data byte	0	0	0	1	Data byte is received; ACK is returned
0x58	Data byte has been received; not-ACK has been returned.	Read data byte	1	0	0	X	Repeated START condition is transmitted.
		or read data byte	0	1	0	X	STOP condition is transmitted; STO flag is reset.
		or read data byte	1	1	0	X	STOP condition followed by a START condition is transmitted; STO flag is reset.



### 20.1.4.3 Arbitration

If two or more master transmitters simultaneously start a transmission on the bus, an arbitration procedure is invoked. Figure 20-7 shows the arbitration procedure between two devices. The arbitration procedure uses the data presented on SDA by the competing transmitters. The first master transmitter that generates a logic high is overruled by the opposing master generating a logic low. The arbitration procedure gives priority to the device that transmits the serial data stream with the lowest binary value. The master transmitter that lost arbitration switches to the slave receiver mode. If two or more devices send identical first bytes, arbitration continues on the subsequent bytes.

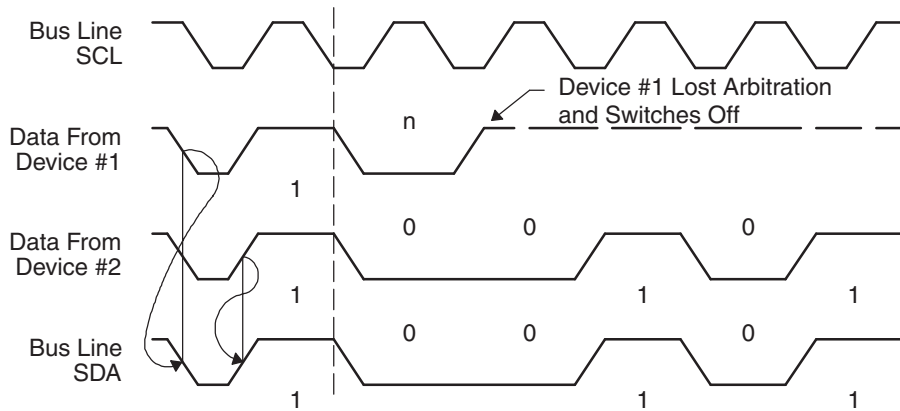


Figure 20-7. Arbitration Procedure Between Two Master Transmitters

### 20.1.5 I<sup>2</sup>C Clock Generation and Synchronization

The I<sup>2</sup>C clock SCL is provided by the master on the I<sup>2</sup>C bus. When the I<sup>2</sup>C module is in master mode, the serial clock generator generates the SCL clock from the system clock. The serial clock generator is switched off when the I<sup>2</sup>C module is in slave mode.

The frequency of the SCL is determined by the system clock frequency and the division factor given by the I2CCFG.CR<sub>x</sub> bits. Example frequencies for a 32-MHz system clock are given in the I2CCFG register description.

During the arbitration procedure, the clocks from the different masters must be synchronized. A device that first generates a low period on SCL overrules the other devices, forcing them to start their own low periods. SCL is then held low by the device with the longest low period. The other devices must wait for SCL to be released before starting their high periods. Figure 20-8 shows the clock synchronization. This allows a slow slave to slow down a fast master.

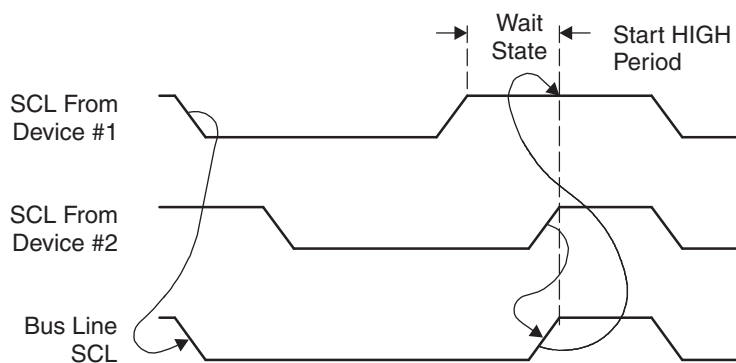


Figure 20-8. Synchronization of Two I<sup>2</sup>C Clock Generators During Arbitration

### 20.1.6 Bus Error

When an incorrect format of a frame is detected, a bus error condition is entered. The cause is that a START or STOP condition was detected during transfer of an address, data, or an acknowledge bit. When a bus error condition is entered, an interrupt is requested. The core leaves the bus error state when the `I2CCFG.STO` flag is set and the interrupt request is cleared. It goes into the slave mode and the `I2CCFG.STO` flag is automatically reset. The SDA and SCL lines are released (the STOP condition is not transmitted).

**Table 20-5. Miscellaneous States**

Status Code (Value of <code>I2CSTAT.STAC</code> )	Status of the I2C	Application Software Response				Next Action Taken by I <sup>2</sup> C Hardware	
		To or From I2CDATA	To I2CCFG				
			STA	STO	SI		AA
0x00	Bus error during MST or selected slave modes	No action	0	1	0	X	Only the internal hardware is affected in the MST or addressed SLV modes. In all cases, the bus is released and I <sup>2</sup> C is switched to the not-addressed SLV mode. The STO flag is reset.

### 20.1.7 I<sup>2</sup>C Interrupt

The I<sup>2</sup>C module has an interrupt line to the CPU to signal that it requires service. The I<sup>2</sup>C module uses interrupt number 6, which is also shared with Port 2 inputs; hence, the interrupt routine must also handle Port 2 interrupts if they are enabled.

For an interrupt request to be generated, `IEN2.P2IE` must be set to 1. When an interrupt request has been generated, the CPU starts executing the ISR if there are no higher-priority interrupts pending.

An interrupt is generated from the I<sup>2</sup>C module when one of the 26 out of 27 possible I<sup>2</sup>C component states is entered. The only state that does not cause an interrupt to be generated is state 0xF8, which indicates that no relevant state information is available. The corresponding `I2CCFG.SI` flag must be cleared by software at the end of the ISR.

### 20.1.8 I<sup>2</sup>C Pins

The SCL and SDA pins of the I<sup>2</sup>C module are connected to pins 2 and 3, respectively, on the CC2533 and CC2541. These pins are pulled up during reset to avoid floating pins. After reset, they are controlled by the I<sup>2</sup>C module and use an internal pullup resistor of 20 kΩ to hold bus signals high. If these pins are not to be used for I<sup>2</sup>C, they can be used as GPIO by setting the `I2CWC.OVR` bit. In this mode, pins 2 and 3 can be set up as outputs, as inputs with optional pullup, or as 4-mA drive-strength outputs like the other GPIO pads on the device by using the configuration bits in `I2CWC`. Their values are read or controlled using the `I2CIO` register. These pins cannot be configured to generate GPIO interrupts.

## 20.2 I<sup>2</sup>C Registers

This section describes all I<sup>2</sup>C registers used for control and status of the I<sup>2</sup>C module.

The registers return to their reset values when the chip enters PM2 or PM3.

**I2CCFG (0x6230) – I<sup>2</sup>C Control**

Bit	Name	Reset	R/W	Description
7	CR2	0	R/W	Clock rate bit 2
6	ENS1	0	R/W	Enable bit 0: I <sup>2</sup> C module disabled SCL and SDA are set to high-impedance inputs. The inputs are ignored by the I <sup>2</sup> C module. Note that setting ENS1 = 0 disables the I <sup>2</sup> C module but does not reset its state. 1: I <sup>2</sup> C module enabled
5	STA	0	R/W	START flag. When set, HW detects when I <sup>2</sup> C is free and generates a START condition.
4	STO	0	R/W1	STOP flag. When set and in master mode, a STOP condition is transmitted on the I <sup>2</sup> C bus. HW is cleared when transmit has completed successfully.
3	SI	0	R/W0	Interrupt flag
2	AA	0	R/W	Assert acknowledge flag for the I <sup>2</sup> C module. When set (AA = 1), an acknowledge is returned when: <ul style="list-style-type: none"> <li>• Slave address is recognized</li> <li>• General call is recognized, when the I<sup>2</sup>C module is enabled</li> <li>• Data byte received while in master or slave receive mode</li> </ul> When not set (AA = 0), an acknowledge is returned when: <ul style="list-style-type: none"> <li>• Data byte is received while in master or slave receive mode</li> </ul>
1	CR1	0	R/W	Clock rate bit 1
0	CR0	0	R/W	Clock rate bit 0

**Table 20-6. Clock Rates Defined at 32 MHz**

CR2	CR1	CR0	Bit Frequency (kHz)	Clock Divided by
0	0	0	123	256
0	0	1	144	244
0	1	0	165	192
0	1	1	197	160
1	0	0	33	960
1	0	1	267	120
1	1	0	533	60
1	1	1	Reserved	N/A

**I2CSTAT (0x6231) – I<sup>2</sup>C Status**

Bit	Name	Reset	R/W	Description
7:3	STAC	1111 1	R	Status code. Contains the state of the I <sup>2</sup> C core. 27 states are defined: 0 to 25 and 31. Interrupt is only requested when in states 0 to 25. The value 0xF8 indicates that there is no relevant state information available and that I2CCFG.SI = 0.
2:0	–	000	R0	Reserved

**I2CDATA (0x6232) – I<sup>2</sup>C Data**

Bit	Name	Reset	R/W	Description
7:0	SD	0000 00 00	R/W	Serial data in/out (MSB is bit 7, LSB is bit 0). Contains data byte to be transmitted or byte which has just been received. Can be read or written while not in the process of shifting a byte. The register is not shadowed or double buffered, so it should only be accessed on an interrupt.

**I2CADDR (0x6233) – I<sup>2</sup>C Own Slave Address**

Bit	Name	Reset	R/W	Description
7:1	ADDR	0000 00 0	R/W	Own slave address
0	GC	0	R/W	General-call address acknowledge. If set, the general-call address is recognized.

**I2CWC (0x6234) – Wrapper Control**

Bit	Name	Reset	R/W	Description
7	OVR	0	R/W	Override enable: 0: I <sup>2</sup> C functionality (ignore other bits in this register) 1: GPIO functionality
6:4	–	000	R0	Reserved
3	SCLPUE	1	R/W	SCL pin pullup enable
2	SDAPUE	1	R/W	SDA pin pullup enable
1	SCLOE	0	R/W	SCL pin output enable
0	SDAOE	0	R/W	SDA pin output enable

**I2CIO (0x6235) – GPIO**

Bit	Name	Reset	R/W	Description
7:2	–	000	R0	Reserved
1	SCLD	0	R/W	SCL data value When I2CWC.SCLOE is set, reading SCLD reads the output register, not the pin. When I2CWC.SCLOE is cleared, reading SCLD reads the pin. Writing SCLD writes to the output register.
0	SDAD	0	R/W	SDA data value When I2CWC.SDAOE is set, reading SDAD reads the output register, not the pin. When I2CWC.SDAOE is cleared, reading SDAD reads the pin. Writing SDAD writes to the output register.

## USB Controller

This section focuses on describing the functionality of the USB controller (in the CC2531 and CC2540 only), and it is assumed that the reader has a good understanding of USB and is familiar with the terms and concepts used. See the Universal Serial Bus Specification for details ([8], [Appendix C](#)).

Standard USB nomenclature is used regarding IN and OUT. That is, IN is always into the host (PC) and OUT is out of the host.

Topic	Page
<b>21.1 USB Introduction</b> .....	<b>182</b>
<b>21.2 USB Enable</b> .....	<b>182</b>
<b>21.3 48-MHz USB PLL</b> .....	<b>182</b>
<b>21.4 USB Interrupts</b> .....	<b>183</b>
<b>21.5 Endpoint 0</b> .....	<b>183</b>
<b>21.6 Endpoint-0 Interrupts</b> .....	<b>183</b>
<b>21.7 Endpoints 1–5</b> .....	<b>185</b>
<b>21.8 DMA</b> .....	<b>189</b>
<b>21.9 USB Reset</b> .....	<b>189</b>
<b>21.10 Suspend and Resume</b> .....	<b>189</b>
<b>21.11 Remote Wake-Up</b> .....	<b>189</b>
<b>21.12 USB Registers</b> .....	<b>190</b>

## 21.1 USB Introduction

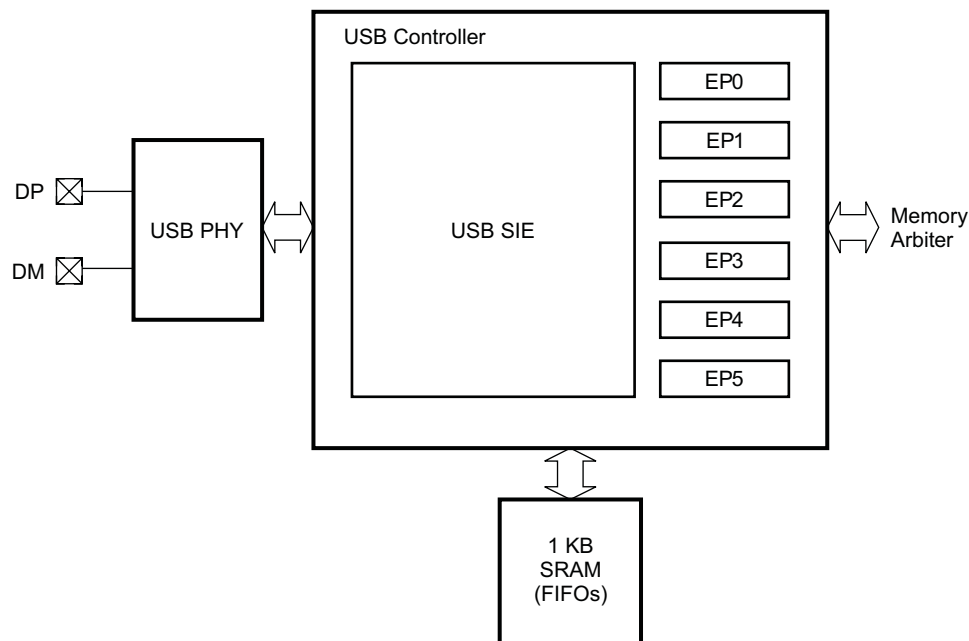
The USB controller monitors the USB for relevant activity and handles packet transfers.

Appropriate response to USB interrupts and loading or unloading of packets into or from endpoint FIFOs is the responsibility of the firmware. The firmware must be able to reply correctly to all standard requests from the USB host and work according to the protocol implemented in the driver on the PC.

The USB controller has the following features:

- Full-speed operation (up to 12 Mbps)
- Five endpoints (in addition to endpoint 0) that can be used as IN, OUT, or IN and OUT, and can be configured as bulk and interrupt or isochronous.
- 1 KB SRAM FIFO available for storing USB packets
- Endpoints supporting packet sizes from 8–512 bytes
- Support for double buffering of USB packets

Figure 21-1 shows a block diagram of the USB controller. The USB PHY is the physical interface with input and output drivers. The USB SIE is the serial-interface engine, which controls the packet transfer to and from the endpoints. The USB controller is connected to the rest of the system through the memory arbiter.



B0305-01

**Figure 21-1. USB Controller Block Diagram**

## 21.2 USB Enable

The USB is enabled by setting `USBCTRL . USB_EN` to 1. Setting `USBCTRL . USB_EN` to 0 resets the USB controller.

## 21.3 48-MHz USB PLL

The 48-MHz internal USB PLL must be powered up and stable for the USB controller to operate correctly. It is important that the crystal oscillator is selected as source and is stable before the USB PLL is enabled. The USB PLL is enabled by setting the `USBCTRL . PLL_EN` bit and waiting for the `USBCTRL . PLL_LOCKED` status flag to go high. When the PLL has locked, it is safe to use the USB controller.

Note: The PLL must be disabled before exiting active mode and re-enabled after entering active mode.

## 21.4 USB Interrupts

There are three interrupt flag registers with associated interrupt-enable mask registers.

**Table 21-1. USB Interrupt Flags Interrupt-Enable Mask Registers**

Interrupt Flag	Description	Associated Interrupt Enable Mask Register
USBCIF	Contains flags for common USB interrupts	USBCIE
USBIIF	Contains interrupt flags for endpoint 0 and all the IN endpoints	USBIIE
USBOIF	Contains interrupt flags for all OUT endpoints	USBOIE
Note: All interrupts except SOF and suspend are initially enabled after reset.		

The USB controller uses interrupt number 6 for USB interrupts. This interrupt number is shared with Port 2 inputs; hence, the interrupt routine must also handle Port 2 interrupts if they are enabled. For an interrupt request to be generated, `IEN2.P2IE` must be set to 1, together with the desired interrupt enable bits from the `USBCIE`, `USBIIE`, and `USBOIE` registers. When an interrupt request has been generated, the CPU starts executing the ISR if there are no higher-priority interrupts pending. The interrupt routine should read all the interrupt flag registers and take action depending on the status of the flags. The interrupt flag registers are cleared when they are read, and the status of the individual interrupt flags should therefore be saved in memory (typically in a local variable on the stack) to allow them to be accessed multiple times.

At the end of the ISR, after the interrupt flags have been read, the interrupt flags should be cleared to allow for new USB and P2 interrupts to be detected. The Port 2 interrupt status flags in the `P2IFG` register should be cleared prior to clearing `IRCON2.P2IF`.

When waking up from suspend (typically in PM1), the USB D+ interrupt flag, `P2IFG.DPIF`, is set. The D+ interrupt flag indicates that there has been a falling edge on the D+ USB data pin. This is a resume event.

## 21.5 Endpoint 0

Endpoint 0 (EP0) is a bidirectional control endpoint, and during the enumeration phase all communication is performed across this endpoint. Before the `USBADDR` register has been set to a value other than 0, the USB controller is only able to communicate through endpoint 0. Setting the `USBADDR` register to a value between 1 and 127 brings the USB function out of the default state in the enumeration phase and into the address state. All configured endpoints are then available for the application.

The EP0 FIFO is only used as either IN or OUT, and double buffering is not provided for endpoint 0. The maximum packet size for endpoint 0 is fixed at 32 bytes.

Endpoint 0 is controlled through the `USBCS0` register by setting the `USBINDEX` register to 0. The `USBCNT0` register contains the number of bytes received.

## 21.6 Endpoint-0 Interrupts

The following events may generate an EP0 interrupt request:

- A data packet has been received (`USBCS0.OUTPKT_RDY = 1`)
- A data packet that was loaded into the EP0 FIFO has been sent to the USB host. (`USBCS0.INPKT_RDY` should be set to 1 when a new packet is ready to be transferred. This bit is cleared by hardware when the data packet has been sent.)
- An IN transaction has been completed (the interrupt is generated during the status stage of the transaction).
- A STALL has been sent (`USBCS0.SENT_STALL = 1`)
- A control transfer ends due to a premature end-of-control transfer (`USBCS0.SETUP_END = 1`)

Any of these events causes `USBIIF.EP0IF` to be asserted, regardless of the status of the EP0 interrupt mask bit `USBIIE.EP0IE`. If the EP0 interrupt mask bit is set to 1, the CPU interrupt flag `IRCON2.P2IF` is also asserted. An interrupt request is only generated if `IEN2.P2IE` and `USBIIE.EP0IE` are both set to 1.

### 21.6.1 Error Conditions

When a protocol error occurs, the USB controller sends a STALL handshake. The `USBCS0.SENT_STALL` bit is asserted, and an interrupt request is generated if the endpoint-0 interrupt is properly enabled. A protocol error can be any of the following:

- An OUT token is received after `USBCS0.DATA_END` has been set to complete the OUT data stage (the host tries to send more data than expected).
- An IN token is received after `USBCS0.DATA_END` has been set to complete the IN data stage (the host tries to receive more data than expected).
- The USB host tries to send a packet that exceeds the maximum packet size during the OUT data stage.
- The size of the DATA1 packet received during the status stage is not 0.

The firmware can also terminate the current transaction by setting the `USBCS0.SEND_STALL` bit to 1. The USB controller then sends a STALL handshake in response to the next request from the USB host.

If an EP0 interrupt is caused by the assertion of the `USBCS0.SENT_STALL` bit, this bit should be de-asserted, and firmware should consider the transfer as aborted (and consequently free the memory buffers, and so forth).

If EP0 receives an unexpected token during the data stage, the `USBCS0.SETUP_END` bit is asserted, and an EP0 interrupt is generated (if enabled properly). EP0 then switches to the IDLE state. Firmware should then set the `USBCS0.CLR_SETUP_END` bit to 1 and abort the current transfer. If `USBCS0.OUTPUT_PKT_RDY` is asserted, this indicates that another setup packet has been received that firmware should process.

### 21.6.2 SETUP Transactions (IDLE State)

The control transfer consists of two or three stages of transactions (setup – data – status or setup – status). The first transaction is a setup transaction. A successful setup transaction comprises three sequential packets (a token packet, a data packet, and a handshake packet), where the data field (payload) of the data packet is exactly 8 bytes long and is referred to as the setup packet. In the setup stage of a control transfer, EP0 is in the IDLE state. The USB controller rejects the data packet if the setup packet is not 8 bytes. Also, the USB controller examines the contents of the setup packet to determine whether or not there is a data stage in the control transfer. If there is a data stage, EP0 switches state to TX (IN transaction) or RX (OUT transaction) when the `USBCS0.CLR_OUTPUT_PKT_RDY` bit is set to 1 (if `USBCS0.DATA_END = 0`).

When a packet is received, the `USBCS0.OUTPUT_PKT_RDY` bit is asserted and an interrupt request is generated (EP0 interrupt) if the interrupt has been enabled. Firmware should perform the following when a setup packet has been received:

1. Unload the setup packet from the EP0 FIFO
2. Examine the contents and perform the appropriate operations
3. Set the `USBCS0.CLR_OUTPUT_PKT_RDY` bit to 1. This denotes the end of the setup stage. If the control transfer has no data stage, the `USBCS0.DATA_END` bit must also be set. If there is no data stage, the USB controller stays in the IDLE state.

### 21.6.3 IN Transactions (TX State)

If the control transfer requires data to be sent to the host, the setup stage is followed by one or more IN transactions in the data stage. In this case, the USB controller is in the TX state and only accepts IN tokens. A successful IN transaction comprises two or three sequential packets (a token packet, a data packet, and a handshake packet<sup>(1)</sup>). If more than 32 bytes (maximum packet size) is to be sent, the data must be split into a number of 32-byte packets followed by a residual packet. If the number of bytes to send is a multiple of 32, the residual packet is a zero-length data packet, because a packet size less than 32 bytes denotes the end of the transfer.

<sup>(1)</sup> For isochronous transfers there would not be a handshake packet from the host.



Firmware should load the EP0 FIFO with the first data packet and set the `USBCS0.INPKT_RDY` bit as soon as possible after the `USBCS0.CLR_OUTPKT_RDY` bit has been set. The `USBCS0.INPKT_RDY` is cleared and an EP0 interrupt is generated when the data packet has been sent. Firmware might then load more data packets as necessary. An EP0 interrupt is generated for each packet sent. Firmware must set `USBCS0.DATA_END` in addition to `USBCS0.INPKT_RDY` when the last data packet has been loaded. This starts the status stage of the control transfer.

EP0 switches to the IDLE state when the status stage has completed. The status stage may fail if the `USBCS0.SEND_STALL` bit is set to 1. The `USBCS0.SENT_STALL` bit is then asserted, and an EP0 interrupt is generated.

If `USBCS0.INPKT_RDY` is not set when receiving an IN token, the USB controller replies with a NAK to indicate that the endpoint is working, but temporarily has no data to send.

#### 21.6.4 OUT Transactions (RX State)

If the control transfer requires data to be received from the host, the setup stage is followed by one or more OUT transactions in the data stage. In this case, the USB controller is in the RX state and only accepts OUT tokens. A successful OUT transaction comprises two or three sequential packets (a token packet, a data packet, and a handshake packet<sup>(2)</sup>). If more than 32 bytes (maximum packet size) is to be received, the data must be split into a number of 32-byte packets followed by a residual packet. If the number of bytes to receive is a multiple of 32, the residual packet is a zero-length data packet, because a data packet with payload less than 32 bytes denotes the end of the transfer.

The `USBCS0.OUTPKT_RDY` bit is set and an EP0 interrupt is generated when a data packet has been received. The firmware should set `USBCS0.CLR_OUTPKT_RDY` when the data packet has been unloaded from the EP0 FIFO. When the last data packet has been received (packet size less than 32 bytes) firmware should also set the `USBCS0.DATA_END` bit. This starts the status stage of the control transfer. The size of the data packet is kept in the `USBCNT0` registers. Note that this value is only valid when `USBCS0.OUTPKT_RDY = 1`.

EP0 switches to the IDLE state when the status stage has completed. The status stage may fail if the DATA1 packet received is not a zero-length data packet or if the `USBCS0.SEND_STALL` bit is set to 1. The `USBCS0.SENT_STALL` bit then is asserted and an EP0 interrupt is generated.

### 21.7 Endpoints 1–5

Each endpoint can be used as an IN only, an OUT only, or IN/OUT. For an IN and OUT endpoint, there are basically two endpoints, an IN endpoint and an OUT endpoint associated with the endpoint number. Configuration and control of IN endpoints is performed through the `USBCSIL` and `USBCSIH` registers. The `USBCSOL` and `USBCSOH` registers are used to configure and control OUT endpoints. Each IN and OUT endpoint can be configured as either an isochronous (`USBCSIH.ISO = 1` and/or `USBCSOH.ISO = 1`) or bulk and interrupt (`USBCSIH.ISO = 0` and/or `USBCSOH.ISO = 0`) endpoint. Bulk and interrupt endpoints are handled identically by the USB controller but have different properties from a firmware perspective.

The `USBINDEX` register must have the value of the endpoint number before the indexed endpoint registers are accessed.

#### 21.7.1 FIFO Management

Each endpoint has a certain number of FIFO memory bytes available for incoming and outgoing data packets. [Table 21-2](#) shows the FIFO size for endpoints 1–5. The firmware is responsible for setting the `USBMAXI` and `USBMAXO` registers correctly for each endpoint to prevent data from being overwritten.

When both the IN and OUT endpoints of an endpoint number do not use double buffering, the sum of `USBMAXI` and `USBMAXO` must not exceed the FIFO size for the endpoint. [Figure 21-2 a\)](#) shows how the IN and OUT FIFO memory for an endpoint is organized with single buffering. The IN FIFO grows down from the top of the endpoint memory region, whereas the OUT FIFO grows up from the bottom of the endpoint memory region.

<sup>(2)</sup> For isochronous transfers, there is no handshake packet from the device.

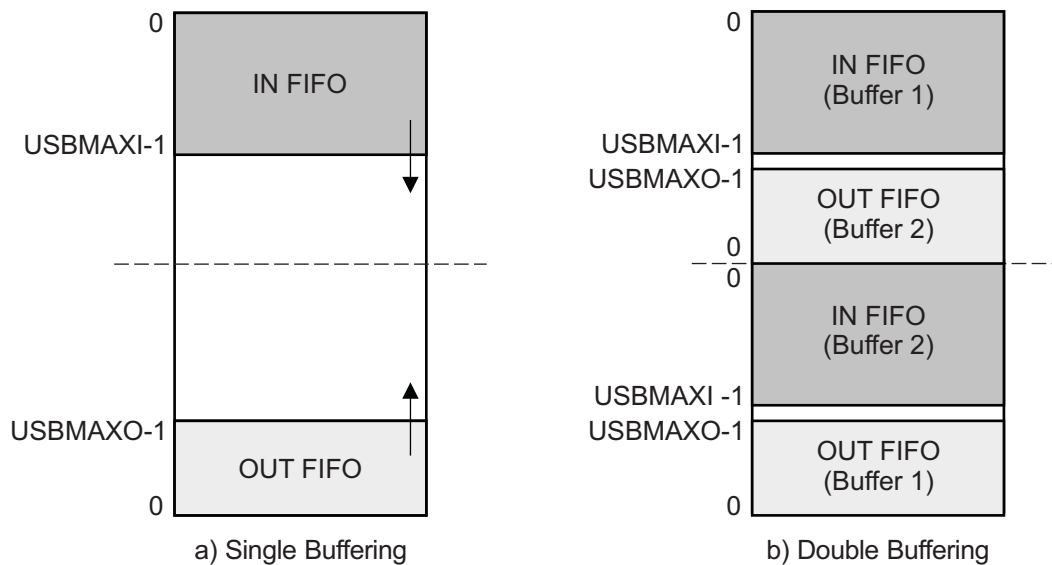
When the IN or OUT endpoint of an endpoint number uses double buffering, the sum of `USBMAXI` and `USBMAXO` must not exceed half the FIFO size for the endpoint. Figure 21-2 b) illustrates the IN and OUT FIFO memory for an endpoint that uses double buffering. Notice that the second OUT buffer starts from the middle of the memory region and grows upward. The second IN buffer also starts from the middle of the memory region but grows downward.

To configure an endpoint as IN-only, set `USBMAXO` to 0, and to configure an endpoint as OUT-only, set `USBMAXI` to 0.

For unused endpoints, both `USBMAXO` and `USBMAXI` should be set to 0.

**Table 21-2. FIFO Sizes for EP 1–5**

EP Number	FIFO Size (in Bytes)
1	32
2	64
3	128
4	256
5	512



M0106-02

**Figure 21-2. IN and OUT FIFOs**

### 21.7.2 Double Buffering

To enable faster transfer and reduce the need for retransmissions, double buffering can be used. This allows two packets to be buffered in the FIFO in each direction. This is highly recommended for isochronous endpoints, which are expected to transfer one data packet every USB frame without any retransmission. For an isochronous endpoint, one data packet is sent or received every USB frame. However, the data packet may be sent or received at any time during the USB frame period, and there is a chance that two data packets may be sent or received at a few-microseconds interval. For isochronous endpoints, an incoming packet is lost if there is no buffer available, and a zero-length data packet is sent if there is no data packet ready for transmission when the USB host requests data. Double buffering is not as critical for bulk and interrupt endpoints as it is for isochronous endpoints, because packets are not lost. Double buffering, however, may improve the effective data rate for bulk endpoints.

To enable double buffering for an IN endpoint, `USBCSIH.IN_DBL_BUF` must be set to 1. To enable double buffering for an OUT endpoint, set `USBCSOH.OUT_DBL_BUF` to 1.

### 21.7.3 FIFO Access

The endpoint FIFOs are accessed by reading and writing to the registers USBF0–USBF6. Writing to a register causes the byte written to be inserted into the IN FIFO. Reading a register causes the next byte in the OUT FIFO to be extracted and the value of this byte to be returned.

When a data packet has been written to an IN FIFO, the `USBCSIL.INPKT_RDY` bit must be set to 1. If double buffering is enabled, the `USBCSIL.INPKT_RDY` bit is cleared immediately after it has been written, and another data packet can be loaded. This does not generate an IN endpoint interrupt, because an interrupt is only generated when a packet has been sent. When double buffering is used, firmware should check the status of the `USBCSIL.PKT_PRESENT` bit before writing to the IN FIFO. If this bit is 0, two data packets can be written. Double-buffered isochronous endpoints should only load two packets the first time the IN FIFO is loaded. After that, one packet is loaded for every USB frame. To send a zero-length data packet, `USBCSIL.INPKT_RDY` should be set to 1 without loading a data packet into the IN FIFO.

A data packet can be read from the OUT FIFO when the `USBCSOL.OUTPKT_RDY` bit is 1. An interrupt is generated when this occurs, if enabled. The size of the data packet is kept in the `USBCNTH:USBCNTL` registers. Note that this value is only valid when `USBCSOL.OUTPKT_RDY = 1`. When the data packet has been read from the OUT FIFO, the `USBCSOL.OUTPKT_RDY` bit must be cleared. If double buffering is enabled, there may be two data packets in the FIFO. If another data packet is ready when the `USBCSOL.OUTPKT_RDY` bit is cleared, the `USBCSOL.OUTPKT_RDY` bit is asserted immediately, and an interrupt is generated (if enabled) to signal that a new data packet has been received. The `USBCSOL.FIFO_FULL` bit is set when there are two data packets in the OUT FIFO.

The AutoClear feature is supported for OUT endpoints. When enabled, the `USBCSOL.OUTPKT_RDY` bit is cleared automatically when `USBMAXO` bytes have been read from the OUT FIFO. The AutoClear feature is enabled by setting `USBCSOH.AUTOCLEAR = 1`. The AutoClear feature can be used to reduce the time the data packet occupies the OUT FIFO buffer and is typically used for bulk endpoints.

A complementary AutoSet feature is supported for IN endpoints. When enabled, the `USBCSIL.INPKT_RDY` bit is set automatically when `USBMAXI` bytes have been written to the IN FIFO. The AutoSet feature is enabled by setting `USBCSIH.AUTOSSET = 1`. The AutoSet feature can reduce the overall time it takes to send a data packet and is typically used for bulk endpoints.

### 21.7.4 Endpoint 1–5 Interrupts

The following events may generate an IN EPx interrupt request (x indicates the endpoint number):

- A data packet that was loaded into the IN FIFO has been sent to the USB host. (`USBCSIL.INPKT_RDY` should be set to 1 when a new packet is ready to be transferred. This bit is cleared by hardware when the data packet has been sent.)
- A STALL has been sent (`USBCSIL.SENT_STALL = 1`). Only bulk and interrupt endpoints can be stalled.
- The IN FIFO is flushed due to the `USBCSIH.FLUSH_PACKET` bit being set to 1.

Any of these events causes `USBIIF.INEPxIF` to be asserted, regardless of the status of the IN EPx interrupt mask bit `USBIIE.INEPxIE`. If the IN EPx interrupt mask bit is set to 1, the CPU interrupt flag `IRCON2.P2IF` is also asserted. An interrupt request is only generated if `IEN2.P2IE` and `USBIIE.INEPxIE` are both set to 1. The x in the register name refers to the endpoint number, 1–5)

The following events may generate an OUT EPx interrupt request:

- A data packet has been received (`USBCSOL.OUTPKT_RDY = 1`).
- A STALL has been sent (`USBCSIL.SENT_STALL = 1`). Only bulk and interrupt endpoints can be stalled.

Any of these events causes `USBOIF.OUTEPxIF` to be asserted, regardless of the status of the OUT EPx interrupt mask bit `USBOIE.OUTEPxIE`. If the OUT EPx interrupt mask bit is set to 1, the CPU interrupt flag `IRCON2.P2IF` is also asserted. An interrupt request is only generated if `IEN2.P2IE` and `USBOIE.OUTEPxIE` are both set to 1.

### 21.7.5 Bulk or Interrupt IN Endpoint

Interrupt IN transfers occur at regular intervals, whereas bulk IN transfers use available bandwidth not allocated to isochronous, interrupt, or control transfers.

Interrupt IN endpoints may set the `USBCSIH.FORCE_DATA_TOG` bit. When this bit is set, the data toggle bit is continuously toggled, regardless of whether an ACK was received or not. This feature is typically used by interrupt IN endpoints that are used to communicate rate feedback for isochronous endpoints.

A bulk or interrupt IN endpoint can be stalled by setting the `USBCSIL.SEND_STALL` bit to 1. When the endpoint is stalled, the USB controller responds with a STALL handshake to IN tokens. The `USBCSIL.SEND_STALL` bit is then set, and an interrupt is generated, if enabled.

A bulk transfer longer than the maximum packet size is performed by splitting the transfer into a number of data packets of maximum size followed by a smaller data packet containing the remaining bytes. If the transfer length is a multiple of the maximum packet size, a zero-length data packet is sent last. This means that a packet with a size less than the maximum packet size denotes the end of the transfer. The AutoSet feature can be useful in this case, because many data packets are of maximum size.

### 21.7.6 Isochronous IN Endpoint

An isochronous IN endpoint is used to transfer periodic data from the USB controller to the host (one data packet every USB frame).

If there is no data packet loaded in the IN FIFO when the USB host requests data, the USB controller sends a zero-length data packet, and the `USBCSIL.UNDERRUN` bit is asserted.

Double buffering requires that a data packet is loaded into the IN FIFO during the frame preceding the frame where it should be sent. If the first data packet is loaded before an IN token is received, the data packet is sent during the same frame as it was loaded and hence violates the double-buffering strategy. Thus, when double buffering is used, the `USBPOW.ISO_WAIT_SOF` bit should be set to 1 to avoid this. Setting this bit ensures that a loaded data packet is not sent until the next SOF token has been received.

The AutoSet feature typically is not used for isochronous endpoints, because the packet size increases or decreases from frame to frame.

### 21.7.7 Bulk or Interrupt OUT Endpoint

Interrupt OUT transfers occur at regular intervals, whereas bulk OUT transfers use available bandwidth not allocated to isochronous, interrupt, or control transfers.

A bulk or interrupt OUT endpoint can be stalled by setting the `USBCSOL.SEND_STALL` bit to 1. When the endpoint is stalled, the USB controller responds with a STALL handshake when the host is done sending the data packet. The data packet is discarded and is not placed in the OUT FIFO. The USB controller asserts the `USBCSOL.SEND_STALL` bit when the STALL handshake is sent and generates an interrupt request if the OUT endpoint interrupt is enabled.

As the AutoSet feature is useful for bulk IN endpoints, the AutoClear feature is useful for OUT endpoints, because many packets are of maximum size.

### 21.7.8 Isochronous OUT Endpoint

An isochronous OUT endpoint is used to transfer periodic data from the host to the USB controller (one data packet every USB frame).

If there is no buffer available when a data packet is being received, the `USBCSOL.OVERRUN` bit is asserted and the packet data is lost. Firmware can reduce the chance for this to happen by using double buffering and using DMA to unload data packets effectively.

An isochronous data packet in the OUT FIFO may have bit errors. The hardware detects this condition and sets `USBCSOL.DATA_ERROR`. Firmware should therefore always check this bit when unloading a data packet.

The AutoClear feature typically is not used for isochronous endpoints, because the packet size increases or decreases from frame to frame.

## 21.8 DMA

DMA should be used to fill the IN endpoint FIFOs and empty the OUT endpoint FIFOs. Using DMA improves the read and write performance significantly compared to using the CPU. It is therefore highly recommended to use DMA unless timing is not critical or only a few bytes are to be transferred.

There are no DMA triggers for the USB controller, meaning that DMA transfers must be triggered by firmware.

Byte-size transfer should be used.

## 21.9 USB Reset

When reset signaling is detected on the bus, the `USBCIF.RSTIF` flag is asserted. If `USBCIE.RSTIE` is enabled, `IRCON2.P2IF` is also asserted, and an interrupt request is generated if `IEN2.P2IE = 1`. The firmware should take appropriate action when a USB reset occurs. A USB reset should place the device in the default state, where it only responds to address 0 (the default address). One or more resets normally take place during the enumeration phase, immediately after the USB cable is connected.

The following actions are performed by the USB controller when a USB reset occurs:

- `USBADDR` is set to 0.
- `USBINDEX` is set to 0.
- All endpoint FIFOs are flushed.
- `USBMAXI`, `USBCS0`, `USBCSIL`, `USBCSIH`, `USBMAXO`, `USBCSOL`, `USBCSOH`, `USBCNT0`, `USBCNTL`, and `USBCNTH` are cleared.
- All interrupts, except SOF and suspend, are enabled.
- An interrupt request is generated (if `IEN2.P2IE = 1` and `USBCIE.RSTIE = 1`).

Firmware should close all pipes and wait for a new enumeration phase when USB reset is detected.

## 21.10 Suspend and Resume

The USB controller asserts `USBCIF.SUSPENDIF` and enters suspend mode when the USB has been continuously idle for 3 ms, provided that `USBPOW.SUSPEND_EN =`

1. `IRCON2.P2IF` is asserted if `USBCIE.SUSPENDIE` is enabled, and an interrupt request is generated if `IEN2.P2IE = 1`.

While in suspend mode, only limited current can be sourced from the USB. See the USB 2.0 Specification [3] for details about this. To be able to meet the suspend-current requirement, the device should be taken down to PM1 when suspend is detected. The device should not enter PM2 or PM3, because this resets the USB controller. Before entering PM1, the 48-MHz USB PLL must be turned off. This is done by setting `USBCTRL.PLL_EN` to 0 and waiting for `USBCTRL.PLL_LOCKED` to be cleared.

Any valid nonidle signaling on the USB causes `USBCIF.RESUMEIF` to be asserted and an interrupt request to be generated, and wakes up the system if the USB resume interrupt is enabled.

When the system wakes up (enters active mode) from suspend, no USB registers except `USBCTRL` can be accessed before the 48-MHz USB PLL has been activated. This is done by setting `USBCTRL.PLL_EN` to 1 and waiting until `USBCTRL.PLL_LOCKED` is set.

A USB reset also wakes up the system from suspend. A USB resume interrupt request is generated if the interrupt is enabled, but the `USBCIF.RSTIF` interrupt flag is set instead of the `USBCIF.RESUMEIF` interrupt flag.

## 21.11 Remote Wake-Up

The USB controller can resume from suspend by signaling resume to the USB hub. Resume is performed by setting `USBPOW.RESUME` to 1 for approximately 10 ms. According to the USB 2.0 Specification [3], the resume signaling must be present for at least 1 ms and no more than 15 ms. It is, however, recommended to keep the resume signaling for approximately 10 ms. Notice that support for remote wakeup must be declared in the USB descriptor, and that the USB host must grant the device the privilege to perform remote wakeup (through a `SET_FEATURE` request).

## 21.12 USB Registers

This section describes all USB registers used for control and status for the USB. The USB registers reside in XDATA memory space in the region 0x6200–0x622B. These registers can be divided into three groups: The common USB registers, the indexed endpoint registers, and the endpoint FIFO registers. The indexed endpoint registers represent the currently selected endpoint. The `USBINDEX` register is used to select the endpoint.

The registers return to their reset values and the FIFOs are cleared when the chip enters PM2 or PM3.

### USBADDR (0x6200) – Function Address

Bit	Name	Reset	R/W	Description
7	UPDATE	0	R	This bit is set when the <code>USBADDR</code> register is written and cleared when the address becomes effective.
6:0	USBADDR[6:0]	000 0000	R/W	Device address

### USBPOW (0x6201) – Power and Control Register

Bit	Name	Reset	R/W	Description
7	ISO_WAIT_SOF	0	R/W	When this bit is set to 1, the USB controller sends zero-length data packets from the time <code>INPKT_RDY</code> is asserted and until the first SOF token has been received. This only applies to isochronous endpoints.
6:4	–	000	R0	Reserved
3	RST	0	R	During reset signaling, this bit is set to 1.
2	RESUME	0	R/W	Drives resume signaling for remote wakeup. According to the USB Specification, the duration of driving resume must be at least 1 ms and no more than 15 ms. It is recommended to keep this bit set for approximately 10 ms.
1	SUSPEND	0	R	Suspend mode entered. This bit is only used when <code>SUSPEND_EN</code> = 1. Reading the <code>USBCIF</code> register or asserting <code>RESUME</code> clears this bit.
0	SUSPEND_EN	0	R/W	Suspend enable. When this bit is set to 1, suspend mode is entered when the USB has been idle for 3 ms.

### USBIIF (0x6202) – IN Endpoints and EP0 Interrupt Flags

Bit	Name	Reset	R/W	Description
7:6	–	00	R0	Reserved
5	INEP5IF	0	R, H0	Interrupt flag for IN endpoint 5. Cleared by hardware when read
4	INEP4IF	0	R, H0	Interrupt flag for IN endpoint 4. Cleared by hardware when read
3	INEP3IF	0	R, H0	Interrupt flag for IN endpoint 3. Cleared by hardware when read
2	INEP2IF	0	R, H0	Interrupt flag for IN endpoint 2. Cleared by hardware when read
1	INEP1IF	0	R, H0	Interrupt flag for IN endpoint 1. Cleared by hardware when read
0	EP0IF	0	R, H0	Interrupt flag for endpoint 0. Cleared by hardware when read

### USBOIF (0x6204) – OUT-Endpoint Interrupt Flags

Bit	Name	Reset	R/W	Description
7:6	–	–	R0	Reserved
5	OUTEP5IF	0	R, H0	Interrupt flag for OUT endpoint 5. Cleared by hardware when read
4	OUTEP4IF	0	R, H0	Interrupt flag for OUT endpoint 4. Cleared by hardware when read
3	OUTEP3IF	0	R, H0	Interrupt flag for OUT endpoint 3. Cleared by hardware when read
2	OUTEP2IF	0	R, H0	Interrupt flag for OUT endpoint 2. Cleared by hardware when read
1	OUTEP1IF	0	R, H0	Interrupt flag for OUT endpoint 1. Cleared by hardware when read
0	–	–	R0	Reserved

**USBCIF (0x6206) – Common USB Interrupt Flags**

Bit	Name	Reset	R/W	Description
7:4	–	–	R0	Reserved
3	SOFIF	0	R, H0	Start-of-frame interrupt flag. Cleared by hardware when read
2	RSTIF	0	R, H0	Reset interrupt flag. Cleared by hardware when read
1	RESUMEIF	0	R, H0	Resume interrupt flag. Cleared by hardware when read
0	SUSPENDIF	0	R, H0	Suspend interrupt flag. Cleared by hardware when read

**USBIIE (0x6207) – IN Endpoints and EP0 Interrupt-Enable Mask**

Bit	Name	Reset	R/W	Description
7:6	–	00	R/W	Reserved. Always write 00
5	INEP5IE	1	R/W	IN endpoint-5 interrupt enable 0: Interrupt disabled 1: Interrupt enabled
4	INEP4IE	1	R/W	IN endpoint- 4 interrupt enable 0: Interrupt disabled 1: Interrupt enabled
3	INEP3IE	1	R/W	IN endpoint-3 interrupt enable 0: Interrupt disabled 1: Interrupt enabled
2	INEP2IE	1	R/W	IN endpoint-2 interrupt enable 0: Interrupt disabled 1: Interrupt enabled
1	INEP1IE	1	R/W	IN endpoint-1 interrupt enable 0: Interrupt disabled 1: Interrupt enabled
0	EPOIE	1	R/W	Endpoint-0 interrupt enable 0: Interrupt disabled 1: Interrupt enabled

**USBOIE (0x6209) – OUT Endpoints Interrupt Enable Mask**

Bit	Name	Reset	R/W	Description
7:6	–	00	R/W	Reserved. Always write 00
5	OUTEP5IE	1	R/W	OUT endpoint 5 interrupt enable 0: Interrupt disabled 1: Interrupt enabled
4	OUTEP4IE	1	R/W	OUT endpoint 4 interrupt enable 0: Interrupt disabled 1: Interrupt enabled
3	OUTEP3IE	1	R/W	OUT endpoint 3 interrupt enable 0: Interrupt disabled 1: Interrupt enabled
2	OUTEP2IE	1	R/W	OUT endpoint 2 interrupt enable 0: Interrupt disabled 1: Interrupt enabled
1	OUTEP1IE	1	R/W	OUT endpoint 1 interrupt enable 0: Interrupt disabled 1: Interrupt enabled
0	–	1	R0	Reserved

**USBCIE (0x620B) – Common USB Interrupt-Enable Mask**

Bit	Name	Reset	R/W	Description
7:4	–	–	R0	Reserved
3	SOFIE	0	R/W	Start-of-frame interrupt enable 0: Interrupt disabled 1: Interrupt enabled
2	RSTIE	1	R/W	Reset interrupt enable 0: Interrupt disabled 1: Interrupt enabled
1	RESUMEIE	1	R/W	Resume interrupt enable 0: Interrupt disabled 1: Interrupt enabled
0	SUSPENDIE	0	R/W	Suspend interrupt enable 0: Interrupt disabled 1: Interrupt enabled

**USBFRLM (0x620C) – Current Frame Number (Low Byte)**

Bit	Name	Reset	R/W	Description
7:0	FRAME[7:0]	0x00	R	Low byte of 11-bit frame number

**USBFRLH (0x620D) – Current Frame Number (High Byte)**

Bit	Name	Reset	R/W	Description
7:3	–	–	R0	Reserved
2:0	FRAME[10:8]	000	R	3 MSBs of 11-bit frame number

**USBINDEX (0x620E) – Current-Endpoint Index Register**

Bit	Name	Reset	R/W	Description
7:4	–	–	R0	Reserved
3:0	USBINDEX[3:0]	0000	R/W	Endpoint selected. Must be set to a value in the range 0–5

**USBCTRL (0x620F) – USB Control Register**

Bit	Name	Reset	R/W	Description
7	PLL_LOCKED	0	R	PLL locked status
6:3	–	–	R0	Reserved
2	–	0	R/W	Reserved. Always write 0
1	PLL_EN	0	R/W	48-MHz USB PLL enable. When this bit is set, the 48-MHz PLL is started. However, the USB must not be accessed before the PLL has locked, that is, PLL_LOCKED is 1. This bit can only be set when USB_EN = 1.  Note: The PLL must be disabled before exiting active mode and re-enabled after entering active mode.
0	USB_EN	0	R/W	USB enable. The USB controller is reset when writing 0 to this bit.

**USBMAXI (0x6210) – Maximum Packet Size for IN Endpoint{1–5}**

Bit	Name	Reset	R/W	Description
7:0	USBMAXI[7:0]	0x00	R/W	Maximum packet size, in units of 8 bytes, for IN endpoint selected by USBINDEX register. The value of this register should correspond to the <i>wMaxPacketSize</i> field in the standard endpoint descriptor for the endpoint. This register must not be set to a value greater than the available FIFO memory for the endpoint.



**USBCS0 (0x6211) – EP0 Control and Status (USBINDEX = 0)**

Bit	Name	Reset	R/W	Description
7	CLR_SETUP_END	0	R/W H0	Set this bit to 1 to de-assert the SETUP_END bit of this register. This bit is cleared automatically.
6	CLR_OUTPKT_RDY	0	R/W H0	Set this bit to 1 to de-assert the OUTPKT_RDY bit of this register. This bit is cleared automatically.
5	SEND_STALL	0	R/W H0	Set this bit to 1 to terminate the current transaction. The USB controller sends the STALL handshake and this bit is de-asserted.
4	SETUP_END	0	R	This bit is set if the control transfer ends due to a premature end-of-control transfer. The FIFO is flushed and an interrupt request (EP0) is generated if the interrupt is enabled. Setting CLR_SETUP_END = 1 de-asserts this bit.
3	DATA_END	0	R/W H0	This bit is used to signal the end of a data transfer and must be asserted in the following three situations: <ol style="list-style-type: none"> <li>1: When the last data packet has been loaded and USBCS0 . INPKT_RDY is set to 1</li> <li>2: When the last data packet has been unloaded and USBCS0 . CLR_OUTPKT_RDY is set to 1</li> <li>3: When USBCS0 . INPKT_RDY has been asserted without having loaded the FIFO (for sending a zero-length data packet).</li> </ol> The USB controller clears this bit automatically.
2	SENT_STALL	0	R/W H1	This bit is set when a STALL handshake has been sent. An interrupt request (EP0) is generated if the interrupt is enabled. This bit must be cleared from firmware.
1	INPKT_RDY	0	R/W H0	Set this bit when a data packet has been loaded into the EP0 FIFO to notify the USB controller that a new data packet is ready to be transferred. When the data packet has been sent, this bit is cleared, and an interrupt request (EP0) is generated if the interrupt is enabled.
0	OUTPKT_RDY	0	R	Data packet received. This bit is set when an incoming data packet has been placed in the OUT FIFO. An interrupt request (EP0) is generated if the interrupt is enabled. Set CLR_OUTPKT_RDY = 1 to de-assert this bit.

**USBCSIL (0x6211) – IN EP{1–5} Control and Status, Low**

Bit	Name	Reset	R/W	Description
7	–	–	R0	Reserved
6	CLR_DATA_TOG	0	R/W H0	Setting this bit resets the data toggle to 0. Thus, setting this bit forces the next data packet to be a DATA0 packet. This bit is automatically cleared.
5	SENT_STALL	0	R/W	This bit is set when a STALL handshake has been sent. The FIFO is flushed and the INPKT_RDY bit in this register is de-asserted. An interrupt request (IN EP{1–5}) is generated if the interrupt is enabled. This bit must be cleared from firmware.
4	SEND_STALL	0	R/W	Set this bit to 1 to make the USB controller reply with a STALL handshake when receiving IN tokens. Firmware must clear this bit to end the STALL condition. It is not possible to stall an isochronous endpoint; thus, this bit only has an effect if the IN endpoint is configured as bulk or interrupt.
3	FLUSH_PACKET	0	R/W H0	Set to 1 to flush next packet that is ready to transfer from the IN FIFO. The INPKT_RDY bit in this register is cleared. If there are two packets in the IN FIFO due to double buffering, this bit must be set twice to completely flush the IN FIFO. This bit is automatically cleared.
2	UNDERRUN	0	R/W	In isochronous mode, this bit is set if an IN token is received when INPKT_RDY = 0, and a zero-length data packet is transmitted in response to the IN token. In bulk or interrupt mode, this bit is set when a NAK is returned in response to an IN token. Firmware should clear this bit.
1	PKT_PRESENT	0	R	This bit is 1 when there is at least one packet in the IN FIFO.
0	INPKT_RDY	0	R/W H0	Set this bit when a data packet has been loaded into the IN FIFO to notify the USB controller that a new data packet is ready to be transferred. When the data packet has been sent, this bit is cleared, and an interrupt request (IN EP{1–5}) is generated if the interrupt is enabled.

**USBCSIH (0x6212) – IN EP{1–5} Control and Status, High**

Bit	Name	Reset	R/W	Description
7	AUTOSET	0	R/W	When this bit is 1, the USBCSIL.INPKT_RDY bit is automatically asserted when a data packet of maximum size (specified by USBMAXI) has been loaded into the IN FIFO.
6	ISO	0	R/W	Selects IN endpoint type 0: Bulk or interrupt 1: Isochronous
5:4		10	R/W	Reserved. Always write 10
3	FORCE_DATA_TOG	0	R/W	Setting this bit forces the IN endpoint data toggle to switch and the data packet to be flushed from the IN FIFO, even though an ACK was received. This feature can be useful when reporting rate feedback for isochronous endpoints.
2:1		–	R0	Reserved
0	IN_DBL_BUF	0	R/W	Double buffering enable (IN FIFO) 0: Double buffering disabled 1: Double buffering enabled

**USBMAXO (0x6213) – Max. Packet Size for OUT EP{1–5}**

Bit	Name	Reset	R/W	Description
7:0	USBMAXO[7:0]	0x00	R/W	Maximum packet size, in units of 8 bytes, for OUT endpoint selected by USBINDEX register. The value of this register should correspond to the wMaxPacketSize field in the standard endpoint descriptor for the endpoint. This register must not be set to a value greater than the available FIFO memory for the endpoint.

**USBCSOL (0x6214) – OUT EP{1–5} Control and Status, Low**

Bit	Name	Reset	R/W	Description
7	CLR_DATA_TOG	0	R/W H0	Setting this bit resets the data toggle to 0. Thus, setting this bit forces the next data packet to be a DATA0 packet. This bit is automatically cleared.
6	SENT_STALL	0	R/W	This bit is set when a STALL handshake has been sent. An interrupt request (OUT EP{1–5}) is generated if the interrupt is enabled. This bit must be cleared from firmware.
5	SEND_STALL	0	R/W	Set this bit to 1 to make the USB controller reply with a STALL handshake when receiving OUT tokens. Firmware must clear this bit to end the STALL condition. It is not possible to stall an isochronous endpoint; thus, this bit only has an effect if the IN endpoint is configured as bulk or interrupt.
4	FLUSH_PACKET	0	R/W H0	Set to 1 to flush the next packet that is to be read from the OUT FIFO. The OUTPKT_RDY bit in this register is cleared. If there are two packets in the OUT FIFO due to double buffering, this bit must be set twice to completely flush the OUT FIFO. This bit is automatically cleared after a write to 1.
3	DATA_ERROR	0	R	This bit is set if there is a CRC or bit-stuff error in the packet received. Cleared when OUTPKT_RDY is cleared. This bit is only valid if the OUT endpoint is isochronous.
2	OVERRUN	0	R/W	This bit is set when an OUT packet cannot be loaded into the OUT FIFO. Firmware should clear this bit. This bit is only valid in isochronous mode.
1	FIFO_FULL	0	R	This bit is asserted when no more packets can be loaded into the OUT FIFO because it is full.
0	OUTPKT_RDY	0	R/W	This bit is set when a packet has been received and is ready to be read from the OUT FIFO. An interrupt request (OUT EP{1–5}) is generated if the interrupt is enabled. This bit should be cleared when the packet has been unloaded from the FIFO.

**USBCSOH (0x6215) – OUT EP{1–5} Control and Status, High**

Bit	Name	Reset	R/W	Description
7	AUTOCLEAR	0	R/W	When this bit is set to 1, the USBCSOL.OUTPKT_RDY bit is automatically cleared when a data packet of maximum size (specified by USBMAXO) has been unloaded to the OUT FIFO.
6	ISO	0	R/W	Selects OUT endpoint type 0 Bulk or interrupt 1 Isochronous
5:4		00	R/W	Reserved. Always write 00
3:1		–	R0	Reserved
0	OUT_DBL_BUF	0	R/W	Double buffering enable (OUT FIFO) 0 Double buffering disabled 1 Double buffering enabled

**USBCNT0 (0x6216) – Number of Received Bytes in EP0 FIFO (USBINDEX = 0)**

Bit	Name	Reset	R/W	Description
7:6	–	–	R0	Reserved
5:0	USBCNT0[5:0]	00 0000	R	Number of received bytes into EP 0 FIFO. Only valid when OUTPKT_RDY is asserted

**USBCNTL (0x6216) – Number of Bytes in EP{1–5} OUT FIFO, Low**

Bit	Name	Reset	R/W	Description
7:0	USBCNT[7:0]	0x00	R	8 LSBs of number of received bytes in OUT FIFO selected by USBINDEX register. Only valid when USBCSOL.OUTPKT_RDY is asserted.

**USBCNTH (0x6217) – Number of Bytes in EP{1–5} OUT FIFO, High**

Bit	Name	Reset	R/W	Description
7:3	–	–	R0	Reserved
2:0	USBCNT[10:8]	000	R	3 MSBs of number of received bytes in OUT FIFO selected by USBINDEX register. Only valid when USBCSOL.OUTPKT_RDY is set

**USBF0 (0x6220) – Endpoint 0 FIFO**

Bit	Name	Reset	R/W	Description
7:0	USBF0[7:0]	0x00	R/W	Endpoint 0 FIFO. Reading this register unloads one byte from the EP0 FIFO. Writing to this register loads one byte into the EP0 FIFO. <i>Note: The FIFO memory for EP0 is used for both incoming and outgoing data packets.</i>

**USBF1 (0x6222) – Endpoint 1 FIFO**

Bit	Name	Reset	R/W	Description
7:0	USBF1[7:0]	0x00	R/W	Endpoint 1 FIFO register. Reading this register unloads one byte from the EP1 OUT FIFO. Writing to this register loads one byte into the EP1 IN FIFO.

**USBF2 (0x6224) – Endpoint 2 FIFO**

Bit	Name	Reset	R/W	Description
7:0	USBF2[7:0]	0x00	R/W	Endpoint 2 FIFO register. Reading this register unloads one byte from the EP2 OUT FIFO. Writing to this register loads one byte into the EP2 IN FIFO.

**USBF3 (0x6226) – Endpoint 3 FIFO**

Bit	Name	Reset	R/W	Description
7:0	USBF3[7:0]	0x00	R/W	Endpoint 3 FIFO register. Reading this register unloads one byte from the EP3 OUT FIFO. Writing to this register loads one byte into the EP3 IN FIFO.

**USBF4 (0x6228) – Endpoint 4 FIFO**

Bit	Name	Reset	R/W	Description
7:0	USBF4[7:0]	0x00	R/W	Endpoint 4 FIFO register. Reading this register unloads one byte from the EP4 OUT FIFO. Writing to this register loads one byte into the EP4 IN FIFO.

**USBF5 (0x622A) – Endpoint 5 FIFO**

Bit	Name	Reset	R/W	Description
7:0	USBF5[7:0]	0x00	R/W	Endpoint 5 FIFO register. Reading this register unloads one byte from the EP5 OUT FIFO. Writing to this register loads one byte into the EP5 IN FIFO.

## Timer 2 (MAC Timer)

Timer 2 is mainly used to provide timing for 802.15.4 command-strobe-processor algorithms and for general timekeeping in the 802.15.4 MAC layer on CC253x devices, for timekeeping in the BLE link layer on CC2540 and CC2541, and for general radio timekeeping when running the radio in proprietary mode on CC2541. Timer 2 must not be used by the application on the CC2540 or CC2541 when the BLE stack is running. When Timer 2 is used together with the Sleep Timer, the timing function is provided even when the system enters low-power modes PM1 and PM2. The timer runs at a speed according to the system clock. If Timer 2 is to be used with the Sleep Timer, the system clock source must be the 32-MHz crystal whenever Timer 2 is running, and an external 32-kHz XOSC should be used for accurate results.

The main features of Timer 2 are the following:

- 16-bit timer up-counter providing, for example, a symbol period of 16  $\mu$ s or a frame period of 320  $\mu$ s
- Adjustable period with accuracy of 31.25 ns
- 2  $\times$  16-bit timer compare function
- 24-bit overflow count
- 2  $\times$  24-bit overflow compare function
- Start-of-frame-delimiter capture function
- Timer start and stop synchronous with 32-kHz clock and timekeeping maintained by Sleep Timer.
- Interrupts generated on compare and overflow
- DMA trigger capability
- Possible to adjust timer value while counting by introducing delay

Topic	Page
<b>22.1 Timer Operation</b> .....	<b>198</b>
<b>22.2 Interrupts</b> .....	<b>199</b>
<b>22.3 Event Outputs (DMA Trigger and Radio Events)</b> .....	<b>200</b>
<b>22.4 Timer Start-and-Stop Synchronization</b> .....	<b>200</b>
<b>22.5 Timer 2 Registers</b> .....	<b>202</b>

## 22.1 Timer Operation

This section describes the operation of the timer.

### 22.1.1 General

After a reset, the timer is in the timer IDLE mode, where it is stopped. The timer starts running when `T2CTRL.RUN` is set to 1. The timer then enters the timer RUN mode. Either the entry is immediate, or it is performed synchronously with the 32-kHz clock. See [Section 22.4](#) for a description of the synchronous start-and-stop mode.

Once the timer is running in RUN mode, it can be stopped by writing a 0 to `T2CTRL.RUN`. The timer then enters the timer IDLE mode. The stopping of the timer is performed either immediately or synchronously with the 32-kHz clock.

### 22.1.2 Up Counter

Timer 2 contains a 16-bit timer, which increments on each clock cycle. The counter value can be read from registers `T2M1:T2M0` with register `T2MSEL.T2MSEL` set to 000. Note that the register content in `T2M1` is latched when `T2M0` is read, meaning that `T2M0` must always be read first.

When the timer is idle, the counter can be modified by writing to registers `T2M1:T2M0` with register `T2MSEL.T2MSEL` set to 000. `T2M0` must be written first.

### 22.1.3 Timer Overflow

At the same time as the timer counts to a value that is equal to the set timer period, a timer overflow occurs. When the timer overflow occurs, the timer is set to 0x0000. If the overflow interrupt mask bit `T2IRQM.TIMER2_PERM` is 1, an interrupt request is generated. The interrupt flag bit `T2IRQF.TIMER2_PERF` is set to 1, regardless of the interrupt mask value.

### 22.1.4 Timer Delta Increment

The timer period may be adjusted once during a timer period by writing a timer delta value. When the timer is running and a timer delta value is written to multiplexed registers `T2M1:T2M0` with `T2MSEL.T2MSEL` set to 000, the 16-bit timer halts at its current value and a delta counter starts counting. The `T2M0` register must be written before `T2M1`. The delta counter starts counting from the delta value written, down to zero. Once the delta counter reaches zero, the 16-bit timer starts counting again.

The delta counter decrements at the same rate as the timer. When the delta counter has reached zero, it does not start counting again until a delta value is written once again. In this way, a timer period may be increased by the delta value in order to make adjustments to the timer overflow events over time.

### 22.1.5 Timer Compare

A timer compare occurs at the same time as the timer counts to a value that is equal to one of the 16-bit compare values set. When a timer compare occurs, the interrupt flag `T2IRQF.TIMER2_COMPARE1F` or `T2IRQF.TIMER2_COMPARE2F` is set to 1, depending of which compare value is reached. An interrupt request is also generated if the corresponding interrupt mask in `T2IRQM.TIMER2_COMPARE1M` or `T2IRQM.TIMER2_COMPARE2M` is set to 1.

### 22.1.6 Overflow Count

At each timer overflow, the 24-bit overflow counter is incremented by 1. The overflow counter value is read through registers `T2MOV2:T2MOV1:T2MOV0` with register `T2MSEL.T2MOVEFSEL` set to 000. The registers are latched as in the following description.

If one wants a unique timestamp, where both timer and overflow counter are latched at the same time, do the following: Read `T2M0` with `T2MSEL.T2MSEL` set to 000 and `T2CTRL.LATCH_MODE` set to 1. This returns the low byte of the timer value, and also latches the high byte of the timer and the entire overflow counter, so the rest of the timestamp is ready to be read.

If one wants to read just the overflow counter without reading timer first, read `T2MOVF0` with `T2MSEL.T2MOVFSSEL` set to 000 and `T2CTRL.LATCH_MODE` set to 0. This returns the low byte of the overflow counter, and latches the two most-significant bytes of the overflow counter so the values are ready to be read.

### 22.1.7 Overflow-Count Update

The overflow count value can be updated by writing to registers `T2MOVF2:T2MOVF1:T2MOVF0` with `T2MSEL.T2MOVFSSEL` set to 000. Always write the least-significant byte first, and always write all three bytes. The write takes effect once the high byte is written.

### 22.1.8 Overflow-Count Overflow

At the same time as the overflow counter counts to a value that is equal to the overflow period setting, an overflow period event occurs. When the period event occurs, the overflow counter is set to 0x00 0000. If the overflow interrupt mask bit `T2IRQM.TIMER2_OVF_PERM` is 1, an interrupt request is generated. The interrupt flag bit `T2IRQF.TIMER2_OVF_PERF` is set to 1, regardless of the interrupt mask value.

### 22.1.9 Overflow-Count Compare

Two compare values may be set for the overflow counter. The compare values are set by writing to `T2MOVF2:T2MOVF1:T2MOVF0` with register `T2MSEL.T2MOVFSSEL` set to 011 or 100. At the same time as the overflow counter counts to a value equal to one of the overflow count compare values, an overflow count compare event occurs. If the corresponding overflow compare interrupt mask bit `T2IRQM.TIMER2_OVF_COMPARE1M` or `T2IRQM.TIMER2_OVF_COMPARE2M` is 1, an interrupt request is generated. The interrupt flag bits `T2IRQF.TIMER2_OVF_COMPARE1F` and `T2IRQF.TIMER2_OVF_COMPARE2F` are set to 1, regardless of the interrupt mask value.

### 22.1.10 Capture Input

Timer 2 has a timer capture function, which captures the time when the start-of-frame delimiter (SFD) status in the radio goes high.

When the capture event occurs, the current timer value is captured in the capture register. The capture value can be read from registers `T2M1:T2M0` if register `T2MSEL.T2MSEL` is set to 001. The value of the overflow count is also captured at the time of the capture event and can be read from registers `T2MOVF2:T2MOVF1:T2MOVF0` if `T2MSEL.T2MOVFSSEL` is set to 001.

### 22.1.11 Long Compare (CC2541 Only)

In the CC2541, two compare values may be set for the combination of the 16-bit timer and the overflow counter. The compare values are a combination of either timer compare 1 and overflow compare 1, or timer compare 2 and overflow compare 2. These combinations are known as the long compare 1 and long compare 2 values, respectively. At the same time as the combination of the 16-bit timer and the 24-bit overflow counter counts to a value equal to one of the long compare values, a long compare event occurs. If the corresponding overflow compare interrupt mask bit `T2IRQM.TIMER2_LONG_COMPARE1M` or `T2IRQM.TIMER2_LONG_COMPARE2M` is 1, an interrupt request is generated. The corresponding interrupt flag bit `T2IRQF.TIMER2_LONG_COMPARE1F` or `T2IRQF.TIMER2_LONG_COMPARE2F` is set to 1, regardless of the interrupt mask value.

## 22.2 Interrupts

The timer has six (eight on CC2541) individually maskable interrupt sources. These are the following:

- Timer overflow
- Timer compare 1
- Timer compare 2
- Overflow-count overflow
- Overflow-count compare 1
- Overflow-count compare 2

- Long compare 1 (CC2541 only)
- Long compare 2 (CC2541 only)

The interrupt flags are given in the `T2IRQF` registers. The interrupt flag bits are set only by hardware and can be cleared only by writing to the SFR register.

Each interrupt source can be masked by its corresponding mask bit in the `T2IRQM` register. An interrupt is generated when the corresponding mask bit is set; otherwise, the interrupt is not generated. The interrupt flag bit is set, however, regardless of the state of the interrupt mask bit.

### 22.3 Event Outputs (DMA Trigger and Radio Events)

Timer 2 has two event outputs, `T2_EVENT1` and `T2_EVENT2`. These can be used as DMA triggers, as inputs to the radio, for conditions in conditional instructions in the CSP on CC253x, for use by the BLE stack on CC2540 and CC2541, or for timing TX or RX in CC2541 when running the radio in proprietary mode. The event outputs can be configured individually to any of the following events:

- Timer overflow
- Timer compare 1
- Timer compare 2
- Overflow-count overflow
- Overflow-count compare 1
- Overflow-count compare 2
- Long compare 1 (CC2541 only)
- Long compare 2 (CC2541 only)

The DMA triggers are configured using `T2EVTCFG.TIMER2_EVENT1_CFG` and `T2EVTCFG.TIMER2_EVENT2_CFG`.

### 22.4 Timer Start-and-Stop Synchronization

This section describes the synchronized timer start and stop.

#### 22.4.1 General

The timer can be started and stopped synchronously with the 32-kHz clock rising edge. Note that this event is derived from a 32-kHz clock signal, but is synchronous with the 32-MHz system clock and thus has a period approximately equal to that of the 32-kHz clock period. Synchronous starting and stopping must not be attempted unless both the 32-kHz clock and 32-MHz XOSC are running and stable.

At the time of a synchronous start, the timer is reloaded with new calculated values for the timer and overflow count such that it appears that the timer has not been stopped.

#### 22.4.2 Timer Synchronous Stop

After the timer has started running, that is, entered timer RUN mode, it is stopped synchronously by writing 0 to `T2CTRL.RUN` when `T2CTRL.SYNC` is 1. After `T2CTRL.RUN` has been set to 0, the timer continues running until the 32-kHz clock rising edge is sampled as 1. When this occurs, the timer is stopped, the current Sleep Timer value is stored, and `T2CTRL.STATE` goes from 1 to 0.



### 22.4.3 Timer Synchronous Start

When the timer is in the IDLE mode, it is started synchronously by writing 1 to `T2CTRL.RUN` when `T2CTRL.SYNC` is 1. After `T2CTRL.RUN` has been set to 1, the timer remains in the IDLE mode until the 32-kHz clock rising edge is detected. When this occurs, the timer first calculates new values for the 16-bit timer value and for the 24-bit timer overflow count, based on the current and stored Sleep Timer values and the current 16-bit timer values. The new Timer 2 and overflow count values are loaded into the timer, and the timer enters the RUN mode. `T2CTRL.STATE = 1` indicates that the module is running. This synchronous start process takes 86 clock cycles from the time when the 32-kHz clock rising edge is sampled high. The synchronous start-and-stop function requires that the system clock frequency is selected to be 32 MHz. If the 16-MHz clock is selected, an offset is added to the new calculated value.

If a synchronous start is done without a previous synchronous stop, the timer is loaded with unpredictable values. To avoid this, do the first start of the timer asynchronously, then enable synchronous mode for subsequent stops and starts.

The method for calculating the new Timer 2 value and overflow-count value is given as follows. Because the Timer 2 and Sleep Timer clocks are asynchronous with a noninteger clock ratio, there is an error of maximum  $\pm 1$  in the calculated timer value compared to the ideal timer value, not taking clock inaccuracies into account.

**Calculation of New Timer Value and Overflow Count Value**
 $N_c = \text{Current Sleep Timer value}$ 
 $N_{ST} = \text{Stored Sleep Timer value}$ 
 $K_{ck} = \text{Clock ratio} = 976.5625^{(1)}$ 
 $stw = \text{Sleep Timer width} = 24$ 
 $P_T = \text{Timer 2 period}$ 
 $P_{OVF} = \text{Overflow period}$ 
 $O_{ST} = \text{Stored overflow-count value}$ 
 $O_{TICK} = \text{Overflow ticks while sleeping}$ 
 $t_{ST} = \text{Stored timer value}$ 
 $T_{OH} = \text{Overhead} = 86$ 
 $N_t = N_c - N_{ST}$ 
 $N_t \leq 0 \rightarrow N_d = 2^{stw} + N_t; N_t > 0 \rightarrow N_d = N_t$ 
 $C = N_d \times K_{ck} + T_{ST} + T_{OH}$  (rounded to nearest integer value)

 $T = C \bmod P_T$ 
 $\text{Timer2Value} = T$ 

$$O_{TICK} = \frac{(C - T)}{P_T}$$

 $O = (O_{TICK} + O_{ST}) \bmod P_{OVF}$ 
 $\text{Timer2OverflowCount} = O$ 
<sup>(1)</sup> Clock ratio of Timer 2 clock frequency (32 MHz) and Sleep Timer clock frequency (32 kHz)

For a given Timer 2 period value,  $P_T$ , there is a maximum duration between Timer 2 synchronous stop and start for which the timer value is correctly updated after starting. The maximum value is given in terms of the number of Sleep Timer clock periods, that is, 32-kHz clock periods,  $t_{ST(max)}$ .

$$t_{ST(max)} \leq \frac{(2^{24} - 1) \times P_T + T_{OH}}{K_{ck}}$$

## 22.5 Timer 2 Registers

The SFR registers associated with Timer 2 are listed in this section. These registers are the following:

- T2MSEL – Timer 2 multiplexed register control
- T2M1 – Timer 2 multiplexed count high
- T2M0 – Timer 2 multiplexed count low
- T2MOV2 – Timer 2 multiplexed overflow count 2
- T2MOV1 – Timer 2 multiplexed overflow count 1
- T2MOV0 – Timer 2 multiplexed overflow count 0
- T2IRQF – Timer 2 interrupt flags
- T2IRQM – Timer 2 interrupt masks
- T2EVTCFG – Timer 2 event output configuration
- T2CTRL – Timer 2 configuration

Timer 2 has several multiplexed registers. This is to be able to fit all the registers into the limited SFR address space. The internal registers listed in [Table 22-1](#) can be accessed indirectly through T2M0, T2M1, T2MOVF0, T2MOVF1, and T2MOVF2.

**Table 22-1. Internal Registers**

Register Name	Reset	R/W	Function
t2tim[15:0]	0x0000	R/W	Holds the 16-bit upcounter
t2_cap[15:0]	0x0000	R	Holds the last captured value of the upcounter
t2_per[15:0]	0x0000	R/W	Holds the period of the upcounter
t2_cmp1[15:0]	0x0000	R/W	Holds compare value 1 for the upcounter
t2_cmp2[15:0]	0x0000	R/W	Holds compare value 2 for the upcounter
t2ovf[23:0]	0x00 0000	R/W	Holds the 24-bit overflow counter
t2ovf_cap[23:0]	0x00 0000	R	Holds the last captured value of the overflow counter
t2ovf_per[23:0]	0x00 0000	R/W	Holds the period of the overflow counter
t2ovf_cmp1[23:0]	0x00 0000	R/W	Holds compare value 1 for the overflow counter
t2ovf_cmp2[23:0]	0x00 0000	R/W	Holds compare value 2 for the overflow counter

The registers listed in the remainder of this section are directly accessible in the SFR address space.

**T2MSEL (0xC3) – Timer 2 Multiplex Select**

Bit No.	Name	Reset	R/W	Function
7	–	0	R0	Reserved. Read as 0
6:4	T2MOVFSEL	000	R/W	The value of this register selects the internal registers that are modified or read when accessing T2MOVF0, T2MOVF1, and T2MOVF2. 000: <b>t2ovf</b> (overflow counter) 001: <b>t2ovf_cap</b> (overflow capture) 010: <b>t2ovf_per</b> (overflow period) 011: <b>t2ovf_cmp1</b> (overflow compare 1) 100: <b>t2ovf_cmp2</b> (overflow compare 2) 101 to 111: Reserved
3	–	0	R0	Reserved. Read as 0
2:0	T2MSEL	000	R/W	The value of this register selects the internal registers that are modified or read when accessing T2M0 and T2M1. 000: <b>t2tim</b> (timer count value) 001: <b>t2_cap</b> (timer capture) 010: <b>t2_per</b> (timer period) 011: <b>t2_cmp1</b> (timer compare 1) 100: <b>t2_cmp2</b> (timer compare 2) 101 to 111: Reserved

**T2M0 (0xA2) – Timer 2 Multiplexed Register 0**

Bit No.	Name	Reset	R/W	Function
7:0	T2M0	0x00	R/W	Indirectly returns/modifies bits [7:0] of an internal register depending on the T2MSEL.T2MSEL value. When reading the T2M0 register with T2MSEL.T2MSEL set to 000 and T2CTRL.LATCH_MODE set to 0, the timer (t2tim) value is latched. When reading the T2M0 register with T2MSEL.T2MSEL set to 000 and T2CTRL.LATCH_MODE set to 1, the timer (t2tim) and overflow counter (t2ovf) values are latched.

**T2M1 (0xA3) – Timer 2 Multiplexed Register 1**

Bit No.	Name	Reset	R/W	Function
7:0	T2M1	0x00	R/W	Indirectly returns or modifies bits [15:8] of an internal register, depending on T2MSEL.T2MSEL value. When reading the T2M0 register with T2MSEL.T2MSEL set to 000, the timer ( <b>t2tim</b> ) value is latched. Reading this register with T2MSEL.T2MSEL set to 000 returns the latched value of <b>t2tim</b> [15:8].

**T2MOVFO (0xA4) – Timer 2 Multiplexed Overflow Register 0**

Bit No.	Name	Reset	R/W	Function
7:0	T2MOVFO	0x00	R/W	Indirectly returns or modifies bits [7:0] of an internal register, depending on the T2MSEL.T2MOVFSSEL value. When reading the T2MOVFO register with T2MSEL.T2MOVFSSEL set to 000 and T2CTRL.LATCH_MODE set to 0, the overflow counter value ( <b>t2ovf</b> ) is latched. When reading the T2M0 register with T2MSEL.T2MOVFSSEL set to 000 and T2CTRL.LATCH_MODE set to 1, the overflow counter value ( <b>t2ovf</b> ) is latched.

**T2MOVF1 (0xA5) – Timer 2 Multiplexed Overflow Register 2**

Bit No.	Name	Reset	R/W	Function
7:0	T2MOVF1	0x00	R/W	Indirectly returns/modifies bits [15:8] of an internal register, depending on the T2MSEL.T2MOVFSSEL value. Reading this register with T2MSEL.T2MOVFSSEL set to 000 returns the latched value of <b>t2ovf</b> [15:8].

**T2MOVF2 (0xA6) – Timer 2 Multiplexed Overflow Register 2**

Bit No.	Name	Reset	R/W	Function
7:0	T2MOVF2	0x00	R/W	Indirectly returns/modifies bits [23:16] of an internal register, depending on the T2MSEL.T2MOVFSSEL value. Reading this register with T2MSEL.T2MOVFSSEL set to 000 returns the latched value of <b>t2ovf</b> [23:16].

**T2IRQF (0xA1) – Timer 2 Interrupt Flags (CC253x and CC2540)**

Bit No.	Name	Reset	R/W	Function
7:6	–	00	R0	Reserved. Read as 0
5	TIMER2_OVF_COMPARE2F	0	R/W0	Set when the Timer 2 overflow counter counts to the value set at <b>t2ovf_cmp2</b>
4	TIMER2_OVF_COMPARE1F	0	R/W0	Set when the Timer 2 overflow counter counts to the value set at Timer 2 <b>t2ovf_cmp1</b>
3	TIMER2_OVF_PERF	0	R/W0	Set when the Timer 2 overflow counter would have counted to a value equal to <b>t2ovf_per</b> , but instead wraps to 0
2	TIMER2_COMPARE2F	0	R/W0	Set when the Timer 2 counter counts to the value set at <b>t2_cmp2</b>
1	TIMER2_COMPARE1F	0	R/W0	Set when the Timer 2 counter counts to the value set at <b>t2_cmp1</b>
0	TIMER2_PERF	0	R/W0	Set when the Timer 2 counter would have counted to a value equal to <b>t2_per</b> , but instead wraps to 0.

**T2IRQF (0xA1) – Timer 2 Interrupt Flags (CC2541 Only)**

Bit No.	Name	Reset	R/W	Function
7	TIMER2_LONG_COMPARE2F	0	R/W0	Set when the Timer 2 overflow counter is equal to <b>t2ovf_cmp2</b> and the timer counts to the value set at <b>t2_cmp2</b>
6	TIMER2_LONG_COMPARE1F	0	R/W0	Set when the Timer 2 overflow counter is equal to <b>t2ovf_cmp1</b> and the timer counts to the value set at <b>t2_cmp1</b>
5	TIMER2_OVF_COMPARE2F	0	R/W0	Set when the Timer 2 overflow counter counts to the value set at <b>t2ovf_cmp2</b>
4	TIMER2_OVF_COMPARE1F	0	R/W0	Set when the Timer 2 overflow counter counts to the value set at Timer 2 <b>t2ovf_cmp1</b>
3	TIMER2_OVF_PERF	0	R/W0	Set when the Timer 2 overflow counter would have counted to a value equal to <b>t2ovf_per</b> , but instead wraps to 0
2	TIMER2_COMPARE2F	0	R/W0	Set when the Timer 2 counter counts to the value set at <b>t2_cmp2</b>
1	TIMER2_COMPARE1F	0	R/W0	Set when the Timer 2 counter counts to the value set at <b>t2_cmp1</b>
0	TIMER2_PERF	0	R/W0	Set when the Timer 2 counter would have counted to a value equal to <b>t2_per</b> , but instead wraps to 0.

**T2IRQM (0xA7) – Timer 2 Interrupt Mask (CC253x and CC2540)**

Bit No.	Name	Reset	R/W	Function
7:6	–	00	R0	Reserved. Read as 0
5	TIMER2_OVF_COMPARE2M	0	R/W	Enables the TIMER2_OVF_COMPARE2 interrupt
4	TIMER2_OVF_COMPARE1M	0	R/W	Enables the TIMER2_OVF_COMPARE1 interrupt
3	TIMER2_OVF_PERM	0	R/W	Enables the TIMER2_OVF_PER interrupt
2	TIMER2_COMPARE2M	0	R/W	Enables the TIMER2_COMPARE2 interrupt
1	TIMER2_COMPARE1M	0	R/W	Enables the TIMER2_COMPARE1 interrupt
0	TIMER2_PERM	0	R/W	Enables the TIMER2_PER interrupt

**T2IRQM (0xA7) – Timer 2 Interrupt Mask (CC2541 Only)**

Bit No.	Name	Reset	R/W	Function
7	TIMER2_LONG_COMPARE2M	0	R/W	Enables the TIMER2_LONG_COMPARE2F interrupt
6	TIMER2_LONG_COMPARE1M	0	R/W	Enables the TIMER2_LONG_COMPARE1F interrupt
5	TIMER2_OVF_COMPARE2M	0	R/W	Enables the TIMER2_OVF_COMPARE2 interrupt
4	TIMER2_OVF_COMPARE1M	0	R/W	Enables the TIMER2_OVF_COMPARE1 interrupt
3	TIMER2_OVF_PERM	0	R/W	Enables the TIMER2_OVF_PER interrupt
2	TIMER2_COMPARE2M	0	R/W	Enables the TIMER2_COMPARE2 interrupt
1	TIMER2_COMPARE1M	0	R/W	Enables the TIMER2_COMPARE1 interrupt
0	TIMER2_PERM	0	R/W	Enables the TIMER2_PER interrupt

**T2CTRL (0x94) – Timer 2 Control Register**

Bit No.	Name	Reset	R/W	Function
7:4	–	0000	R0	Reserved. Read as 0
3	LATCH_MODE	0	R/W	0: Reading T2M0 with T2MSEL.T2MSEL = 000 latches the high byte of the timer, making it ready to be read from T2M1. Reading T2MOVF0 with T2MSEL.T2MOVFSSEL = 000 latches the two most-significant bytes of the overflow counter, making it possible to read these from T2MOVF1 and T2MOVF2. 1: Reading T2M0 with T2MSEL.T2MSEL = 000 latches the high byte of the timer and the entire overflow counter at once, making it possible to read the values from T2M1, T2MOVF0, T2MOVF1, and T2MOVF2.
2	STATE	0	R	State of Timer 2 0: Timer idle 1: Timer running
1	SYNC	1	R/W	0: Starting and stopping of timer is immediate, i.e., synchronous with clk_rf_32m. 1: Starting and stopping of timer happens at the first positive edge of the 32-kHz clock. Read <a href="#">Section 22.4</a> for more details regarding timer start and stop.
0	RUN	0	R/W	Write 1 to start timer, write 0 to stop timer. When read, it returns the last written value.

**T2EVTCFG (0x9C) – Timer 2 Event Configuration (CC253x and CC2540)**

Bit No.	Name	Reset	R/W	Function
7	–	0	R0	Reserved. Read as 0
6:4	TIMER2_EVENT2_CFG	000	R/W	Selects the event that triggers a T2_EVENT2 pulse 000: t2_per_event 001: t2_cmp1_event 010: t2_cmp2_event 011: t2ovf_per_event 100: t2ovf_cmp1_event 101: t2ovf_cmp2_event 110: Reserved 111: No event
3	–	0	R0	Reserved. Read as 0
2:0	TIMER2_EVENT1_CFG	000	R/W	Selects the event that triggers a T2_EVENT1 pulse 000: t2_per_event 001: t2_cmp1_event 010: t2_cmp2_event 011: t2ovf_per_event 100: t2ovf_cmp1_event 101: t2ovf_cmp2_event 110: Reserved 111: No event

**T2EVTCFG (0x9C) – Timer 2 Event Configuration (CC2541)**

Bit No.	Name	Reset	R/W	Function
7:4	TIMER2_EVENT2_CFG	0000	R/W	Selects the event that triggers a T2_EVENT2 pulse 0000: t2_per_event 0001: t2_cmp1_event 0010: t2_cmp2_event 0011: t2ovf_per_event 0100: t2ovf_cmp1_event 0101: t2ovf_cmp2_event 0110: Reserved 0111: No event 1000: t2ovf_long_cmp1_event 1001: t2ovf_long_cmp2_event 1010–1110: Reserved 1111: No event
3:0	TIMER2_EVENT1_CFG	0000	R/W	Selects the event that triggers a T2_EVENT1 pulse 0000: t2_per_event 0001: t2_cmp1_event 0010: t2_cmp2_event 0011: t2ovf_per_event 0100: t2ovf_cmp1_event 0101: t2ovf_cmp2_event 0110: Reserved 0111: No event 1000: t2ovf_long_cmp1_event 1001: t2ovf_long_cmp2_event 1010–1110: Reserved 1111: No event

## CC253x Radio

The **RF Core** controls the analog radio modules. In addition, it provides an interface between the MCU and the radio which makes it possible to issue commands, read status, and automate and sequence radio events.

Topic	Page
23.1 RF Core .....	209
23.2 FIFO Access .....	213
23.3 DMA .....	213
23.4 Memory Map .....	213
23.5 Frequency and Channel Programming .....	215
23.6 IEEE 802.15.4-2006 Modulation Format.....	215
23.7 IEEE 802.15.4-2006 Frame Format.....	217
23.8 Transmit Mode .....	218
23.9 Receive Mode .....	222
23.10 RXFIFO Access .....	232
23.11 Radio-Control State Machine .....	234
23.12 Random-Number Generation .....	236
23.13 Packet Sniffing and Radio Test Output Signals .....	237
23.14 Command Strobe Processor.....	238
23.15 Registers.....	255



## 23.1 RF Core

The **RF Core** controls the analog radio modules. In addition, it provides an interface between the MCU and the radio which makes it possible to issue commands, read status, and automate and sequence radio events.

The **FSM** submodule controls the RF transceiver state, the transmitter and receiver FIFOs, and most of the dynamically controlled analog signals, such as power up and power down of analog modules. The FSM is used to provide the correct sequencing of events (such as performing an FS calibration before enabling the receiver/transmitter). Also, it provides step-by-step processing of incoming frames from the demodulator: reading the frame length, counting the number of bytes received, checking the FCS, and finally, optionally handling automatic transmission of ACK frames after successful frame reception. It performs similar tasks in TX, including performing an optional CCA before transmission and automatically going to RX after the end of transmission to receive an ACK frame. Finally, the FSM controls the transfer of data between modulator or demodulator and the TXFIFO or RXFIFO in RAM.

The **modulator** transforms raw data into I/Q signals to the transmitter DAC. This is done in compliance with the IEEE 802.15.4 standard.

The **demodulator** is responsible for retrieving the over-the-air data from the received signal.

The amplitude information from the demodulator is used by the **automatic gain control (AGC)**. The AGC adjusts the gain of the analog LNA so that the signal level within the receiver is approximately constant.

The **frame filtering and source matching** supports the FSM in the RF Core by performing all operations needed in order to do frame filtering and source address matching, as defined by IEEE 802.15.4.

The **frequency synthesizer (FS)** generates the carrier wave for the RF signal.

The **command strobe processor (CSP)** processes all commands issued by the CPU. It also has a short program memory of 24 bytes, making it possible to automate CSP algorithms.

The **radio RAM** holds a FIFO for transmit data (TXFIFO) and a FIFO for receive data (RXFIFO). Both FIFOs are 128 bytes long. In addition, the RAM holds parameters for frame filtering and source matching, and for which 128 bytes are reserved.

**Timer 2 (MAC Timer)** is used for timing of radio events and to capture time stamps of incoming packets. This timer keeps counting even in power modes PM1 and PM2.

### 23.1.1 Interrupts

The radio is associated with two **interrupt** vectors on the CPU. These are the RFERR interrupt (interrupt 0) and the RF interrupt (interrupt 12) with the following functions.

- RFERR: Error situations in the radio are signaled using this interrupt.
- RF: Interrupts coming from normal operation are signaled using this interrupt.

The RF interrupt vector combines the interrupts in RFIF. Note that these RF interrupts are rising-edge triggered. Thus, an interrupt is generated when, for example, the SFD status flag goes from 0 to 1. The RFIF interrupt flags are described in [Section 23.1.2](#).

### 23.1.2 Interrupt Registers

Two of the main interrupt control SFR registers are used to enable the RF and RFERR interrupts. These are the following:

- RFERR: `IEN0.RFERRIE`
- RF: `IEN2.RFIE`

Two main interrupt flag SFR registers hold the RF and RFERR interrupt flags. These are the following:

- RFERR: `TCON.RFERRIF`
- RF: `S1CON.RFIF`

The two interrupts generated by the RF Core are a combination of several sources within the RF Core. Each of the individual sources has its own enable and interrupt flags in the RF Core. Flags can be found in `RFIRQF0`, `RFIRQF1`, and `RFIERRF`. Interrupt masks can be found in `RFIRQM0`, `RFIRQM1`, and `RFERRM`.

The interrupt-enable bits in the mask registers are used to enable individual interrupt sources for the two RF interrupts. Note that masking an interrupt source does not affect the updating of the status in the flag registers.

Due to the use of individual interrupt masks in the RF Core, the interrupts coming from RF Core have two-layered masking, and care must be taken when processing these interrupts. The procedure is described as follows.

To clear an interrupt from the RF Core, one must clear two flags, both the flag set in RF Core and the one set in `S1CON` or `TCON` (depending on which interrupt is triggered). If a flag is cleared in the RF Core and there are other unmasked flags standing, another interrupt is generated.

**RFIRQ0 (0xE9) RF Interrupt Flags**

Bit	Name	Reset	R/W	Description
7	RXMASKZERO	0	R/W0	The <code>RXENABLE</code> register has gone from a nonzero state to an all-zero state. 0: No interrupt pending 1: Interrupt pending
6	RXPKTDONE	0	R/W0	A complete frame has been received. 0: No interrupt pending 1: Interrupt pending
5	FRAME_ACCEPTED	0	R/W0	Frame has passed frame filtering. 0: No interrupt pending 1: Interrupt pending
4	SRC_MATCH_FOUND	0	R/W0	Source match found 0: No interrupt pending 1: Interrupt pending
3	SRC_MATCH_DONE	0	R/W0	Source matching complete 0: No interrupt pending 1: Interrupt pending
2	FIFOP	0	R/W0	The number of bytes in the <code>RXFIFO</code> is above the threshold. Also raised when a complete frame has been received, and when a packet has been read out completely and there are more complete packets available. 0: No interrupt pending 1: Interrupt pending
1	SFD	0	R/W0	SFD has been received or transmitted. 0: No interrupt pending 1: Interrupt pending
0	ACT_UNUSED	0	R/W0	Reserved 0: No interrupt pending 1: Interrupt pending

**RFIRQF1 (0x91) RF Interrupt Flags**

Bit	Name	Reset	R/W	Description
7:6	–	00	R0	Reserved. Read as 0
5	CSP_WAIT	0	R/W0	Execution continued after a wait instruction in CSP. 0: No interrupt pending 1: Interrupt pending
4	CSP_STOP	0	R/W0	CSP has stopped program execution. 0: No interrupt pending 1: Interrupt pending
3	CSP_MANINT	0	R/W0	Manual interrupt generated from CSP 0: No interrupt pending 1: Interrupt pending
2	RFIDLE	0	R/W0	Radio state machine has entered the idle state. 0: No interrupt pending 1: Interrupt pending
1	TXDONE	0	R/W0	A complete frame has been transmitted. 0: No interrupt pending 1: Interrupt pending
0	TXACKDONE	0	R/W0	An acknowledgment frame has been completely transmitted. 0: No interrupt pending 1: Interrupt pending

**RFERRF (0xBF) – RF Error Interrupt Flags**

Bit	Name	Reset	R/W	Description
7	–	0	R0	Reserved. Read as 0
6	STROBEERR	0	R/W0	A command strobe was issued at a time it could not be processed. Triggered if trying to disable radio when already disabled, or when trying to do a SACK, SACKPEND, or SNACK command when not in active RX. 0: No interrupt pending 1: Interrupt pending
5	TXUNDERF	0	R/W0	TXFIFO underflowed 0: No interrupt pending 1: Interrupt pending
4	TXOVERF	0	R/W0	TXFIFO overflowed 0: No interrupt pending 1: Interrupt pending
3	RXUNDERF	0	R/W0	RXFIFO underflowed 0: No interrupt pending 1: Interrupt pending
2	RXOVERF	0	R/W0	RXFIFO overflowed 0: No interrupt pending 1: Interrupt pending
1	RXABO	0	R/W0	Reception of a frame was aborted. 0: No interrupt pending 1: Interrupt pending
0	NLOCK	0	R/W0	Frequency synthesizer failed to achieve lock after time-out, or lock is lost during reception. Receiver must be restarted to clear this error situation. 0: No interrupt pending 1: Interrupt pending

**RFIRQM0 (0x61A3) – RF Interrupt Masks**

Bit	Name	Reset	R/W	Description
7	RXMASKZERO	0	R/W	The RXENABLE register has gone from a nonzero state to an all-zero state. 0: Interrupt disabled 1: Interrupt enabled
6	RXPKTDONE	0	R/W	A complete frame has been received. 0: Interrupt disabled 1: Interrupt enabled
5	FRAME_ACCEPTED	0	R/W	Frame has passed frame filtering. 0: Interrupt disabled 1: Interrupt enabled
4	SRC_MATCH_FOUND	0	R/W	Source match found 0: Interrupt disabled 1: Interrupt enabled
3	SRC_MATCH_DONE	0	R/W	Source matching complete 0: Interrupt disabled 1: Interrupt enabled
2	FIFOP	0	R/W	The number of bytes in the RXFIFO is above the threshold. Also raised when a complete frame has been received, or when a complete packet has been read out and there are more complete packets available. 0: Interrupt disabled 1: Interrupt enabled
1	SFD	0	R/W	SFD has been received or transmitted. 0: Interrupt disabled 1: Interrupt enabled
0	ACT_UNUSED	0	R/W	Reserved 0: Interrupt disabled 1: Interrupt enabled

**RFIRQM1 (0x61A4) – RF Interrupt Masks**

Bit	Name	Reset	R/W	Description
7:6	–	00	R0	Reserved. Read as 0
5	CSP_WAIT	0	R/W	Execution continued after a wait instruction in CSP. 0: Interrupt disabled 1: Interrupt enabled
4	CSP_STOP	0	R/W	CSP has stopped program execution. 0: Interrupt disabled 1: Interrupt enabled
3	CSP_MANINT	0	R/W	Manual interrupt generated from CSP. 0: Interrupt disabled 1: Interrupt enabled
2	RFIDLE	0	R/W	Radio state machine has entered the idle state. 0: Interrupt disabled 1: Interrupt enabled
1	TXDONE	0	R/W	A complete frame has been transmitted. 0: Interrupt disabled 1: Interrupt enabled
0	TXACKDONE	0	R/W	An acknowledgment frame has been completely transmitted. 0: Interrupt disabled 1: Interrupt enabled

**RFERRM (0x61A5) – RF Error Interrupt Masks**

Bit	Name	Reset	R/W	Description
7	–	0	R0	Reserved. Read as 0
7:6	STROBEERR	0	R/W	A command strobe was issued at a time it could not be processed. Triggered if trying to disable radio when already disabled, or when trying to do a SACK, SACKPEND, or SNACK command when not in active RX. 0: Interrupt disabled 1: Interrupt enabled
5	TXUNDERF	0	R/W	TXFIFO underflowed 0: Interrupt disabled 1: Interrupt enabled
4	TXOVERF	0	R/W	TXFIFO overflowed 0: Interrupt disabled 1: Interrupt enabled
3	RXUNDERF	0	R/W	RXFIFO underflowed 0: Interrupt disabled 1: Interrupt enabled
2	RXOVERF	0	R/W	RXFIFO overflowed 0: Interrupt disabled 1: Interrupt enabled
1	RXABO	0	R/W	Reception of a frame was aborted. 0: Interrupt disabled 1: Interrupt enabled
0	NLOCK	0	R/W	Frequency synthesizer failed to achieve lock after time-out, or lock is lost during reception. Receiver must be restarted to clear this error situation. 0: Interrupt disabled 1: Interrupt enabled

**23.2 FIFO Access**

The TXFIFO and RXFIFO may be accessed through the SFR register `RFD` (0xD9). Data is written to the TXFIFO when writing to the `RFD` register. Data is read from the RXFIFO when the `RFD` register is read.

The XREG registers `RXFIFOCNT` and `TXFIFOCNT` provide information on the amount of data in the FIFOs. The FIFO contents can be cleared by issuing `SFLUSHRX` and `SFLUSHTX`.

**RFD (0xD9) – RF Data**

Bit	Name	Reset	R/W	Description
7:0	RFD[7:0]	0x00	R/W	Data written to the register is written to the TXFIFO. When reading this register, data from the RXFIFO is read.

**23.3 DMA**

It is possible to use direct memory access (DMA) to move data between memory and the radio. The DMA controller is described in [Chapter 8](#). See this section for a detailed description on how to set up and use DMA transfers.

To support the DMA controller, there is one DMA trigger associated with the radio, the RADIO DMA trigger (DMA trigger 19). The RADIO DMA trigger is activated by two events. The first event to cause a RADIO DMA trigger is when the first data is present in the RXFIFO, that is, when the RXFIFO goes from the empty state to a nonempty state. The second event that causes a RADIO DMA trigger is when data is read from the RXFIFO (through `RFD`) and there is still more data available in the RXFIFO.

**23.4 Memory Map**

The RF Core contains 384 bytes of physical RAM located at addresses 0x6000 to 0x0617F. The configuration and status registers of the RF Core are located at addresses from 0x6180 to 0x61EF. Configuration registers, RXFIFO, and TXFIFO are all preserved during sleep modes.

### 23.4.1 RXFIFO

The RXFIFO memory area is located at addresses 0x6000 to 0x607F and is thus 128 bytes. Although this memory area is intended for the RXFIFO, it is not protected in any way, so it is still accessible in the XREG memory space. Normally, only the designated instructions should be used to manipulate the contents of the RXFIFO. The RXFIFO can contain more than one frame at a time.

### 23.4.2 TXFIFO

The TXFIFO memory area is located at addresses 0x6080 to 0x60FF and is thus 128 bytes. Although this memory area is intended for the TXFIFO, it is not protected in any way, so it is still accessible in the XREG memory space. Normally, only the designated instructions should be used to manipulate the contents of the TXFIFO. The TXFIFO can only contain one frame at a time.

### 23.4.3 Frame-Filtering and Source-Matching Memory Map

The frame-filtering and source-address-matching functions use a 128 byte block of the RF Core RAM to store local-address information and source-matching configuration and results; this is located in the area 0x6100 to 0x617F. This memory space is described in [Table 23-1](#). Values that do not fill an entire byte or word are in the least-significant part of the byte or word. Note that the values in these registers are unknown after reset. However, the values are retained during power modes.

**Table 23-1. Frame Filtering and Source Matching Memory Map**

ADDRESS	REGISTER/VARIABLE	ENDIAN	DESCRIPTION
<b>RESERVED</b>			
0x6176–0x617F	Temporary storage		Memory space used for temporary storage of variables
<b>LOCAL ADDRESS INFORMATION</b>			
0x6174–0x6175	SHORT_ADDR	LE	The short address used during destination address filtering
0x6172–0x6173	PAN_ID	LE	The PAN ID used during destination address filtering
0x616A–0x6171	EXT_ADD	LE	The IEEE extended address used during destination address filtering
<b>SOURCE ADDRESS MATCHING CONTROL</b>			
0x6169	SRCSHORTPENDEN2		8 MSBs of the 24-bit mask that enables and disables automatic pending for each of the 24 short addresses
0x6168	SRCSHORTPENDEN1		8 middle bits of the 24-bit mask that enables and disables automatic pending for each of the 24 short addresses
0x6167	SRCSHORTPENDEN0		8 LSBs of the 24-bit mask that enables and disables automatic pending for each of the 24 short addresses
0x6166	SRCEXTPENDEN2		8 MSBs of the 24-bit mask that enables and disables automatic pending for each of the 12 extended addresses. Entry n is mapped to SRCEXTPENDEN[2n]. All SRCEXTPENDEN[2n + 1] bits are don't care.
0x6165	SRCEXTPENDEN1		8 middle bits of the 24-bit mask that enables and disables automatic pending for each of the 12 extended addresses. Entry n is mapped to SRCEXTPENDEN[2n]. All SRCEXTPENDEN[2n + 1] bits are don't care.
0x6164	SRCEXTPENDEN0		8 LSBs of the 24-bit mask that enables and disables automatic pending for each of the 12 extended addresses. Entry n is mapped to SRCEXTPENDEN[2n]. All SRCEXTPENDEN[2n + 1] bits are don't care.
<b>SOURCE ADDRESS MATCHING RESULT</b>			
0x6163	SRCRESINDEX		The bit index of the least-significant 1 in SRCRESMASK, or 0x3F when there is no source match. On a match, bit 5 is 0 when the match is on a short address and 1 when it is on an extended address. On a match, bit 6 is 1 when the conditions for automatic pending bit in acknowledgment have been met (see the description of SRCMATCH.AUTOPEND). The bit gives no indication of whether or not the acknowledgment actually is transmitted, and does not take the PENDING_OR register bit and the SACK, SACKPEND, and SNACK strobes into account.

**Table 23-1. Frame Filtering and Source Matching Memory Map (continued)**

ADDRESS	REGISTER/VARIABLE	ENDIAN	DESCRIPTION		
0x6162	SRCRESMASK2		24-bit mask that indicates source address match for each individual entry in the source address table		
0x6161	SRCRESMASK1		Short address matching. When there is a match on entry panid_n + short_n, bit n is set in SRCRESMASK.		
0x6160	SRCRESMASK0		Extended address matching. When there is a match on entry ext_n, bits 2n and 2n + 1 are set in SRCRESMASK.		
SOURCE ADDRESS TABLE					
0x615E–0x615F	short_23	ext_11	LE	LE	Two individual short-address entries (combination of 16-bit PAN ID and 16-bit short address) or one extended address entry
0x615C–0x615D	panid_23		LE		
0x615A–0x615B	short_22		LE		
0x6158–0x6159	panid_22		LE		
...	...	...	...	...	...
0x610E–0x610F	short_03	ext_01	LE	LE	Two individual short address entries (combination of 16-bit PAN ID and 16-bit short address) or one extended address entry
0x610C–0x610D	panid_03		LE		
0x610A–0x610B	short_02		LE		
0x6108–0x6109	panid_02		LE		
0x6106–0x6107	short_01	ext_00	LE	LE	Two individual short address entries (combination of 16-bit PAN ID and 16-bit short address) or one extended address entry
0x6104–0x6105	panid_01		LE		
0x6102–0x6103	short_00		LE		
0x6100–0x6101	panid_00		LE		

## 23.5 Frequency and Channel Programming

The carrier frequency is set by programming the 7-bit frequency word located in `FREQCTRL.FREQ[6:0]`. Changes take effect after the next recalibration. Carrier frequencies in the range from 2394 MHz to 2507 MHz are supported. The carrier frequency  $f_c$ , in MHz, is given by  $f_c = (2394 + \text{FREQCTRL.FREQ}[6:0])$  MHz, and is programmable in 1-MHz steps.

IEEE 802.15.4-2006 specifies 16 channels within the 2.4-GHz band. They are numbered 11 through 26 and are 5 MHz apart. The RF frequency of channel k is given by [Equation 4](#).

$$f_c = 2405 + 5(k - 11) \quad [\text{MHz}] \quad k \in [11, 26] \quad (4)$$

For operation in channel k, the `FREQCTRL.FREQ` register should therefore be set to `FREQCTRL.FREQ = 11 + 5(k - 11)`.

## 23.6 IEEE 802.15.4-2006 Modulation Format

This section is meant as an introduction to the 2.4-GHz direct-sequence spread-spectrum (DSSS) RF modulation format defined in IEEE 802.15.4-2006. For a complete description, see the standard document [1].

The modulation and spreading functions are illustrated at the block level in [Figure 23-1](#). Each byte is divided into two symbols, 4 bits each. The least-significant symbol is transmitted first. For multibyte fields, the least-significant byte is transmitted first, except for security-related fields, where the most-significant byte is transmitted first.

Each symbol is mapped to one out of 16 pseudorandom sequences, 32 chips each. The symbol-to-chip mapping is shown in [Table 23-2](#). The chip sequence is then transmitted at 2 Mc/s, with the least-significant chip ( $C_0$ ) transmitted first for each symbol. The transmitted bit stream and the chip sequences are observable on GPIO pins P1[0:5]. See [Chapter 7](#) for details on how to configure the GPIO to do this.

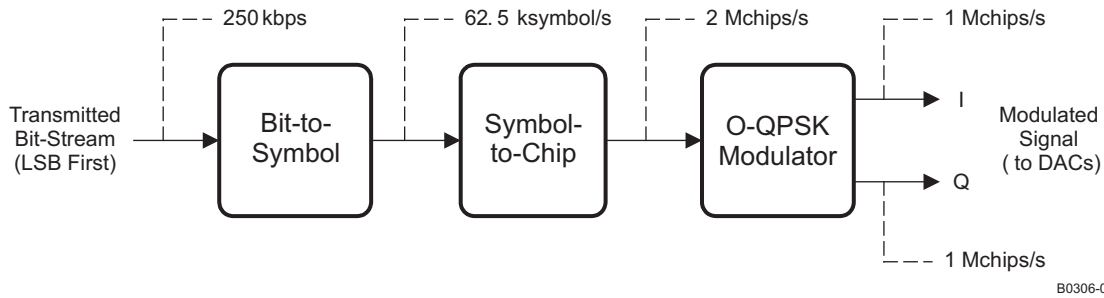


Figure 23-1. Modulation

Table 23-2. IEEE 802.15.4-2006 Symbol-to-Chip Mapping

Symbol	Chip Sequence (C0, C1, C2, ... , C31)
0	1 1 0 1 1 0 0 1 1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0 1 1 1 0
1	1 1 1 0 1 1 0 1 1 0 0 1 1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0
2	0 0 1 0 1 1 1 0 1 1 0 1 1 0 0 1 1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 0
3	0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1 1 0 0 1 1 1 0 0 0 0 1 1 0 1 0 1
4	0 1 0 1 0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1 1 0 0 1 1 1 0 0 0 0 1 1
5	0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1 1 0 0 1 1 1 0 0
6	1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1 1 0 0 1
7	1 0 0 1 1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1
8	1 0 0 0 1 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1
9	1 0 1 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 1 1 1 0 1 1 1
10	0 1 1 1 1 0 1 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 1 1 1
11	0 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0
12	0 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 1 0
13	0 1 1 0 0 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0 1 1 0 0 1 0 0 1
14	1 0 0 1 0 1 1 0 0 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0 1 1 0 0
15	1 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0

The modulation format is offset – quadrature phase shift keying (O-QPSK) with half-sine chip shaping. This is equivalent to MSK modulation. Each chip is shaped as a half-sine, transmitted alternately in the I and Q channels with one-half chip-period offset. This is illustrated for the zero-symbol in Figure 23-2.

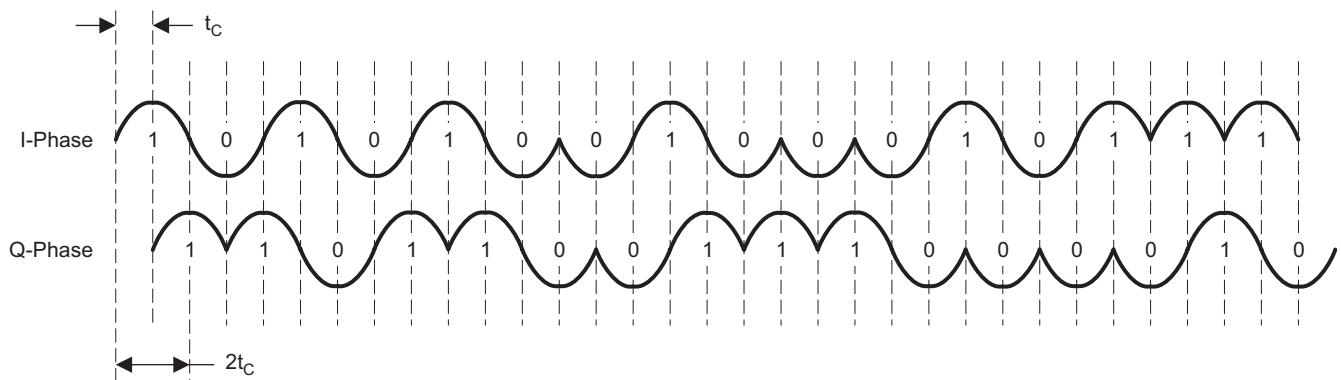


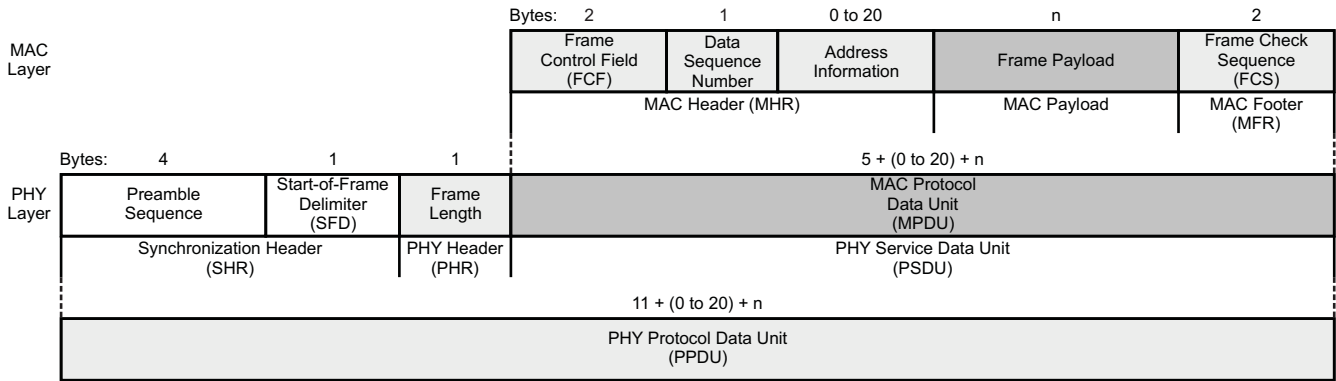
Figure 23-2. I/Q Phases When Transmitting a Zero-Symbol Chip Sequence,  $t_c = 0.5 \mu s$



### 23.7 IEEE 802.15.4-2006 Frame Format

This section gives a brief summary of the IEEE 802.15.4 frame format [1]. The radio has built-in support for processing of parts of the frame. This is described in the following sections.

Figure 23-3 shows a schematic view of the IEEE 802.15.4 frame format. Similar figures describing specific frame formats (data frames, beacon frames, acknowledgment frames, and MAC command frames) are included in the standard document [1].



M0108-01

Figure 23-3. Schematic View of the IEEE 802.15.4 Frame Format [1]

#### 23.7.1 PHY Layer

##### Synchronization Header

The synchronization header (SHR) consists of the preamble sequence followed by the start-of-frame delimiter (SFD). In the IEEE 802.15.4 specification [1], the preamble sequence is defined to be 4 bytes of 0x00. The SFD is one byte with value 0xA7.

##### PHY Header

The PHY header consists only of the frame-length field. The frame-length field defines the number of bytes in the MPDU. Note that the value of the frame-length field does not include the frame-length field itself. It does, however, include the frame-check sequence (FCS), even if this is inserted automatically by the hardware.

The frame-length field is 7 bits long and has a maximum value of 127. The most-significant bit in the frame-length field is reserved, and should always be set to zero.

##### PHY Service Data Unit

The PHY service data unit (PSDU) contains the MAC protocol data unit (MPDU). The function of the MAC layer is to generate or interpret the MPDU, and the radio has built-in support for processing of some of the MPDU subfields.

#### 23.7.2 MAC Layer

The FCF, data sequence number, and address information follow the frame-length field as shown in Figure 23-3. Together with the MAC data payload and frame check sequence, they form the MPDU. The format of the FCF is shown in Figure 23-4. For full details, see the IEEE 802.15.4 specification [1].

Bits: 0–2	3	4	5	6	7–9	10–11	12–13	14–15
Frame type	Security enabled	Frame pending	Acknowledge request	Intra PAN	Reserved	Destination addressing mode	Reserved	Source addressing mode

Figure 23-4. Format of the Frame Control Field (FCF)

### Frame-Check Sequence

A 2-byte frame-check sequence (FCS) follows the last MAC payload byte as shown in [Figure 23-3](#). The FCS is calculated over the MPDU, that is, the frame-length field is not part of the FCS.

The FCS polynomial defined in [1] is

$$G(s) = x^{16} + x^{12} + x^5 + 1$$

The radio supports automatic calculation and verification of the FCS. See [Section 23.8.10](#) for details.

## 23.8 Transmit Mode

This section describes how to control the transmitter, how to control the integrated frame processing, and how to use the TXFIFO.

### 23.8.1 TX Control

The radio has many built-in features for frame processing and status reporting. Note that the radio provides features that make it easy to have precise control of the timing of outgoing frames. This is very important in an IEEE 802.15.4/ZigBee® system, because there are strict timing requirements to such systems.

Frame transmission is started by the following actions:

- The STXON command strobe
  - The `SAMPLED_CCA` signal is not updated.
- The STXONCCA command strobe, provided that the CCA signal is high.
  - Aborts ongoing transmission or reception and forces a TX calibration followed by transmission.
  - The `SAMPLED_CCA` signal is updated.

Clear-channel assessment is described in detail in [Section 23.8.12](#).

Frame transmission is aborted by the following command actions:

- The SRXON command strobe
  - Aborts ongoing transmission and forces an RX calibration
- The SRFOFF command strobe
  - Aborts ongoing transmission or reception and forces the FSM to the IDLE state
- The STXON command strobe
  - Aborts ongoing transmission and forces an RX calibration

To enable the receiver after transmission with STXON, the `FRMCTRL1.SET_RXENMASK_ON_TX` bit should be set. This sets bit 6 in `RXENABLE` when STXON is executed. When transmitting with STXONCCA, the receiver is on before the transmission and is turned back on afterwards (unless the `RXENABLE` registers have been cleared in the meantime).

### 23.8.2 TX State Timing

Transmission of preamble begins 192  $\mu$ s after the STXON or STXONCCA command strobe. This is referred to as *TX turnaround time* in [1]. There is an equal delay when returning to receive mode.

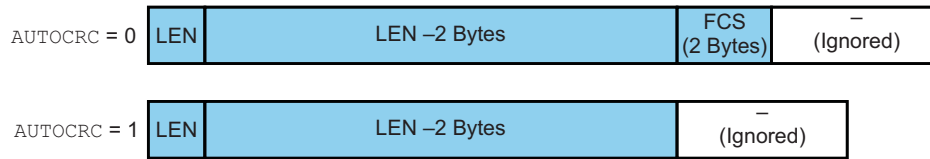
When returning to idle or receive mode, there is a 2- $\mu$ s delay while the modulator ramps down the signals to the DACs. The down ramping happens automatically after the complete MPDU (as defined by the length byte) has been transmitted or if TX underflow occurs. This affects:

- The SFD signal, which is stretched by 2  $\mu$ s.
- The radio FSM transition to the IDLE state, which is delayed by 2  $\mu$ s.

### 23.8.3 TXFIFO Access

The TXFIFO can hold 128 bytes and only one frame at a time. The frame can be buffered before or after the TX command strobe is executed, as long as it does not generate TX underflow (see the error conditions listed in [Section 23.8.5](#)).

Figure 23-5 illustrates what must be written to the TXFIFO (marked blue). Additional bytes are ignored, unless TX overflow occurs (see the error conditions listed in Section 23.8.5).



M0109-01

Figure 23-5. Frame Data Written to the TXFIFO

There are two ways to write to the TXFIFO.

- Write to the RFD register.
- Frame buffering always begins at the start of the TXFIFO memory. By enabling the `FRMCTRL1.IGNORE_TX_UNDERF` bit, it is possible to write directly into the RAM area in the radio memory, which holds the TXFIFO. Note that it is recommended to use the RFD register for writing data to the TXFIFO.

The number of bytes in the TXFIFO is stored in the `TXFIFOCNT` register.

The TXFIFO can be emptied manually with the `SFLUSHTX` command strobe. TX underflow occurs if the FIFO is emptied during transmission.

### 23.8.4 Retransmission

In order to support simple retransmission of frames, the radio does not delete the TXFIFO contents as they are transmitted. After a frame has been successfully transmitted, the FIFO contents are left unchanged. To retransmit the same frame, simply restart TX by issuing an `STXON` or `STXONCCA` command strobe. Note that a retransmission of a packet is only possible if the packet has been completely transmitted; that is, a packet cannot be aborted and then be retransmitted.

If a different frame is to be transmitted, issue an `ISFLUSHTX` strobe and then write the new frame to the TXFIFO.

### 23.8.5 Error Conditions

There are two error conditions associated with the TXFIFO:

- Overflow happens when the TXFIFO is full and another byte write is attempted.
- Underflow happens when the TXFIFO is empty and the radio attempts to fetch another byte for transmission.

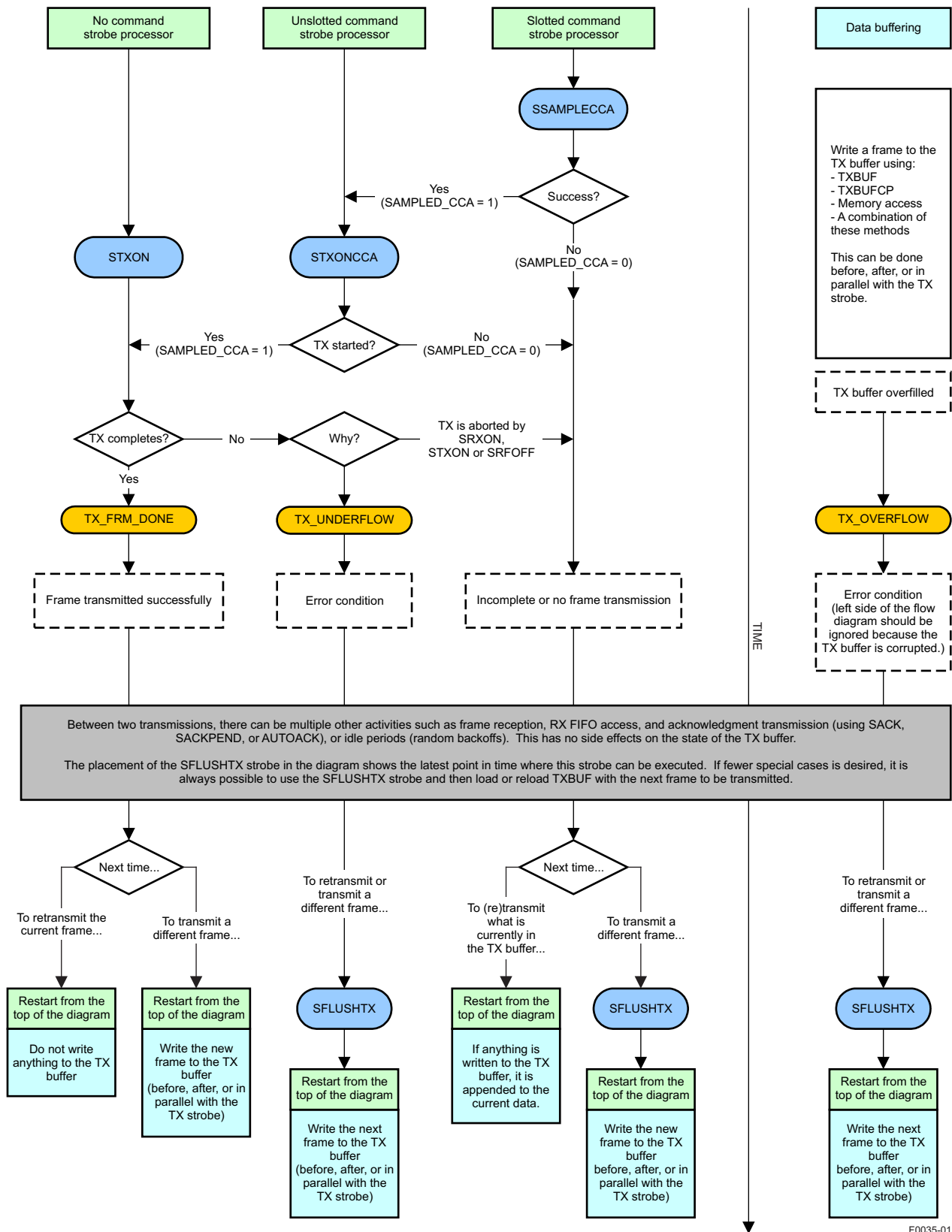
TX overflow is indicated by the `TX_OVERFLOW` interrupt flag being set. When this error occurs, the writing is aborted, that is, the data byte that caused the overflow is lost. The error condition must be cleared with the `SFLUSHTX` strobe.

TX underflow is indicated by the `TX_UNDERFLOW` interrupt flag being set. When this error occurs, the ongoing transmission is aborted. The error condition must be cleared with the `SFLUSHTX` strobe.

The `TX_UNDERFLOW` exception can be disabled by setting the `FRMCTRL1.IGNORE_TX_UNDERF` bit. In this case, the radio continues transmitting the bytes that happen to be in the TXFIFO memory, until the number of bytes given by the first byte (that is, the length byte) have been transmitted.

### 23.8.6 TX Flow Diagram

Figure 23-6 summarizes the previous sections in a flow diagram:

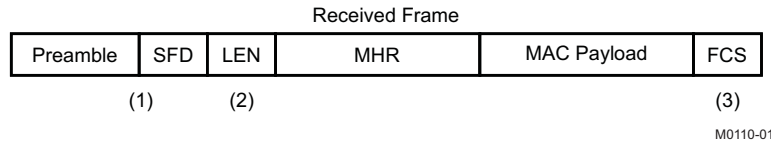


F0035-01

Figure 23-6. TX Flow

### 23.8.7 Transmitted Frame Processing

The radio performs the following frame generation tasks for TX frames:

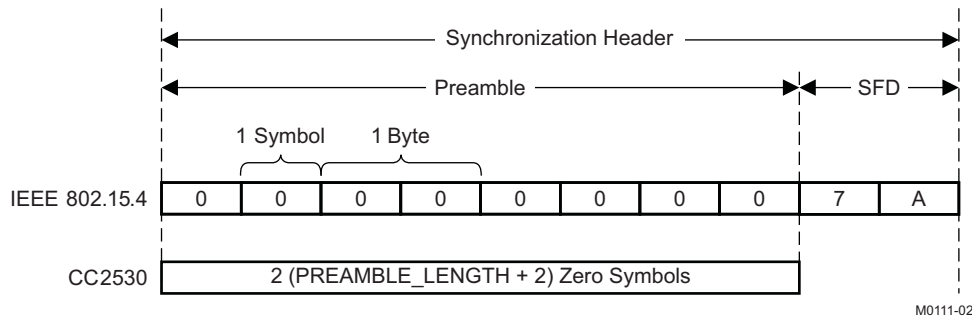


- (1) Generation and automatic transmission of the PHY layer synchronization header, which consists of the preamble and the SFD
- (2) Transmission of the number of bytes specified by the frame-length field
- (3) Calculation of and automatic transmission of the FCS (can be disabled)

**Figure 23-7. Single Transmitted Frame**

The recommended usage is to write the frame-length field followed by the MAC header and MAC payload to the TXFIFO and let the radio handle the rest. Note that the frame-length field must include the two FCS bytes, even though the radio adds these automatically.

### 23.8.8 Synchronization Header



**Figure 23-8. Transmitted Synchronization Header**

The radio has programmable preamble length. The default value is compliant with [1], and changing the value makes the system noncompliant to IEEE 802.15.4.

The preamble sequence length is set by `MDMCTRL0.PREAMBLE_LENGTH`. Figure 23-8 shows how the synchronization header relates to the IEEE 802.15.4 specification.

When the required number of preamble bytes has been transmitted, the radio automatically transmits the 1-byte SFD. The SFD is fixed, and it is not possible to change this value from software.

### 23.8.9 Frame-Length Field

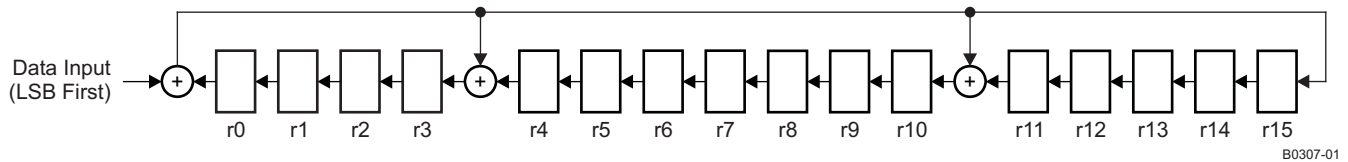
When the SFD has been transmitted, the modulator starts to read data from the TXFIFO. It expects to find the frame-length field followed by the MAC header and MAC payload. The frame-length field is used to determine how many bytes are to be transmitted.

Note that the minimum frame length is 3 bytes when `AUTOCRC = 1` and 1 byte when `AUTOCRC = 0`.

### 23.8.10 Frame Check Sequence

When the `FRMCTRL0.AUTOCRC` control bit is set, the FCS field is automatically generated and appended to the transmitted frame at the position defined by the frame-length field. The FCS is not written to the TXFIFO, but stored in a separate 16-bit register. It is recommended always to have `AUTOCRC` enabled, except possibly for debug purposes. If `FRMCTRL0.AUTOCRC = 0`, then the modulator expects to find the FCS in the TXFIFO, so software must generate the FCS and write it to the TXFIFO along with the rest of the MPDU.

The hardware implementation of the FCS calculator is shown in Figure 23-9. See [1] for further details.


**Figure 23-9. FCS Hardware Implementation**

### 23.8.11 Interrupts

The SFD interrupt is raised when the SFD field of the frame has been transmitted. At the end of the frame, the TX\_FRM\_DONE interrupt is raised when the complete frame has been successfully transmitted.

Note that there is a second SFD signal available on GPIO (through radio observation mux) that should not be confused with the SFD interrupt.

### 23.8.12 Clear-Channel Assessment

The clear-channel assessment (CCA) status signal indicates whether the channel is available for transmission or not. The CCA function is used to implement the command-strobe-processor functionality specified in the IEEE 802.15.4 specification [1]. The CCA signal is valid when the receiver has been enabled for at least eight symbol periods. The `RSSI_VALID` status signal can be used to verify this.

The CCA is based on the RSSI value and a programmable threshold. The exact behavior is configurable in the `CCCTRL0` and `CCCTRL1` registers.

There are two variations of the CCA signal, one that is updated at every new RSSI sample and one that is only updated on `SSAMPLECCA/ISAMPLECCA` and `STXONCCA/ISTXONCCA` command strobes. They are both available in the `FSMSTAT1` register.

Note that the CCA signal is updated four clock cycles (system clock) after the `RSSI_VALID` signal has been set.

### 23.8.13 Output Power Programming

The RF output power is controlled by the 7-bit value in the `TXPOWER` register. The device data sheet ([Appendix C](#)) shows typical output power and current consumption for recommended settings when the center frequency is set to 2.440 GHz. Note that the recommended settings are only a small subset of all the possible register settings.

### 23.8.14 Tips and Tricks

- Note that there is no requirement to have the complete frame in the TXFIFO before starting a transmission. Bytes may be added to the TXFIFO during transmission.
- It is possible to transmit non-IEEE 802.15.4 compliant frames by setting `MDMTEST1.MODULATION_MODE = 1`.

## 23.9 Receive Mode

This section describes how to control the receiver, control the integrated RX frame processing, and how to use the RXFIFO.

### 23.9.1 RX Control

The receiver is turned on and off with the `SRXON` and `SRFOFF` command strobes, and with the `RXENABLE` registers. The command strobes provide a *hard* on-off mechanism, whereas `RXENABLE` manipulation provides a *soft* on-off mechanism.

The receiver is turned on by the following actions:

- The `SRXON` strobe:

- Sets `RXENABLE[ 7 ]`
- Aborts ongoing transmission or reception by forcing a transition to RX calibration.
- The STXON strobe, when `FRMCTRL1.SET_RXENMASK_ON_TX` is enabled:
  - Sets `RXENABLE[ 6 ]`
  - The receiver is enabled after transmission completes.
- Setting `RXENABLE != 0x00` by writing to `RXENMASKOR`:
  - Does not abort ongoing transmission or reception.

The receiver is turned off by the following actions:

- The SRFOFF strobe:
  - Clears `RXENABLE[ 7 : 0 ]`
  - Aborts ongoing transmission or reception by forcing the transition to IDLE mode.
- Setting `RXENABLE = 0x00` by writing to `RXENMASKAND`
  - Does not abort ongoing transmission or reception. Once the ongoing transmission or reception is finished, the radio returns to the IDLE state.

There are several ways to manipulate the `RXENABLE` registers:

- The `SRXMASKBITSET` and `SRXMASKBITCLR` strobes (affecting `RXENABLE[ 5 ]`)
- The `SRXON`, `SRFOFF` and `STXON` strobes, including the `FRMCTRL1.SET_RXMASK_ON_TX` setting

### 23.9.2 RX State Timing

The receiver is ready 192  $\mu$ s after RX has been enabled by one of the methods described in [Section 23.9.1](#). This is referred to as *RX turnaround time* in [1].

When returning to receive mode after frame reception, there is by default an interval of 192  $\mu$ s where SFD detection is disabled. This interval can be disabled by clearing `FSMCTRL.RX2RX_TIME_OFF`.

### 23.9.3 Received-Frame Processing

The radio integrates critical portions of the RX requirements in IEEE 802.15.4-2003 and -2006 in hardware. This reduces the CPU interruption rate, simplifies the software that handles frame reception, and provides the results with minimum latency.

During reception of a single frame, the following frame-processing steps are performed:



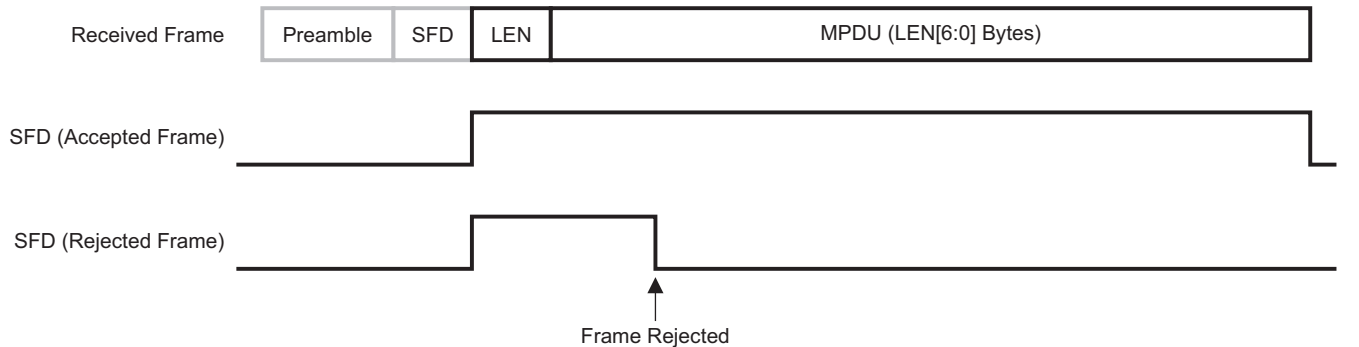
M0110-02

- (1) Detection and removal of the received PHY synchronization header (preamble and SFD), and reception of the number of bytes specified by the frame-length field.
- (2) Frame filtering as specified by [1], section 7.5.6.2, third filtering level.
- (3) Matching of the source address against a table containing up to 24 short addresses or 12 extended IEEE addresses. The source address table is stored in the radio RAM.
- (4) Automatic FCS checking, and attaching this result and other status values (RSSI, correlation, and source-match result) to received frames.
- (5) Automatic acknowledgment transmission with correct timing, and correct setting of the frame-pending bit, based on the results from source address matching and FCS checking.

**Figure 23-10. Single Received Frame and Transmitted Acknowledgment Frame**

### 23.9.4 Synchronization Header and Frame-Length Fields

Frame reception starts with detection of a start-of-frame delimiter (SFD), followed by the length byte, which determines when the reception is complete. The SFD signal, which can be output on GPIO, can be used to capture the start of received frames:



T0319-01

**Figure 23-11. SFD Signal Timing**

Preamble and SFD are not written to the RXFIFO.

The radio uses a correlator to detect the SFD. The correlation threshold value in `MDMCTRL1.CORR_THR` determines how closely the received SFD must match an *ideal* SFD. The threshold must be adjusted with care:

- If set too high, the radio misses many actual SFDs, effectively reducing the receiver sensitivity.
- If set too low, the radio detects many false SFDs. Although this does not reduce the receiver sensitivity, the effect is similar, because false frames might overlap with the SFDs of actual frames. It also increases the risk of receiving false frames with correct FCS.

In addition to SFD detection, it is also possible to require a number of valid preamble symbols (also above the correlation threshold) prior to SFD detection. See the register descriptions of `MDMCTRL0` and `MDMCTRL1` for available options and recommended settings.

### 23.9.5 Frame Filtering

The frame filtering function rejects nonintended frames as specified by [1], section 7.5.6.2, third filtering level. In addition, it provides filtering on:

- The eight different frame types (see the `FRMFILT1` register)
- The reserved bits in the frame control field (FCF)

The function is controlled by:

- The `FRMFILT0` and `FRMFILT1` registers
- The `PAN_ID`, `SHORT_ADDR` and `EXT_ADDR` values in RAM

#### Filtering Algorithm

The `FRMFILT0.FRAME_FILTER_EN` bit controls whether frame filtering is applied or not. When disabled, the radio accepts all received frames. When enabled (which is the default setting), the radio only accepts frames that fulfill all of the following requirements:

- The length byte must be equal to or higher than the minimum frame length, which is derived from the source- and destination-address mode and PAN ID compression subfields of the FCF.
- The reserved FCF bits [9:7] ANDed together with `FRMFILT0.FCF_RESERVED_BITMASK` must equal 000b.
- The value of the frame version subfield of the FCF cannot be higher than `FRMFILT0.MAX_FRAME_VERSION`.
- The source and destination address modes cannot be reserved values (1).
- Destination address:



- If a destination PAN ID is included in the frame, it must match PAN\_ID or must be the broadcast PAN identifier (0xFFFF).
- If a short destination address is included in the frame, it must match either SHORT\_ADDR or the broadcast address (0xFFFF).
- If an extended destination address is included in the frame, it must match EXT\_ADDR.
- Frame type:
  - Beacon frames (0) are only accepted when:
    - FRMFILT1.ACCEPT\_FT0\_BEACON = 1
    - Length byte  $\geq$  9
    - The destination address mode is 0 (no destination address).
    - The source address mode is 2 or 3 (that is, a source address is included).
    - The source PAN ID matches PAN\_ID, or PAN\_ID equals 0xFFFF.
  - Data (1) frames are only accepted when:
    - FRMFILT1.ACCEPT\_FT1\_DATA = 1
    - Length byte  $\geq$  9
    - A destination address and/or source address is included in the frame. If no destination address is included in the frame, the FRMFILT0.PAN\_COORDINATOR bit must be set, and the source PAN ID must equal PAN\_ID.
  - Acknowledgment (2) frames are only accepted when:
    - FRMFILT1.ACCEPT\_FT2\_ACK = 1
    - Length byte = 5
  - MAC command (3) frames are only accepted when:
    - FRMFILT1.ACCEPT\_FT3\_MAC\_CMD = 1
    - Length byte  $\geq$  9
    - A destination address and/or source address is included in the frame. If no destination address is included in the frame, the FRMFILT0.PAN\_COORDINATOR bit must be set, and the source PAN ID must equal PAN\_ID for the frame to be accepted.
  - Reserved frame types (4, 5, 6, and 7) are only accepted when
    - FRMFILT1.ACCEPT\_FT4TO7\_RESERVED = 1 (default is 0)
    - Length byte  $\geq$  9

The following operations are performed before the filtering begins, with no effect on the frame data stored in the RXFIFO:

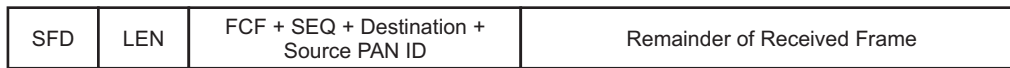
- Bit 7 of the length byte is masked out (don't care).
- If FRMFILT1.MODIFY\_FT\_FILTER is not equal to zero, the MSB of the frame-type subfield of the FCF is either inverted or forced to 0 or 1.

If a frame is rejected, the radio only starts searching for a new frame after the rejected frame has been completely received (as defined by the frame-length field) to avoid detecting false SFDs within the frame. Note that a rejected frame can generate RX overflow if it occurs before the frame is rejected.

### Interrupts

When frame filtering is enabled and the filtering algorithm accepts a received frame, an RX\_FRM\_ACCEPTED interrupt is generated. It is not generated if frame filtering is disabled or RX\_OVERFLOW or RX\_FRM\_ABORTED is generated before the filtering result is known.

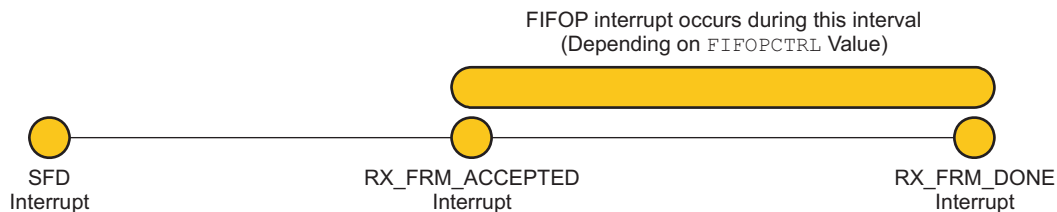
Figure 23-12 illustrates the three different scenarios (not including the overflow and abort-error conditions).



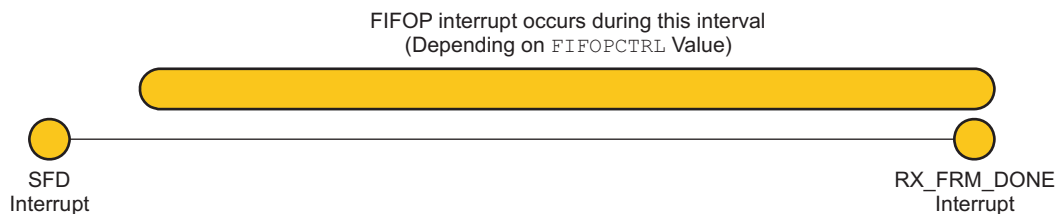
**Filtering is Enabled, Frame Rejected**



**Filtering is Enabled, Frame Accepted**



**Filtering is Disabled**



M0112-01

**Figure 23-12. Filtering Scenarios (Exceptions Generated During Reception)**

The `FSMSTAT1.SFD` register bit goes high when a start-of-frame delimiter is completely received and remains high until either the last byte in MPDU is received or the received frame has failed to pass address recognition and been rejected.

**Tips and Tricks**

The following register settings must be configured correctly:

- `FRMFILT0.PAN_COORDINATOR` must be set if the device is a PAN coordinator, and cleared if not.
- `FRMFILT0.MAX_FRAME_VERSION` must correspond to the supported version(s) of the IEEE 802.15.4 standard.
- The local address information must be loaded into RAM.

To avoid completely the receiving of frames during energy-detection scanning, set `FRMCTRL0.RX_MODE = 11b` and then (re)start RX. This disables symbol search and thereby prevents SFD detection.

To resume normal RX mode, set `FRMCTRL0.RX_MODE = 00b` and (re)start RX.

During operation in a busy IEEE 802.15.4 environment, the radio receives large numbers of nonintended acknowledgment frames. To block reception of these frames effectively, use the `FRMFILT1.ACCEPT_FT2_ACK` bit to control when acknowledgment frames should be received:

- Set `FRMFILT1.ACCEPT_FT2_ACK` after successfully starting a transmission with acknowledgment request, and clear the bit again after the acknowledgment frame has been received or the time-out has been reached.

- Keep the bit cleared otherwise.

It is not necessary to turn off the receiver while changing the values of the `FRMFILT0` and `FRMFILT1` registers and the local address information stored in RAM. However, if the changes take place between reception of the SFD byte and the source PAN ID (that is, between the *SFD* and *RX\_FRM\_ACCEPTED* exceptions), the modified values must be considered as don't care for that particular frame (the radio uses either the old or the new value).

Note that it is possible to make the radio ignore all IEEE 802.15.4 incoming frames by setting `MDMTEST1.MODULATION_MODE = 1`.

### 23.9.6 Source Address Matching

The radio supports matching of the source address in received frames against a table stored in the on-chip memory. The table is 96 bytes long, and hence it can contain up to:

- 24 short addresses (2-byte PAN ID + 2-byte short address).
- 12 IEEE extended addresses (8 bytes each).

Source address matching is only performed when frame filtering is also enabled and the received frame has been accepted. The function is controlled by:

- The `SRCMATCH`, `SRCSHORTEN0`, `SRCSHORTEN1`, `SRCSHORTEN2`, `SRCEXTEN0`, `SRCEXTEN1`, and `SRCEXTEN2` registers
- The source address table in RAM

#### Applications

*Automatic acknowledgment transmission with correct setting of the frame-pending bit:* When using indirect frame transmission, the devices send data requests to poll frames stored on the coordinator. To indicate whether it actually has a frame stored for the device, the coordinator must set or clear the frame-pending bit in the returned acknowledgment frame. On most 8- and 16-bit MCUs, however, there is not enough time to determine this, and so the coordinator ends up setting the pending bit regardless of whether there are pending frames for the device (as required by IEEE 802.15.4 [1]). This is wasteful in terms of power consumption, because the polling device must keep its receiver enabled for a considerable period of time, even if there are no frames for it. By loading the destination addresses in the indirect frame queue into the source address table and enabling the `AUTOPEND` function, the radio sets the pending bit in outgoing acknowledgment frames automatically. This way, the operation is no longer timing-critical, as the effort done by the microcontroller is when adding or removing frames in the indirect frame queue and updating the source address table accordingly.

*Security material look-up:* To reduce the time needed to process secured frames, the source address table can be set up so the entries match the table of security keys on the CPU. A second level of masking on the table entries allows this application to be combined with automatic setting of the pending bit in acknowledgment frames.

*Other applications:* The two previous applications are the main targets for the source-address matching function. However, for proprietary protocols that only rely on the basic IEEE 802.15.4 frame format, there are several other useful applications. For instance, it is possible to create firewall functionality where only a specified set of nodes is to be acknowledged.

#### The Source Address Table

The source address table begins at address `0x6100` in RAM. The space is shared between short and extended addresses, and the `SRCSHORTEN0`, `SRCSHORTEN1`, `SRCSHORTEN2` and `SRCEXTEN0`, `SRCEXTEN1`, `SRCEXTEN2` registers are used to control which entries are enabled. All values in the table are little-endian (as in the received frames).

- A *short address entry* starts with the 16-bit PAN ID followed by the 16-bit short address. These entries are stored at address `0x6100 + (4 × n)`, where `n` is a number between 0 and 23.
- An *extended address entry* consists only of the 64-bit IEEE extended address. These entries are stored at address `0x6100 + (8 × n)`, where `n` is a number between 0 and 11.

#### Address Enable Registers

Software is responsible for allocating table entries and for making sure that active short and extended address entries do not overlap. There are separate enable bits for short and extended addresses:

- Short address entries are enabled in the SRCSHORTEN0, SRCSHORTEN1, and SRCSHORTEN2 registers. Register bit *n* corresponds to short address entry *n*.
- Extended address entries are enabled in the SRCEXTEN0, SRCEXTEN1, and SRCEXTEN2 registers. In this case, register bit *2n* corresponds to extended address entry *n*. This mapping is convenient when creating a combined bit vector (of short and extended enable bits) to find unused entries. Moreover, when read, register bit *2n + 1* always has the same value as register bit *2n*, because an extended address occupies the same memory as two short-address entries.

**Matching Algorithm**

The SRCMATCH.SRC\_MATCH\_EN bit controls whether source address matching is enabled or not. When enabled (which is the default setting) and a frame passes the filtering algorithm, the radio applies one of the algorithms outlined in Figure 23-13, depending on which type of source address is present.

The result is reported in two different forms:

- A 24-bit vector called SRCRESMASK contains a 1 for each enabled short entry with a match, or two 1s for each enabled extended entry with a match (the bit mapping is the same as for the address-enable registers on read access).
- A 7-bit value called SRCRESINDEX:
  - When no source address is present in the received frame, or there is no match on the received source address:
    - Bits 6:0: 011 1111
  - If there is a match on the received source address:
    - Bits 4:0: The index of the first entry (that is, the one with the lowest index number) with a match, 0–23 for short addresses or 0–11 for extended addresses.
    - Bit 5: 0 if the match is on a short address, 1 if the match is on an extended address
    - Bit 6: The result of the AUTOPEND function

Short Source Address (Mode 2)	Extended Source Address (Mode 3)
<p>The received source PAN ID is called srcPanid. The received short address is called srcShort.</p> <pre> SRCRESMASK = 0x000000; SRCRESINDEX = 0x3F; for (n = 0; n &lt; 24; n++) {     bitVector = 0x000001 &lt;&lt; n;     if (SRCSHORTEN &amp; bitVector) {         if ((panid[n] == srcPanid) &amp;&amp;             (short[n] == srcShort)) {             SRCRESMASK  = bitVector;             if (SRCRESINDEX == 0x3F) {                 SRCRESINDEX = n;             }         }     } } </pre>	<p>The received extended address is called srcExt.</p> <pre> SRCRESMASK = 0x000000; SRCRESINDEX = 0x3F; for (n = 0; n &lt; 12; n++) {     bitVector = 0x000003 &lt;&lt; (2*n);     if (SRCEXTEN &amp; bitVector) {         if (ext[n] == srxExt) {             SRCRESMASK  = bitVector;             if (SRCRESINDEX == 0x3F) {                 SRCRESINDEX = n   0x20;             }         }     } } </pre>

**Figure 23-13. Matching Algorithm for Short and Extended Addresses**

SRCRESMASK and SRCRESINDEX are written to RF Core memory as soon as the result is available.

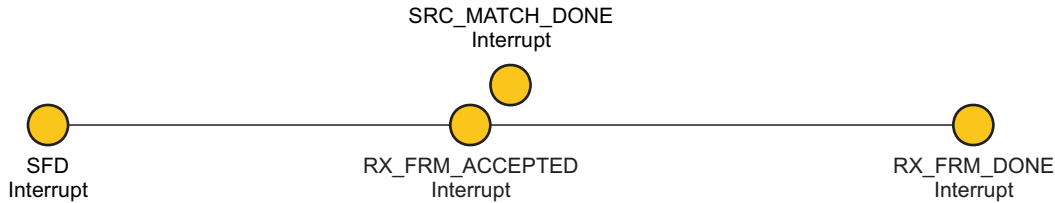
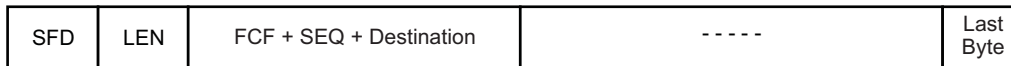
SRCRESINDEX is also appended to received frames if the FRMCTRL0.AUTOCRC and FRMCTRL0.APPEND\_DATA\_MODE bits have been set. The value then replaces the 7-bit correlation value of the 16-bit status word.

**Interrupts**

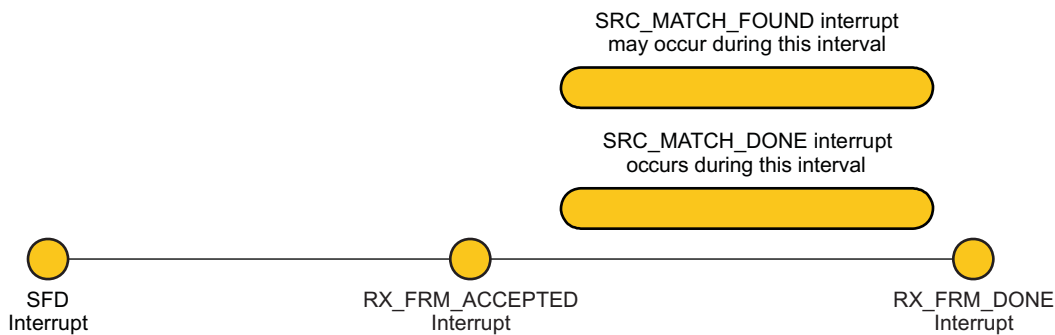
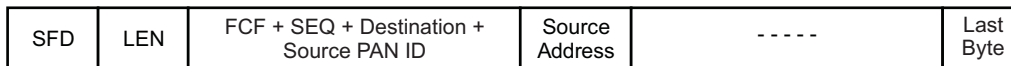
When source address matching is enabled and the matching algorithm completes, the SRC\_MATCH\_DONE interrupt flag is set, regardless of the result. If a match is found, the SRC\_MATCH\_FOUND flag is also set immediately before SRC\_MATCH\_DONE.

Figure 23-14 illustrates the timing of the setting of flags:

**When There Is No Source Address:**



**When There Is a Source Address:**



M0113-01

**Figure 23-14. Interrupts Generated by Source Address Matching**

**Tips and Tricks**

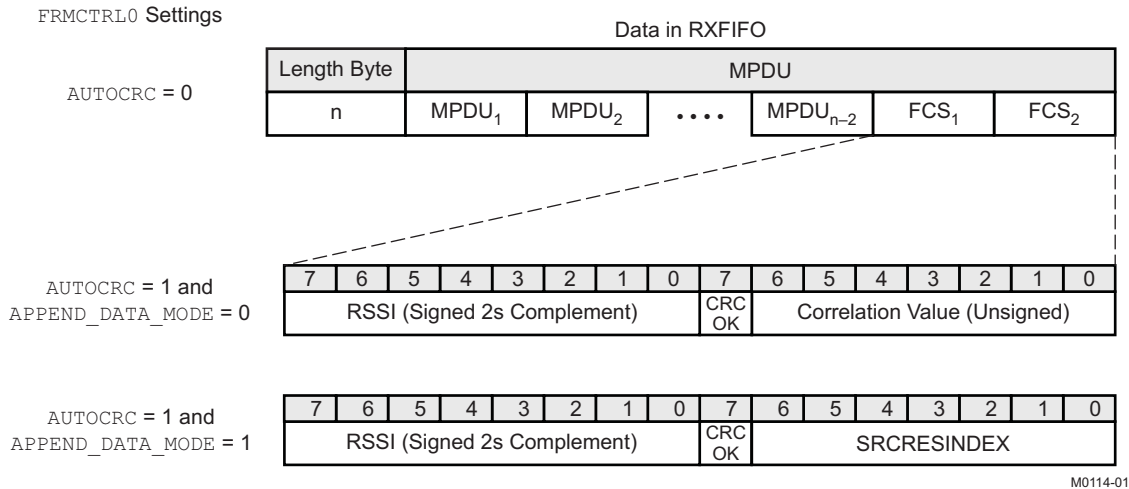
- The source address table can be modified safely during frame reception. If one address replaces another while the receiver is active, the corresponding enable bit should be turned off during the modification. This prevents the RF Core from using a combination of old and new values, because it only considers entries that are enabled throughout the whole source matching process.

The following measures can be taken to avoid the next received frame overwriting the results from source address matching:

- Use the appended SRCRESINDEX result instead of the value written to RAM (this is the recommended approach).
- Read the results from RAM before RX\_FRM\_ACCEPTED occurs in the next received frame. For the shortest frame type, this happens after the sequence number, so the total available time (absolute worst-case with a small safety margin) becomes:
 
$$16 \mu\text{s (required preamble)} + 32 \mu\text{s (SFD)} + 128 \mu\text{s (4 bytes)} = 176 \mu\text{s}$$
- To increase the available time, clear the `FSMCTRL.RX2RX_TIME_OFF` bit. This adds another 192  $\mu\text{s}$ , for a total of 368  $\mu\text{s}$ . This also reduces the risk of RX overflow.

### 23.9.7 Frame-Check Sequence

In receive mode, the FCS is verified by hardware if `FRMCTRL0.AUTOCRC` is enabled. The user is normally only interested in the correctness of the FCS, not the FCS sequence itself. The FCS sequence itself is therefore not written to the RXFIFO during receive. Instead, when `FRMCTRL0.AUTOCRC` is set, the two FCS bytes are replaced by other more-useful values. The values that are substituted for the FCS sequence are configurable in the `FRMCTRL0` register.



**Figure 23-15. Data in RXFIFO for Different Settings**

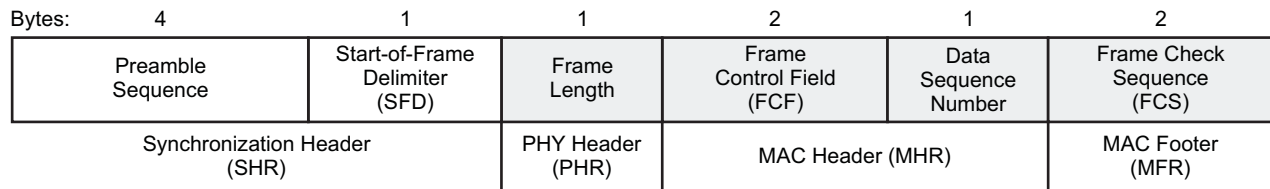
Field Descriptions:

- The *RSSI* value is measured over the first eight symbols following the SFD.
- The *CRC\_OK* bit indicates whether the FCS is correct (1) or incorrect (0). When incorrect, software is responsible for discarding the frame.
- The *correlation value* is the average correlation value over the first eight symbols following the SFD.
- *SRCRESINDEX* is the same value that is written to RAM after completion of source address matching.

Calculation of the LQI value used by IEEE 802.15.4 is described in [Section 23.10.4](#).

### 23.9.8 Acknowledgement Transmission

The radio includes hardware support for acknowledgment transmission after successful frame reception (that is, the FCS of the received frame must be correct). [Figure 23-16](#) shows the format of the acknowledgment frame.



**Figure 23-16. Acknowledge Frame Format**

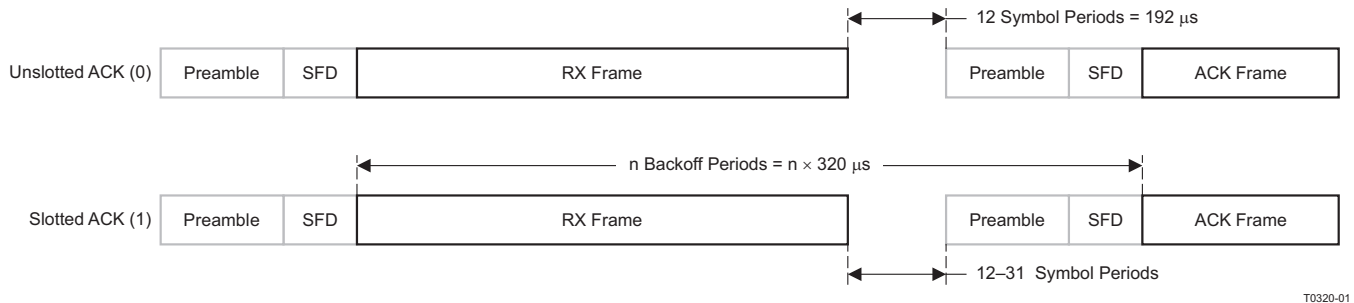
There are three variable fields in the generated acknowledgment frame:

- The pending bit, which may be controlled with command strobes and the AUTOPEND feature (in the FCF field)
- The data sequence number (DSN), which is taken automatically from the last received frame
- The FCS, which is given implicitly

There are three different sources for setting the pending bit in an ACK frame (that is, the SACKPEND strobe, the `PENDING_OR` register bit, and the AUTOPEND feature). The pending bit is set if one or more of these sources are set.

**Transmission Timing**

Acknowledgment frames can only be transmitted immediately after frame reception. The transmission timing is controlled by the `FSMCTRL.SLOTTED_ACK` bit.

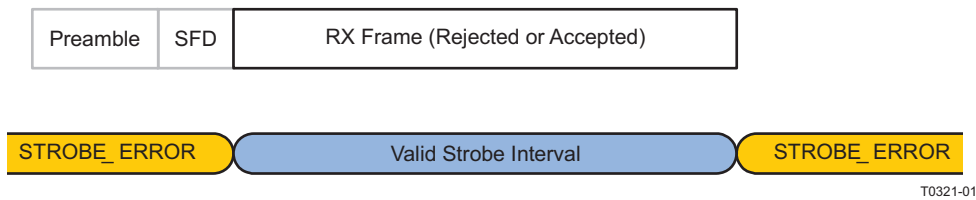


**Figure 23-17. Acknowledgment Timing**

The IEEE 802.15.4 requires unslotted mode in nonbeacon-enabled PANs, and slotted mode for beacon-enabled PANs.

**Manual Control**

The SACK, SACKPEND, and SNACK command strobes can only be issued during frame reception. If the strobes are issued at any other time, they have no effect but generating a `STROBE_ERROR` interrupt.



**Figure 23-18. Command Strobe Timing**

The command strobes may be issued several times during reception; however, only the last strobe has an effect:

- No strobe, or SNACK, or incorrect FCS: No acknowledgment transmission
- SACK: Acknowledgment transmission with the frame-pending bit cleared
- SACKPEND: Acknowledgment transmission with the frame-pending bit set

**Automatic Control (AUTOACK)**

When `FRMFILTO.FRM_FILTER_EN` and `FRMCTRL0.AUTOACK` are both enabled, the radio determines automatically whether or not to transmit acknowledgment frames:

- The RX frame must be accepted by frame filtering (indicated by the `RX_FRM_ACCEPTED` exception).
- The acknowledgment request bit must be set in the RX frame.
- The RX frame must not be a beacon or an acknowledgment frame.
- The FCS of the RX frame must be correct.

Automatic acknowledgments can be overridden by the SACK, SACKPEND, and SNACK command strobes. For instance, if the microcontroller is low on memory resources and cannot store a received frame, the SNACK strobe can be issued during reception and prevent acknowledging the discarded frame.

By default, the AUTOACK feature never sets the frame-pending bit in the acknowledgment frames. Apart from manual override with command strobes, there are two options:

- Automatic control, using the AUTOPEND feature
- Manual control, using the `FRMCTRL1.PENDING_OR` bit

### Automatic Setting of the Frame Pending Field (AUTOPEND)

When the `SRCMATCH.AUTOPEND` bit is set, the result from source address matching determines the value of the frame-pending field. On reception of a frame, the frame-pending field in the (possibly) returned acknowledgment is set, given that:

- `FRMFILTO.FRAME_FILTER_EN` is set.
- `SRCMATCH.SRC_MATCH_EN` is set.
- `SRCMATCH.AUTOPEND` is set.
- The received frame matches the current `SRCMATCH.PEND_DATAREQ_ONLY` setting.
- The received source address matches at least one source-match table entry, which is enabled in both `SRCSHORTEN` and `SRCSHORTPENDEN`, or `SRCEXTEN` and `SRCEXTPENDEN`.

If the source-matching table runs full, the `FRMCTRL1.PENDING_OR` bit may be used to override the AUTOPEND feature and temporarily acknowledge all frames with the frame-pending field set.

## 23.10 RXFIFO Access

The RXFIFO can hold one or more received frames, provided that the total number of bytes is 128 or less. There are two ways to determine the number of bytes in the RXFIFO:

- Reading the `RXFIFOCNT` register
- Using the FIFOP and FIFO signals in combination with the `FIFOPCTRL.FIFOPTHR` setting

The RXFIFO is accessed through the `RFD` register.

The data in the RXFIFO can also be accessed by accessing the radio RAM directly. The FIFO pointers are readable in `RXFIRST_PTR`, `RXLAST_PTR`, and `RXP1_PTR`. This can be useful if one wants to access quickly a certain byte within a frame without having to read out the entire frame first. Note that when using this direct accessing, no FIFO pointers are updated.

The `ISFLUSHRX` command strobe resets the RXFIFO, resetting all FIFO pointers and clearing all counters, status signals, and sticky error conditions. However, if the receiver is actively receiving a frame when the FIFO is flushed, the `RFERRF.ABO` flag is asserted.

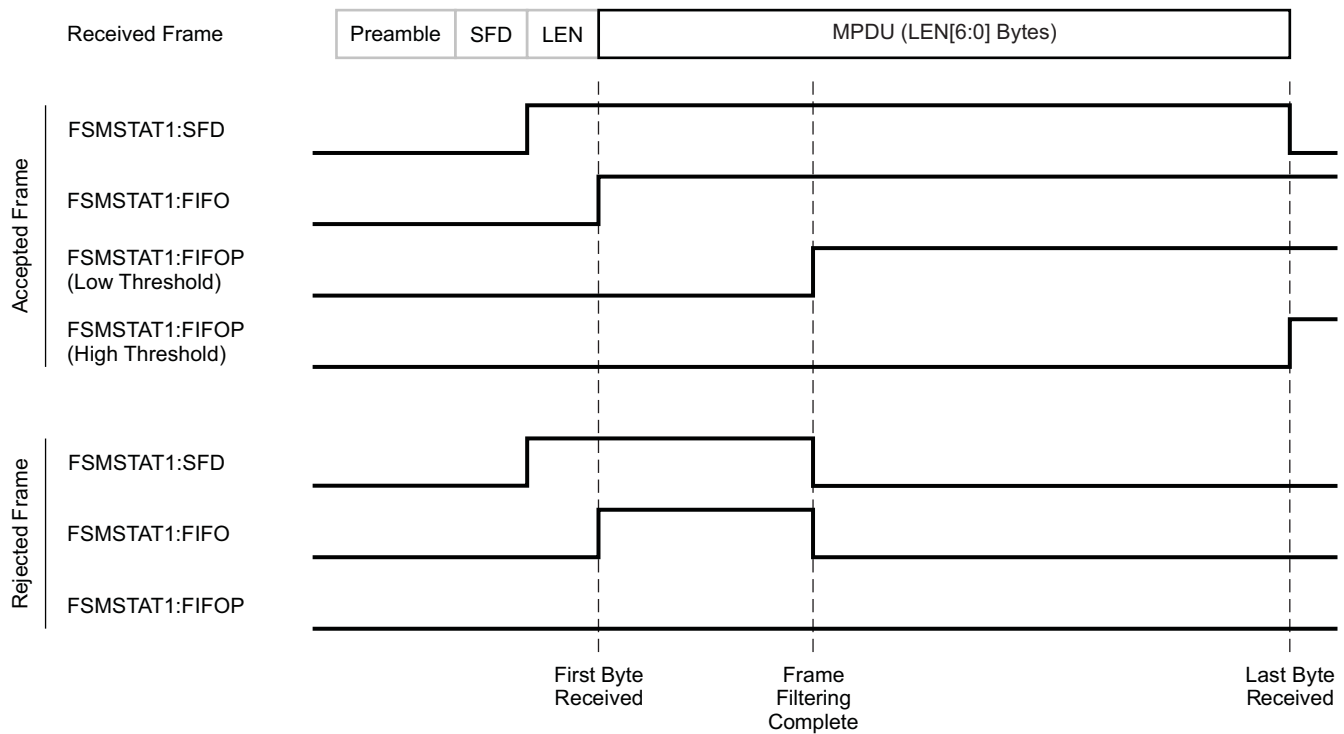
The `SFLUSHRX` command strobe resets the RXFIFO, removing all received frames and clearing all counters, status signals, and sticky-error conditions.

### 23.10.1 Using the FIFO and FIFOP

The FIFO and FIFOP signals are useful when reading out received frames in small portions while the frame is received:

- `FSMSTAT1.FIFO` goes high when one or more bytes are in the RXFIFO, but low when RX overflow has occurred.
- The `FSMSTAT1.FIFOP` signal goes high when:
  - The number of valid bytes in the RXFIFO exceeds the FIFOP threshold value programmed into `FIFOPCTRL`. When frame filtering is enabled, the bytes in the frame header are not considered valid until the frame has been accepted.
  - The last byte of a new frame is received, even if the FIFOP threshold is not exceeded. If so, FIFOP goes back to low at the next RXFIFO read access.





T0322-01

Figure 23-19. Behavior of FIFO and FIFOP Signals

When using the FIFOP as an interrupt source for the microcontroller, the FIFOP threshold should be adjusted by the interrupt service routine to prepare for the next interrupt. When preparing for the last interrupt for a frame, the threshold should match the number of bytes remaining.

### 23.10.2 Error Conditions

There are two error conditions associated with the RXFIFO:

- Overflow, in which case the RXFIFO is full when another byte is received
- Underflow, in which case software attempts to read a byte from an empty RXFIFO

RX overflow is indicated by the `RFERRF.RXOVERF` flag being set and by the signal values `FSMSTAT1.FIFO = 0` and `FSMSTAT1.FIFOP = 1`. When the error occurs, frame reception is halted. The frames currently stored in the RXFIFO may be read out before the condition is cleared with the `ISFLUSHRX` strobe. Note that rejected frames can generate RX overflow if the condition occurs before the frame is rejected.

RX underflow is indicated by the `RFERRF.RXUNDERF` flag being set. RX underflow is a serious error condition that should not occur in error-free software, and the `RXUNDERF` event should only be used for debugging or in a *watchdog* function. Note that the `RXUNDERF` error is not generated when the read operation occurs simultaneously with the reception of a new byte.

### 23.10.3 RSSI

The radio has a built-in received signal-strength indication (RSSI), which calculates an 8-bit signed digital value that can be read from a register or automatically appended to received frames. The RSSI value is the result of averaging the received power over eight symbol periods (128  $\mu$ s) as specified by IEEE 802.15.4 [1].

The RSSI value is a 2s-complement signed number on a logarithmic scale with 1-dB steps.

The status bit `RSSI_VALID` should be checked before reading the `RSSI` value register. `RSSI_VALID` indicates that the `RSSI` value in the register is in fact valid, which means that the receiver has been enabled for at least eight symbol periods.

To find the actual signal power  $P$  at the RF pins with reasonable accuracy, an offset must be added to the `RSSI` value.

$$P = \text{RSSI} - \text{OFFSET [dBm]}$$

For example, with an offset of 73 dB, reading an `RSSI` value of  $-10$  from the `RSSI` register means that the RF input power is approximately  $-83$  dBm. For the correct offset value to use, see the data sheet ([Appendix C](#)).

There are two ways the radio can update the `RSSI` register after it has first become valid. If `FRMCTRL0.ENERGY_SCAN = 0` (default), the `RSSI` register contains the latest value available, but if this bit is set to 1, a peak search is performed, and the `RSSI` register contains the largest value since the energy scan was enabled.

### 23.10.4 Link Quality Indication

The link quality indication (LQI) is a measurement of the strength and/or quality of the received frame as defined by the IEEE 802.15.4 standard [1]. The LQI value is required by the IEEE 802.15.4 standard [1] to be limited to the range 0 through 255, with at least eight unique values. The radio does not provide an LQI value directly, but reports several measurements that can be used by the microcontroller to calculate an LQI value.

The `RSSI` value can be used by the MAC software to calculate the LQI value. This approach has the disadvantage that, for example, a narrowband interferer inside the channel bandwidth can increase the `RSSI` and thus the LQI value, although the true link quality actually decreases. The radio therefore also provides an average correlation value for each incoming frame, based on the first eight symbols following the SFD. This unsigned 7-bit value can be looked on as a measurement of the *chip error rate*, although the radio does not do chip decision.

As described in [Section 23.9.7](#), the average correlation value for the first eight symbols is appended to each received frame, together with the `RSSI` and CRC OK or not-OK, when `FRMCTRL0.AUTOCRC` is set. A correlation value of approximately 110 indicates a maximum-quality frame, whereas a value of approximately 50 is typically the lowest-quality frame detectable by the radio.

Software must convert the correlation value to the range 0–255 as defined by [1], for instance by calculating:

$$\text{LQI} = (\text{CORR} - a)b$$

limited to the range 0–255, where  $a$  and  $b$  are found empirically based on PER measurements as a function of the correlation value.

A combination of `RSSI` and correlation values may also be used to generate the LQI value.

### 23.11 Radio-Control State Machine

The FSM module is responsible for maintaining the TXFIFO and RXFIFO pointers, control of analog *dynamic* signals such as power up and power down, control of the data flow within the RF Core, generation of automatic acknowledgment frames, and control of all analog RF calibration.

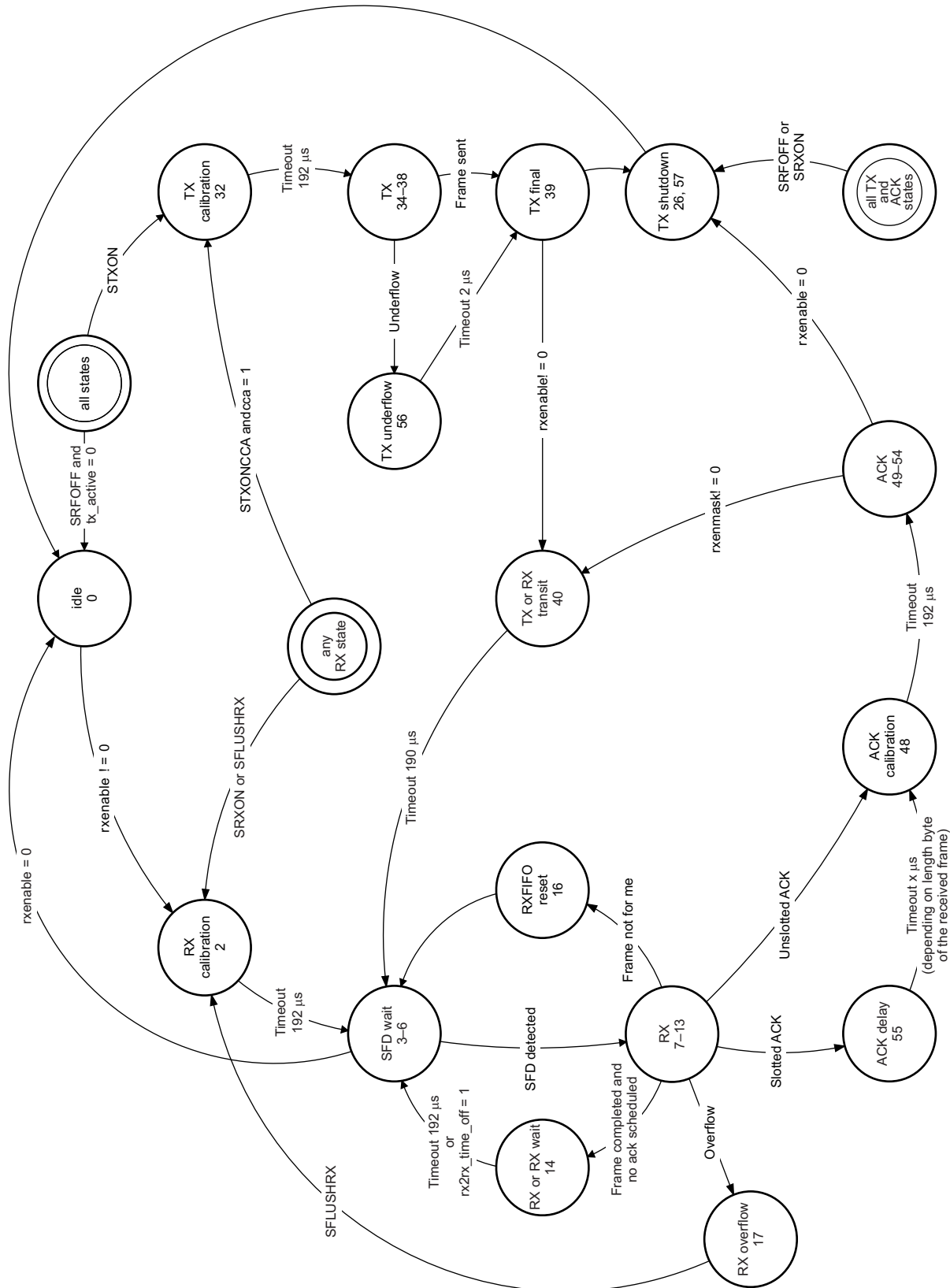


Figure 23-20. Main FSM

Table 23-3 shows the mapping from FSM state to the number which can be read from the `FSMSTAT0` register. Note that although it is possible to read the state of the FSM, this information should not be used to control the program flow in the application software. The states may change very quickly (every 32-MHz clock cycle), and an 8-MHz SPI is not able to capture all the activities.

**Table 23-3. FSM State Mapping**

State Name	State Number, Decimal	Number, Hex	TX_ACTIVE	RX_ACTIVE
Idle	0	0x00	0	0
RX calibration	2	0x02	0	1
SFD wait	3–6	0x03–0x06	0	1
RX	7–13	0x07–0x0D	0	1
TX or RX wait	14	0x0E	0	1
RXFIFO reset	16	0x10	0	1
RX overflow	17	0x11	0	0
TX calibration	32	0x20	1	0
TX	34–38	0x22–0x26	1	0
TX final	39	0x27	1	0
TX or RX transit	40	0x28	1	0
ACK calibration	48	0x30	1	0
ACK	49–54	0x31–0x36	1	0
ACK delay	55	0x37	1	0
TX underflow	56	0x38	1	0
TX shutdown	26, 57	0x1A, 0x39	1	0

## 23.12 Random-Number Generation

The RF Core can generate random bits. The chip should be in RX when generation of random bits is required. One must also make sure that the chip has been in RX long enough for the transients to have died out. A convenient way to do this is to wait for the RSSI-valid signal to go high.

Single random bits from either the I or Q channel can be read from the register `RFRND`.

Randomness tests show good results for this module. However, a slight dc component exists. In a simple test where the `RFRND`.`IRND` register was read a number of times and the data grouped into bytes, about 20 million bytes were read. When interpreted as unsigned integers between 0 and 255, the mean value was 127.6518, which indicates that there is a dc component.

The FFT of the first  $2^{14}$  bytes is shown in [Figure 23-21](#). Note that the dc component is clearly visible. A histogram (32 bins) of the 20 million values is shown in [Figure 23-22](#).

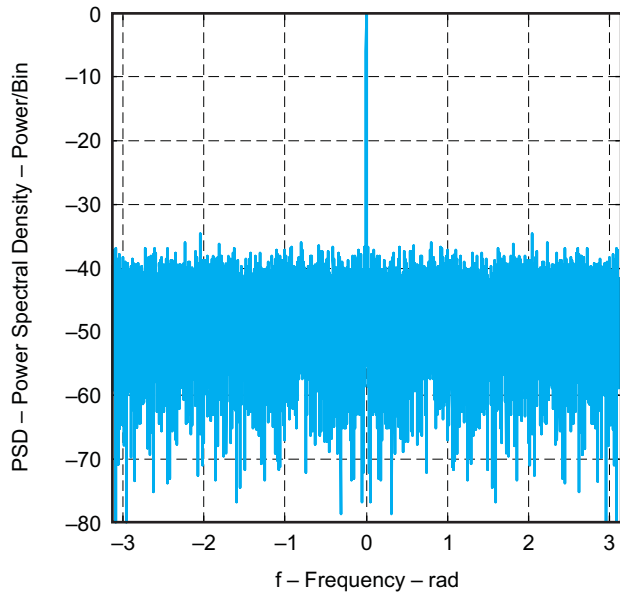


Figure 23-21. FFT of the Random Bytes

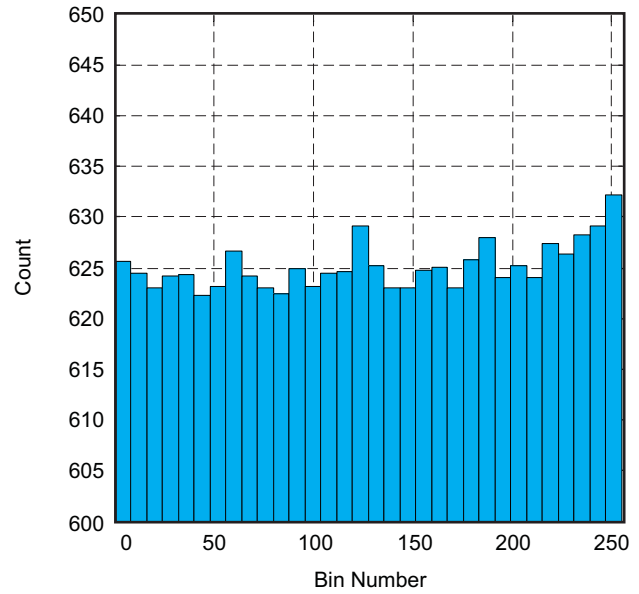


Figure 23-22. Histogram of 20 Million Bytes Generated With the RANDOM Instruction

For the first 20 million individual bits, the probability of a one is  $P(1) = 0.500602$  and  $P(0) = 1 - P(1) = 0.499398$ .

Note that to fully qualify the random generator as *true random*, much more elaborate tests are required. There are software packages available on the internet that may be useful in this respect.

### 23.13 Packet Sniffing and Radio Test Output Signals

Packet sniffing is a nonintrusive way of observing data that is either being transmitted or received. The packet sniffer outputs a clock and a data signal, which should be sampled on the rising edges of the clock. The two packet sniffer signals are observable as GPIO outputs. For accurate time stamping, the SFD signal should also be output.

Because the radio has a data rate of 250 kbps, the packet sniffer clock frequency is 250 kHz. The data is output serially, with the MSB of each byte first, which is opposite of the actual RF transmission, but more convenient when processing the data. It is possible to use a SPI slave to receive the data stream.

When sniffing frames in TX mode, the data that is read from the TXFIFO by the modulator is the same data that is output by the packet sniffer. However, if automatic CRC generation is enabled, the packet sniffer does NOT output these 2 bytes. Instead, it replaces the CRC bytes with 0x8080. This value can never occur as the last two bytes of a received frame (when automatic CRC checking is enabled), and thus it provides a way for the receiver of the sniffed data to separate frames that were transmitted and frames that were received.

When sniffing frames in RX mode, the data that is written to the RXFIFO by the demodulator is the same data that is output by the packet sniffer. In other words, the last two bytes are either the received CRC value or the CRC OK, RSSI, correlation, or SRCRESINDEX value that may automatically replace the CRC value, depending on configuration settings.

To set up the packet sniffer signals or some of the other RF Core observation outputs (in total maximum 3; `rfc_obs_sig0`, `rfc_obs_sig1`, and `rfc_obs_sig2`), the user must perform the following steps:

**Step1:** Determine which signal (`rfc_obs_sig`) to output on which GPIO pin (P1[0:5]). This is done using the OBSSELx control registers (OBSSEL0–OBSSEL5) that control the observation output to pins P1[0:5] (overriding the standard GPIO behavior for those pins).

**Step2:** Set the RFC\_OBS\_CTRL control registers (RFC\_OBS\_CTRL0–RFC\_OBS\_CTRL2) to select the correct signals (`rfc_obs_sig`); for example, for packet sniffing one needs the `rfc_sniff_data` for the packet sniffer data signal and `rfc_sniff_clk` for the corresponding clock signal.

**Step3:** For packet sniffing, the packet sniffer module must be enabled in the MDMTEST1 register.

### 23.14 Command Strobe Processor

The command strobe processor (CSP) provides the control interface between the CPU and the radio.

The CSP interfaces with the CPU through the SFR register RFST and the XREG registers CSPX, CSPY, CSPZ, CSPT, CSPSTAT, CSPCTRL, and CSPPROG<n> (where n is in the range 0 to 23). The CSP produces interrupt requests to the CPU. In addition, the CSP interfaces with the MAC Timer by observing MAC Timer events.

The CSP allows the CPU to issue command strobes to the radio, thus controlling the operation of the radio.

The CSP has two modes of operation, which are described as follows.

- Immediate command strobe execution
- Program execution

Immediate command strobes are written as Immediate Command Strobe instructions to the CSP, which are issued instantly to the radio module. The Immediate Command Strobe instructions are also used to control the CSP. The Immediate Command Strobe instructions are described in [Section 23.14.8](#).

Program execution mode means that the CSP executes a sequence of instructions, comprising a short user-defined program, from a program memory or instruction memory. The available instructions are from a set of 20 instructions. The instruction set is defined in [Section 23.14.8](#). The required program is first loaded into the CSP by the CPU, and then the CPU instructs the CSP to start executing the program.

The program execution mode, together with the MAC Timer, allows the CSP to automate command-strobe-processor algorithms and thus act as a coprocessor for the CPU.

The operation of the CSP is described in detail in the following sections. The command strobes and other instructions supported by the CSP are given in [Section 23.14.9](#).

**RFST (0xE1) – RF CommandStrobe Processor**

Bit	Name	Reset	R/W	Description
7:0	INSTR[7:0]	0xD0	R/W	Data written to this register is written to the CSP instruction memory. Reading this register returns the CSP instruction currently being executed.

#### 23.14.1 Instruction Memory

The CSP executes single-byte program instructions which are read from a 24-byte instruction memory. Writes to the instruction memory are sequential, written through SFR register RFST. An instruction write pointer is maintained within the CSP to hold the location within the instruction memory where the next instruction written to RFST is to be stored. For debugging purposes, the program currently loaded into the CSP can be read from the XREG registers CSPPROG<n>. Following a reset, the write pointer is reset to location 0. During each RFST register write, the write pointer is incremented by 1 until the end of memory is reached, at which time the write pointer stops incrementing. The first instruction written to RFST is stored in location 0, the location where program execution starts. Thus, a complete 24-instruction program is written to the instruction memory by writing each instruction in the desired order to the RFST register.

The write pointer can be reset to 0 by writing the immediate command strobe instruction ISSTOP. In addition, the write pointer is reset to 0 when the command strobe SSTOP is executed in a program.

Following a reset, the instruction memory is filled with SNOP (No Operation) instructions (opcode value 0xC0). The immediate strobe ISCLEAR clears the instruction memory, filling it with SNOP instructions.

While the CSP is executing a program, there must be no attempts to write instructions to the instruction memory by writing to RFST. Failure to observe this rule can lead to incorrect program execution and corrupt instruction memory contents. However, Immediate Command Strobe instructions may be written to RFST (see [Section 23.14.3](#)).

### 23.14.2 Data Registers

The CSP has four data registers, CSPT, CSPX, CSPY, and CSPZ, which are read- and write-accessible for the CPU as XREG registers. These registers are read or modified by some instructions, thus allowing the CPU to set parameters to be used by a CSP program, or allowing the CPU to read CSP program status.

The CSPT data register is not modified by any instruction. The CSPT data register is used to set a MAC Timer overflow-compare value. Once program execution has started on the CSP, the content of this register is decremented by 1 each time the MAC Timer overflows. When CSPT reaches zero, program execution is halted and the interrupt IRQ\_CSP\_STOP is asserted. The CSPT register is not decremented if the CPU writes 0xFF to this register.

---

**NOTE:** If the CSPT register compare function is not used, this register must be set to 0xFF before the program execution is started.

---

### 23.14.3 Program Execution

After the instruction memory has been filled, program execution is started by writing the immediate command strobe instruction ISSTART to the RFST register. Program execution continues until the instruction at the last location has been executed, the CSPT data register content is zero, an SSTOP instruction has been executed, an immediate ISSTOP instruction is written to RFST, or a SKIP instruction returns a location beyond the last location in the instruction memory. The CSP runs at the set system clock frequency, which must be set to 32 MHz for correct radio operation.

Immediate command strobe instructions may be written to RFST while a program is being executed. In this case, the immediate instruction is executed before the instruction in the instruction memory, which is executed once the immediate instruction has been completed.

During program execution, reading RFST returns the current instruction being executed. An exception to this is the execution of immediate command strobos, during which RFST returns 0xD0.

### 23.14.4 Interrupt Requests

The CSP has three interrupt flags which can produce the RF interrupt vector. These are the following:

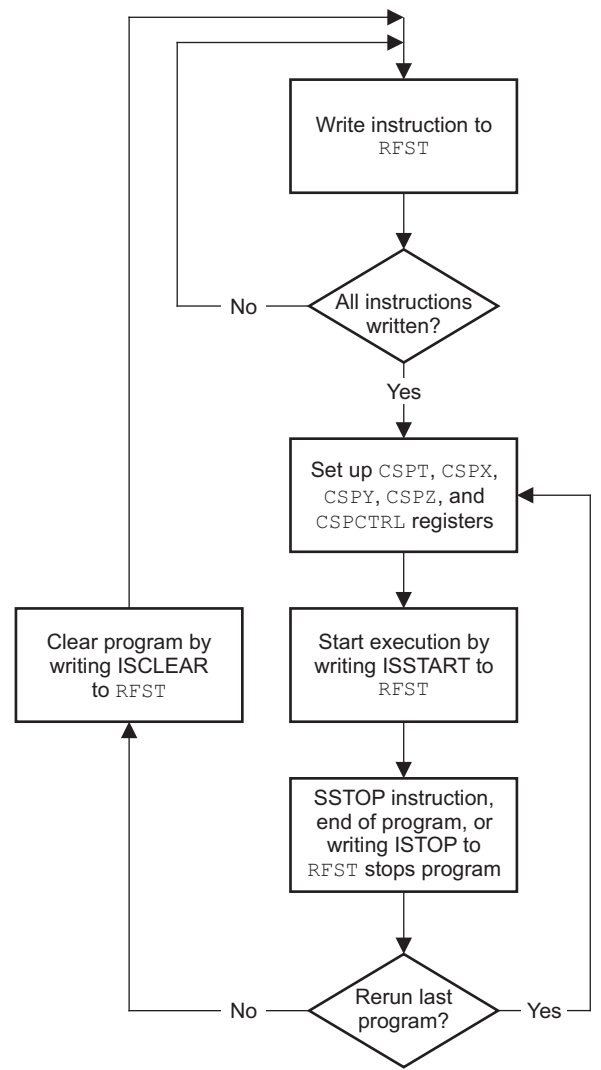
- IRQ\_CSP\_STOP: asserted when the processor has executed the last instruction in memory or when the processor stops due to an SSTOP or ISSTOP instruction or the CSPT register being equal to zero
- IRQ\_CSP\_WT: asserted when the processor continues executing the next instruction after a WAIT W or WAITX instruction
- IRQ\_CSP\_INT: asserted when the processor executes an INT instruction

### 23.14.5 Random Number Instruction

There is a delay in the update of the random number used by the RANDXY instruction. Therefore, if the instruction RANDXY, which uses this value, is issued immediately after a previous RANDXY instruction, the random value read may be the same in both cases.

### 23.14.6 Running CSP Programs

The basic flow for loading and running a program on the CSP is shown in [Figure 23-23](#). When program execution stops at the end of the program, the current program remains in program memory so that the same program can be run again by starting execution once again with the ISSTART command. To clear the program contents, use the ISCLEAR instruction.



F0037-01

**Figure 23-23. Running a CSP Program**

### 23.14.7 Registers

#### CSPROG<N> (N Ranging From 0 to 23) (0x61C0 + N) – CSP Program

Bit	Name	Reset	R/W	Description
7:0	CSP_INSTR	0xD0	R	Byte N of the CSP program memory

#### CSPCTRL (0x61E0) – CSP Control Bit

Bit	Name	Reset	R/W	Description
7:1	–	0000 000	R0	Reserved. Read as 0
0	MCU_CTRL	0	R/W	CSP MCU control input



**CSPSTAT (0x61E1) – CSP Status Register**

Bit	Name	Reset	R/W	Description
7:6	–	00	R0	Reserved. Read as 0
5	CSP_RUNNING	0	R	1: CSP is running. 0: CSP is idle.
4:0	CSP_PC	0 0000	R	CSP program counter

**CSPX (0x61E2) – CSP X Register**

Bit	Name	Reset	R/W	Description
7:0	CSPX	0x00	R/W	CSP X data register. Used by CSP instructions WAITX, RANDXY, INCX, DECX, and conditional instructions

**CSPY (0x61E3) – CSP Y Register**

Bit	Name	Reset	R/W	Description
7:0	CSPY	0x00	R/W	CSP Y data register. Used by CSP instructions RANDXY, INCY, DECY, and conditional instructions

**CSPZ (0x61E4) – CSP Z Register**

Bit	Name	Reset	R/W	Description
7:0	CSPZ	0x00	R/W	CSP Z data register. Used by CSP instructions INCZ, DECZ, and conditional instructions

**CSPT (0x61E5) – CSP T Register**

Bit	Name	Reset	R/W	Description
7:0	CSPT	0xFF	R/W	CSP T data register. Content is decremented each time the MAC Timer overflows while the CSP program is running. The SCP program stops when decremented to 0. Setting CSPT = 0xFF prevents the register from being decremented.

### 23.14.8 Instruction Set Summary

This section gives an overview of the instruction set. This is intended as a summary and definition of instruction opcodes. See [Section 23.14.9](#) for a description of each instruction. Each instruction consists of one byte, which is written to the *RFST* register to be stored in the instruction memory.

The Immediate Strobe instructions (ISxxx) are not used in a program. When these instructions are written to the *RFST* register, they are executed immediately. If the CSP is already executing a program, the current instruction is delayed until the immediate strobe instruction has completed.

For undefined opcodes, the behavior of the CSP is defined as a no-operation strobe command (SNOP).

**Table 23-4. Instruction Set Summary**

Mnemonic	7	6	5	4	3	2	1	0	Description
SKIP <C>, <S>	0	S2	S1	S0	N	C2	C1	C0	Skip S instructions on condition C. When condition (C XOR N) is true, skip the next S instructions, else execute the next instruction. If S = 0, re-execute the conditional jump (that is, busy loop until condition is false). Skipping past the last instruction in the command buffer results in an implicit STOP command. The conditions are:  C = 0 CCA true C = 1 Synchronization word received and still receiving packet or synchronization word transmitted and still transmitting packet (SFD found, not yet frame end) C = 2 MCU control bit is 1. C = 3 Reserved C = 4 Register X = 0 C = 5 Register Y = 0 C = 6 Register Z = 0 C = 7 RSSI_VALID = 1
WAIT <W>	1	0	0	W4	W3	W2	W1	W0	Wait for MAC Timer to overflow W times. Waits until the MAC Timer has overflowed W times (W = 0 waits 32 times), then continues execution. Generates an IRQ_CSP_WAIT interrupt request when execution continues.
RPT <C>	1	0	1	0	N	C2	C1	C0	Repeat loop while condition C. If condition C is true, go to the instruction following the last LABEL instruction (address in loop-start register); if the condition is false or no LABEL instruction has been executed, go to the next instruction.  Note condition C is as defined for SKIP, defined previously in this table. It is not possible to have a RPT instruction placed at index 23 of the command buffer.
WEVENT1	1	0	1	1	1	0	0	0	Wait for mact_event1 to go high, and then continue execution.
WEVENT2	1	0	1	1	1	0	0	1	Wait for mact_event2 to go high, and then continue execution.
INT	1	0	1	1	1	0	1	0	Generate an IRQ_CSP_MANINT. Issues an IRQ_CSP_MANINT interrupt request.
LABEL	1	0	1	1	1	0	1	1	Set the next instruction as the start of a repeat loop. Enters the address of the next instruction into the loop-start register.
WAITX	1	0	1	1	1	1	0	0	Wait for MAC Timer to overflow [X] times, where [X] is the value of register X. Each time a MAC Timer overflow is detected, X is decremented. Execution continues as soon as X = 0. (If X = 0 when instruction is run, no wait is performed and execution continues directly). An IRQ_CSP_WAIT interrupt request is generated when execution continues.
RANDXY	1	0	1	1	1	1	0	1	Load the [Y] LSBs of register X with random value.
SETCMP1	1	0	1	1	1	1	1	0	Set the output csp_mact_setcmp1 high. This sets the compare value of the MAC Timer to the current timer value.
INCX	1	1	0	0	0	0	0	0	Increment register X.
INCY	1	1	0	0	0	0	0	1	Increment register Y.
INCZ	1	1	0	0	0	0	1	0	Increment register Z.
DECX	1	1	0	0	0	0	1	1	Decrement register X.
DECY	1	1	0	0	0	1	0	0	Decrement register Y.
DECZ	1	1	0	0	0	1	0	1	Decrement register Z.
INCMAXY <M>	1	1	0	0	1	M2	M1	M0	Register Y ≤ min(Y + 1, M). Increment Y, but not beyond M.
Sxxx	1	1	0	1	S3	S2	S1	S0	Execute command strobe S. Send command strobe S to FFCTRL. Up to 32 command strobes are supported. In addition to the regular command strobes, two additional command strobes that only apply to the command strobe processor are supported:  SNOP: Do nothing.  SSTOP: Stops the command strobe processor execution and invalidates any set label. An IRQ_CSP_STOP interrupt request is issued.

**Table 23-4. Instruction Set Summary (continued)**

Mnemonic	7	6	5	4	3	2	1	0	Description
ISxxx	1	1	1	0	S3	S2	S1	S0	Execute command strobe S immediately. Send command strobe S to FFCTRL immediately, bypassing the instructions in the command buffer. If the current buffer instruction is a strobe, it is delayed. In addition to the regular command strobes, two additional command strobes that only apply to the command strobe processor are supported:  ISSTART: The command strobe processor starts execution at the first instruction in the command buffer. Do not issue an ISSTART instruction if the CSP is already running.  ISSTOP: Stops the command strobe processor execution and invalidates any set label. An IRQ_CSP_STOP interrupt request is issued.
ISCLEAR	1	1	1	1	1	1	1	1	Clear the CSP program. Reset PC.

### 23.14.9 Instruction Set Definition

There are 20 basic instruction types. Furthermore, the command-strobe and immediate-strobe instructions can each be divided into 16 subinstructions, giving an effective number of 42 different instructions. The following subsections describe each instruction in detail.

*Note: the following definitions are used in this section*

PC = CSP program counter  
X = RF register CSPX  
Y = RF register CSPY  
Z = RF register CSPZ  
T = RF register CSPT

#### 23.14.9.1 DECZ

**Function:** Decrement Z

**Description:** The Z register is decremented by 1. An original value of 0x00 underflows to 0xFF.

**Operation:**  $Z = Z - 1$

Opcode: 0xC5

7	6	5	4	3	2	1	0
1	1	0	0	0	1	0	1

#### 23.14.9.2 DECY

**Function:** Decrement Y

**Description:** The Y register is decremented by 1. An original value of 0x00 underflows to 0xFF.

**Operation:**  $Y = Y - 1$

Opcode: 0xC4

7	6	5	4	3	2	1	0
1	1	0	0	0	1	0	0

#### 23.14.9.3 DECX

**Function:** Decrement X  
**Description:** The X register is decremented by 1. An original value of 0x00 underflows to 0xFF.  
**Operation:**  $X = X - 1$

Opcode: 0xC3

7	6	5	4	3	2	1	0
1	1	0	0	0	0	1	1

#### 23.14.9.4 INCZ

**Function:** Increment Z  
**Description:** The X register is incremented by 1. An original value of 0xFF overflows to 0x00.  
**Operation:**  $Z = Z + 1$

Opcode: 0xC2

7	6	5	4	3	2	1	0
1	1	0	0	0	0	1	0

#### 23.14.9.5 INCY

**Function:** Increment Y  
**Description:** The Y register is incremented by 1. An original value of 0xFF overflows to 0x00.  
**Operation:**  $Y = Y + 1$

Opcode: 0xC1

7	6	5	4	3	2	1	0
1	1	0	0	0	0	0	1

#### 23.14.9.6 INCX

**Function:** Increment X  
**Description:** The X register is incremented by 1. An original value of 0xFF overflows to 0x00.  
**Operation:**  $X = X + 1$

Opcode: 0xC0

7	6	5	4	3	2	1	0
1	1	0	0	0	0	0	0

#### 23.14.9.7 INCMAXY

**Function:** Increment Y not greater than M.  
**Description:** The Y register is incremented by 1 if the result is less than M; otherwise, Y register is loaded with value M.  
**Operation:**  $Y = \min(Y + 1, M)$

**Opcode: 0xC8 | M (M = 0–7)**

7	6	5	4	3	2	1	0
1	1	0	0	1	M		

### 23.14.9.8 RANDXY

**Function:** Load random value into X.

**Description:** The [Y] LSBs of the X register are loaded with a random value. Note that if a second RANDXY instruction is issued immediately (within 13 clock cycles) after the first, the same random value is used in both cases. If Y equals zero or is greater than 7, then an 8-bit random value is loaded into X.

**Operation:**  $X[(Y - 1):0] = \text{RNG\_DOUT}[(Y - 1):0]$ ,  $X[7:Y] = 0$

**Opcode: 0xBD**

7	6	5	4	3	2	1	0
1	0	1	1	1	1	0	1

### 23.14.9.9 INT

**Function:** Interrupt

**Description:** The interrupt IRQ\_CSP\_INT is asserted when this instruction is executed.

**Operation:**  $\text{IRQ\_CSP\_INT} = 1$

**Opcode: 0xBA**

7	6	5	4	3	2	1	0
1	0	1	1	1	0	1	0

### 23.14.9.10 WAITX

**Function:** Wait for X MAC Timer overflows

**Description:** Wait for MAC Timer to overflow [X] times, where [X] is the value of register X. Each time a MAC Timer overflow is detected, the value in register X is decremented. Program execution continues as soon as  $X = 0$ . (If  $X = 0$  when instruction is run, no wait is performed and execution continues directly.) An IRQ\_CSP\_WAIT interrupt request is generated when execution continues. **Note:** The difference compared to WAIT W is that W is a fixed value, whereas X is a register value (which could potentially be changed, such that the number of overflows actually does not correspond to the value of X at the time WAITX instruction is run).

**Operation:**  $X = X - 1$  when MAC Timer overflow = true  
 $\text{PC} = \text{PC}$  while  $X > 0$   
 $\text{PC} = \text{PC} + 1$  when  $X = 0$

**Opcode: 0xBC**

7	6	5	4	3	2	1	0
1	0	1	1	1	1	0	0

### 23.14.9.11 SETCMP1

- Function:** Set the compare value of the MAC Timer to the current timer value.  
**Description:** Set the compare value of the MAC Timer to the current timer value.  
**Operation:** Csp\_mact\_setcmp1 = 1

Opcode: 0xBE

7	6	5	4	3	2	1	0
1	0	1	1	1	1	1	0

### 23.14.9.12 WAIT W

- Function:** Wait for W MAC Timer overflows  
**Description:** Wait until MAC Timer overflows a number of times equal to the value of W. If W = 0, the instruction waits for 32 overflows. Program execution continues with the next instruction, and the interrupt flag IRQ\_CSP\_WT is asserted when the wait condition is true.  
**Operation:** PC = PC while number of MAC Timer overflows < W  
 PC = PC + 1 when number of MAC Timer overflows = W

Opcode: 0x80 | W (W = 0–31)

7	6	5	4	3	2	1	0
1	0	0	W				

### 23.14.9.13 WEVENT1

- Function:** Wait until MAC Timer event 1  
**Description:** Wait until next MAC Timer event. Program execution continues with the next instruction when the wait condition is true.  
**Operation:** PC = PC while MAC Timer compare = false  
 PC = PC + 1 when MAC Timer compare = true

Opcode: 0xB8

7	6	5	4	3	2	1	0
1	0	1	1	1	0	0	0

### 23.14.9.14 WEVENT2

- Function:** Wait until MAC Timer event 2  
**Description:** Wait until next MAC Timer event. Program execution continues with the next instruction when the wait condition is true.  
**Operation:** PC = PC while MAC Timer compare = false  
 PC = PC + 1 when MAC Timer compare = true

Opcode: 0xB9

7	6	5	4	3	2	1	0
1	0	1	1	1	0	0	1

### 23.14.9.15 LABEL

**Function:** Set loop label

**Description:** Sets next instruction as start of loop. If the current instruction is the last instruction in the instruction memory, then the current PC is set as start of loop. If several label instructions are executed, the last label executed is the active label. Earlier labels are removed, which means that only one level of loops is supported.

**Operation:** LABEL = PC + 1

Opcode: 0xBB

7	6	5	4	3	2	1	0
1	0	1	1	1	0	1	1

### 23.14.9.16 RPT C

**Function:** Conditional repeat

**Description:** If condition C is true, then jump to the instruction defined by the last LABEL instruction, that is, jump to start of loop. If the condition is false or if a LABEL instruction has not been executed, then execution continues from next instruction. The condition C may be negated by setting N = 1 and is described in the following table.

Condition Code C	Description	Function
000	CCA is true	CCA = 1
001	Synchronization word received and still receiving packet or synchronization word transmitted and still transmitting packet	SFD = 1
010	CPU control true	CSPCTRL.CPU_CTRL = 1
011	Reserved	
100	Register X = 0	X = 0
101	Register Y = 0	Y = 0
110	Register Z = 0	Z = 0
111	RSSI is valid	RSSI_VALID = 1

**Operation:** PC = LABEL when (C XOR N) = true  
PC = PC + 1 when (C XOR N) = false or LABEL = not set

Opcode: 0xA0 | N | C (N = 0, 8; C = 0–7)

7	6	5	4	3	2	1	0
1	0	1	0	N	C		

### 23.14.9.17 SKIP S, C

**Function:** Conditional skip instruction

**Description:** Skip S instructions on condition C (where condition C may be negated; N = 1). When condition (C XOR N) is true, skip the next S instructions, else execute the next instruction. If S = 0, re-execute the conditional jump (that is, busy loop until condition is false). Skipping past the last instruction in the command buffer results in an implicit STOP command.

Condition Code C	Description	Function
000	CCA is true	CCA = 1
001	Synchronization word received and still receiving packet or synchronization word transmitted and still transmitting packet	SFD = 1
010	CPU control true	CSPCTRL.CPU_CTRL = 1
011	Reserved	
100	Register X = 0	X = 0
101	Register Y = 0	Y = 0
110	Register Z = 0	Z = 0
111	RSSI is valid	RSSI_VALID = 1

**Operation:** PC = PC + S + 1 when (C XOR N) = true  
 PC = PC + 1 when (C XOR N) = false

Opcode: 0x00 | S | N | C

7	6	5	4	3	2	1	0
0	S			N	C		

### 23.14.9.18 STOP

**Function:** Stop program execution  
**Description:** The SSTOP instruction stops the CSP program execution.  
**Operation:** Stop execution

Opcode: 0xD2

7	6	5	4	3	2	1	0
1	1	0	1	0	0	1	0

### 23.14.9.19 SNOP

**Function:** No operation  
**Description:** Operation continues at the next instruction.  
**Operation:** PC = PC + 1

Opcode: 0xD0

7	6	5	4	3	2	1	0
1	1	0	1	0	0	0	0

### 23.14.9.20 SRXON

**Function:** Enable and calibrate frequency synthesizer for RX  
**Description:** The SRXON instruction asserts the output FFCTL\_SRXON\_STRB to enable and calibrate the frequency synthesizer for RX. The instruction waits for the radio to acknowledge the command before executing the next instruction.  
**Operation:** SRXON



**Opcode: 0xD3**

7	6	5	4	3	2	1	0
1	1	0	1	0	0	1	1

**23.14.9.21 STXON**

**Function:** Enable TX after calibration

**Description:** The STXON instruction enables TX after calibration. The instruction waits for the radio to acknowledge the command before executing the next instruction. Sets a bit in `RXENABLE` if `SET_RXENMASK_ON_TX` is set

**Operation:** STXON

**Opcode: 0xD9**

7	6	5	4	3	2	1	0
1	1	0	1	1	0	0	1

**23.14.9.22 STXONCCA**

**Function:** Enable calibration and TX if CCA indicates a clear channel

**Description:** The STXONCCA instruction enables TX after calibration if CCA indicates a clear channel.

**Operation:** STXONCCA

**Opcode: 0xDA**

7	6	5	4	3	2	1	0
1	1	0	1	1	0	1	0

**23.14.9.23 SSAMPLECCA**

**Function:** Sample the current CCA value to `SAMPLED_CCA`

**Description:** The current CCA value is written to `SAMPLED_CCA` in XREG.

**Operation:** SSAMPLECCA

**Opcode: 0xDB**

7	6	5	4	3	2	1	0
1	1	0	1	1	0	1	1

**23.14.9.24 SRFOFF**

**Function:** Disable RX or TX and frequency synthesizer.

**Description:** The SRFOFF instruction disables RX or TX and the frequency synthesizer.

**Operation:** SRFOFF

**Opcode: 0xDF**

7	6	5	4	3	2	1	0
1	1	0	1	1	1	1	1

**23.14.9.25 SFLUSHRX**

- Function:** Flush RXFIFO buffer and reset demodulator
- Description:** The SFLUSHRX instruction flushes the RXFIFO buffer and resets the demodulator. The instruction waits for the radio to acknowledge the command before executing the next instruction.
- Operation:** SFLUSHRX

Opcode: 0xDD

7	6	5	4	3	2	1	0
1	1	0	1	1	1	0	1

**23.14.9.26 SFLUSHTX**

- Function:** Flush TXFIFO buffer
- Description:** The SFLUSHTX instruction flushes the TXFIFO buffer. The instruction waits for the radio to acknowledge the command before executing the next instruction.
- Operation:** SFLUSHTX

Opcode: 0xDE

7	6	5	4	3	2	1	0
1	1	0	1	1	1	1	0

**23.14.9.27 SACK**

- Function:** Send acknowledge frame with pending field cleared
- Description:** The SACK instruction sends an acknowledge frame. The instruction waits for the radio to acknowledge the command before executing the next instruction.
- Operation:** SACK

Opcode: 0xD6

7	6	5	4	3	2	1	0
1	1	0	1	0	1	1	0

**23.14.9.28 SACKPEND**

- Function:** Send acknowledge frame with the pending field set
- Description:** The SACKPEND instruction sends an acknowledge frame with the pending field set. The instruction waits for the radio to acknowledge the command before executing the next instruction.
- Operation:** SACKPEND

Opcode: 0xD7

7	6	5	4	3	2	1	0
1	1	0	1	0	1	1	1

**23.14.9.29 SNACK**

- Function:** Abort sending of acknowledge frame
- Description:** The SACKPEND instruction aborts sending acknowledge to the frame currently being received.
- Operation:** SNACK

**Opcode: 0xD8**

7	6	5	4	3	2	1	0
1	1	0	1	1	0	0	0

**23.14.9.30 SRXMASKBITSET**

- Function:** Set bit in *RXENABLE*
- Description:** The SRXMASKBITSET instruction sets bit 5 in the *RXENABLE* register.
- Operation:** SRXMASKBITSET

**Opcode: 0xD4**

7	6	5	4	3	2	1	0
1	1	0	1	0	1	0	0

**23.14.9.31 SRXMASKBITCLR**

- Function:** Clear bit in *RXENABLE*
- Description:** The SRXMASKBITCLR instruction clears bit 5 in the *RXENABLE* register.
- Operation:** SRXMASKBITCLR

**Opcode: 0xD5**

7	6	5	4	3	2	1	0
1	1	0	1	0	1	0	1

**23.14.9.32 ISSTOP**

- Function:** Stop program execution
- Description:** The ISSTOP instruction stops the CSP program execution and the IRQ\_CSP\_STOP interrupt flag is asserted.
- Operation:** Stop execution

**Opcode: 0xE2**

7	6	5	4	3	2	1	0
1	1	1	0	0	0	1	0

**23.14.9.33 ISSTART**

- Function:** Start program execution
- Description:** The ISSTART instruction starts the CSP program execution from first instruction written to instruction memory. Do not issue an ISSTART instruction if CSP is already running.
- Operation:** PC = 0, start execution

Opcode: 0xE1

7	6	5	4	3	2	1	0
1	1	1	0	0	0	0	1

#### 23.14.9.34 ISRXON

- Function:** Enable and calibrate frequency synthesizer for RX
- Description:** The ISRXON instruction immediately enables and calibrates the frequency synthesizer for RX.
- Operation:** SRXON

Opcode: 0xE3

7	6	5	4	3	2	1	0
1	1	1	0	0	0	1	1

#### 23.14.9.35 ISRXMASKBITSET

- Function:** Set bit in `RXENABLE`
- Description:** The ISRXMASKBITSET instruction immediately sets bit 5 in the `RXENABLE` register.
- Operation:** SRXMASKBITSET

Opcode: 0xE4

7	6	5	4	3	2	1	0
1	1	1	0	0	1	0	0

#### 23.14.9.36 ISRXMASKBITCLR

- Function:** Clear bit in `RXENABLE`
- Description:** The ISRXMASKBITCLR instruction immediately clears bit 5 in the `RXENABLE` register.
- Operation:** SRXMASKBITCLR

Opcode: 0xE5

7	6	5	4	3	2	1	0
1	1	1	0	0	1	0	1

#### 23.14.9.37 ISTXON

- Function:** Enable TX after calibration
- Description:** The ISTXON instruction immediately enables TX after calibration. The instruction waits for the radio to acknowledge the command before executing the next instruction.
- Operation:** STXON\_STRB

**Opcode: 0xE9**

7	6	5	4	3	2	1	0
1	1	1	0	1	0	0	1

**23.14.9.38 ISTXONCCA**

**Function:** Enable calibration and TX if CCA indicates a clear channel

**Description:** The ISTXONCCA instruction immediately enables TX after calibration if CCA indicates a clear channel.

**Operation:** STXONCCA

**Opcode: 0xEA**

7	6	5	4	3	2	1	0
1	1	1	0	1	0	1	0

**23.14.9.39 ISSAMPLECCA**

**Function:** Sample the current CCA value to `SAMPLED_CCA`

**Description:** The current CCA value is immediately written to `SAMPLED_CCA` in XREG.

**Operation:** SSAMPLECCA

**Opcode: 0xEB**

7	6	5	4	3	2	1	0
1	1	1	0	1	0	1	1

**23.14.9.40 ISRFOFF**

**Function:** Disable RX/TX and frequency synthesizer.

**Description:** The ISRFOFF instruction immediately disables RX/TX and the frequency synthesizer.

**Operation:** `FFCTL_SRFOFF_STRB = 1`

**Opcode: 0xEF**

7	6	5	4	3	2	1	0
1	1	1	0	1	1	1	1

**23.14.9.41 ISFLUSHRX**

**Function:** Flush RXFIFO buffer and reset demodulator

**Description:** The ISFLUSHRX instruction immediately flushes the RXFIFO buffer and resets the demodulator.

**Operation:** SFLUSHRX

**Opcode: 0xED**

7	6	5	4	3	2	1	0
1	1	1	0	1	1	0	1

### 23.14.9.42 ISFLUSHTX

**Function:** Flush TXFIFO buffer  
**Description:** The ISFLUSHTX instruction immediately flushes the TXFIFO buffer.  
**Operation:** SFLUSHTX

Opcode: 0xEE

7	6	5	4	3	2	1	0
1	1	1	0	1	1	1	0

### 23.14.9.43 ISACK

**Function:** Send acknowledge frame with the pending field cleared  
**Description:** The ISACK instruction immediately sends an acknowledge frame.  
**Operation:** SACK

Opcode: 0xE6

7	6	5	4	3	2	1	0
1	1	1	0	0	1	1	0

### 23.14.9.44 ISACKPEND

**Function:** Send acknowledge frame with the pending field set  
**Description:** The ISACKPEND instruction immediately sends an acknowledge frame with the pending field set. The instruction waits for the radio to receive and interpret the command before executing the next instruction.  
**Operation:** SACKPEND

Opcode: 0xE7

7	6	5	4	3	2	1	0
1	1	1	0	0	1	1	1

### 23.14.9.45 ISNACK

**Function:** Abort sending of acknowledge frame  
**Description:** The ISNACK instruction immediately prevents sending of an acknowledge frame to the currently received frame.  
**Operation:** SNACK

Opcode: 0xE8

7	6	5	4	3	2	1	0
1	1	1	0	1	0	0	0

### 23.14.9.46 ISCLEAR

- Function:** Clear CSP program memory, reset program counter
- Description:** The ISCLEAR clears the program memory, resets the program counter, and aborts any running program. No stop interrupt is generated. The LABEL pointer is cleared. The ISCLEAR instruction must be issued twice to reset the program counter.
- Operation:** PC = 0, clear program memory

**Opcode: 0xFF**

<b>7</b>	<b>6</b>	<b>5</b>	<b>4</b>	<b>3</b>	<b>2</b>	<b>1</b>	<b>0</b>
1	1	1	1	1	1	1	1

## 23.15 Registers

**Table 23-5. Register Overview**

Address (Hex)	+ 0x000	+ 0x001	+ 0x002	+ 0x003
0x6180	FRMFILT0	FRMFILT1	SRCMATCH	SRCSHORTEN0
0x6184	SRCSHORTEN1	SRCSHORTEN2	SRCEXTEN0	SRCEXTEN1
0x6188	SRCEXTEN2	FRMCTRL0	FRMCTRL1	RXENABLE
0x618C	RXMASKSET	RXMASKCLR	FREQTUNE	FREQCTRL
0x6190	TXPOWER	TXCTRL	FSMSTAT0	FSMSTAT1
0x6194	FIFOPCTRL	FSMCTRL	CCACTRL0	CCACTRL1
0x6198	RSSI	RSSISTAT	RXFIRST	RXFIFOCNT
0x619C	TXFIFOCNT	RXFIRST_PTR	RXLAST_PTR	RXP1_PTR
0x61A0		TXFIRST_PTR	TXLAST_PTR	RFIRQM0
0x61A4	RFIRQM1	RFERRM	MONMUX	RFRND
0x61A8	MDMCTRL0	MDMCTRL1	FREQEST	RXCTRL
0x61AC	FSCTRL		FSCAL1	FSCAL2
0x61B0	FSCAL3	AGCCTRL0	AGCCTRL1	AGCCTRL2
0x61B4	AGCCTRL3	ADCTEST0	ADCTEST1	ADCTEST2
0x61B8	MDMTEST0	MDMTEST1	DACTEST0	DACTEST1
0x61BC	DACTEST2	ATEST	PTEST0	PTEST1
0x61C0	CSPPROG0	CSPPROG1	CSPPROG2	CSPPROG3
0x61C4	CSPPROG4	CSPPROG5	CSPPROG6	CSPPROG7
0x61C8	CSPPROG8	CSPPROG9	CSPPROG10	CSPPROG11
0x61CC	CSPPROG12	CSPPROG13	CSPPROG14	CSPPROG15
0x61D0	CSPPROG16	CSPPROG17	CSPPROG18	CSPPROG19
0x61D4	CSPPROG20	CSPPROG21	CSPPROG22	CSPPROG23
0x61D8				
0x61DC				
0x61E0	CSPCTRL	CSPSTAT	CSPX	CSPY
0x61E4	CSPZ	CSPT		
0x61E8				RFC_OBS_CTRL0
0x61EC	RFC_OBS_CTRL1	RFC_OBS_CTRL2		
0x61F0				
0x61F4				
0x61F8			TXFILTCFG	

### 23.15.1 Register Settings Update

This section contains a summary of the register settings that must be updated from their default value to have optimal performance.

The following settings should be set for both RX and TX. Although not all settings are necessary for both RX and TX, it is recommended for simplicity (allowing one set of settings to be written at the initialization of the code).

**Table 23-6. Registers That Require Update From Their Default Value**

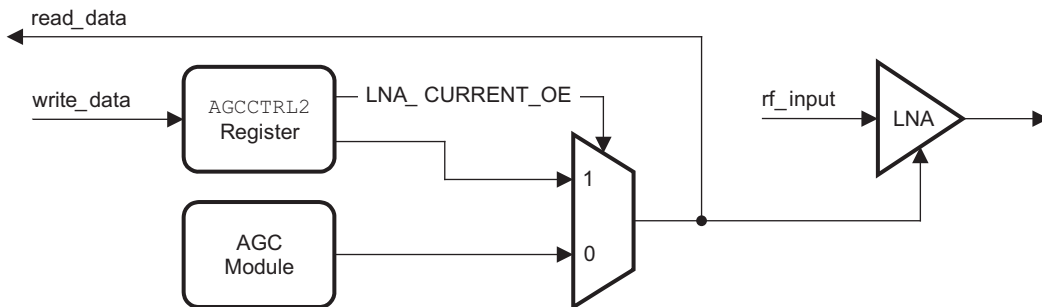
Register Name	New Value (Hex)	Description
AGCTRL1	0x15	Adjusts AGC target value
TXFILTCFG	0x09	Sets TX anti-aliasing filter to appropriate bandwidth
FSCAL1	0x00	Recommended setting for lowest spurious emission

### 23.15.2 Register Access Modes

The *Mode* column in [Table 23-7](#) shows what kind of accesses are allowed for each bit. The *Description* column gives the meaning of the different alternatives.

**Table 23-7. Register-Bit Access Modes**

Mode	Description
R	Read
W	Write
R0	Read constant zero
R1	Read constant one
W1	Only possible to write one
W0	Only possible to write zero
R*	The value read is not the actual register value, but rather the value seen by the module. This is typically used where a configuration value may be generated automatically (through calibration, dynamic control, and so forth) or manually overridden with a register value. An example structure is shown for the AGCTRL2 register in <a href="#">Figure 23-24</a> .



B0308-01

**Figure 23-24. Example Hardware Structure for the R\* Register Access Mode**



### 23.15.3 Register Descriptions

#### FRMFILT0 (0x6180) – Frame Filtering

Bit No.	Name	Reset	R/W	Description
7	–	0	R/W	Reserved. Always write 0
6:4	FCF_RESERVED_MASK[2:0]	000	R/W	Used for filtering on the reserved part of the frame control field (FCF). FCF_RESERVED_MASK[2:0] is ANDed with FCF[9:7]. If the result is nonzero and frame filtering is enabled, the frame is rejected.
3:2	MAX_FRAME_VERSION[1:0]	11	R/W	Used for filtering on the frame version field of the frame control field (FCF). If FCF[13:12] (the frame version subfield) is higher than MAX_FRAME_VERSION[1:0] and frame filtering is enabled, the frame is rejected.
1	PAN_COORDINATOR	0	R/W	Should be set high when the device is a PAN coordinator, to accept frames with no destination address (as specified in section 7.5.6.2 of IEEE 802.15.4(b)) 0: Device is not PAN coordinator. 1: Device is PAN coordinator.
0	FRAME_FILTER_EN	1	R/W	Enables frame filtering When this bit is set, the radio performs frame filtering as specified in section 7.5.6.2 of IEEE 802.15.4(b), third filtering level. FRMFILT0[6:1] and FRMFILT1[7:1], together with the local address information, define the behavior of the filtering algorithm. 0: Frame filtering off. (FRMFILT0[6:1], FRMFILT1[7:1] and SRCMATCH[2:0] are don't care.) 1: Frame filtering on.

#### FRMFILT1 (0x6181) – Frame Filtering

Bit No.	Name	Reset	R/W	Description
7	ACCEPT_FT_4TO7_RESERVED	0	R/W	Defines whether reserved frames are accepted or not. Reserved frames have frame type = 100, 101, 110, or 111. 0: Reject 1: Accept
6	ACCEPT_FT_3_MAC_CMD	1	R/W	Defines whether MAC command frames are accepted or not. MAC command frames have frame type = 011. 0: Reject 1: Accept
5	ACCEPT_FT_2_ACK	1	R/W	Defines whether acknowledgment frames are accepted or not. Acknowledgment frames have frame type = 010. 0: Reject 1: Accept
4	ACCEPT_FT_1_DATA	1	R/W	Defines whether data frames are accepted or not. Data frames have frame type = 001. 0: Reject 1: Accept
3	ACCEPT_FT_0_BEACON	1	R/W	Defines whether beacon frames are accepted or not. Beacon frames have frame type = 000. 0: Reject 1: Accept
2:1	MODIFY_FT_FILTER[1:0]	00	R/W	These bits are used to modify the frame-type field of a received frame before frame-type filtering is performed. The modification does not influence the frame that is written to the RXFIFO. 00: Leave as is 01: Invert MSB 10: Set MSB to 0 11: Set MSB to 1
0	–	0	R/W	Reserved. Always write 0

**SRCMATCH (0x6182) – Source Address Matching and Pending Bits**

Bit No.	Name	Reset	R/W	Description
7:3	–	0000 0	R/W	Reserved. Always write 0
2	PEND_DATAREQ_ONLY	1	R/W	When this bit is set, the AUTOPEND function also requires that the received frame is a <i>DATA REQUEST</i> MAC command frame.
1	AUTOPEND	1	R/W	Automatic acknowledgment pending flag enable. On reception of a frame, the pending bit in the (possibly) returned acknowledgment is set automatically, given that all the following conditions are met: <ul style="list-style-type: none"> <li>– FRMFILT0.FRAME_FILTER_EN is set.</li> <li>– SRCMATCH.SRC_MATCH_EN is set.</li> <li>– SRCMATCH.AUTOPEND is set.</li> <li>– The received frame matches the current SRCMATCH.PEND_DATAREQ_ONLY setting.</li> <li>– The received source address matches at least one source-match table entry, which is enabled in both SHORT_ADDR_EN and SHORT_PEND_EN or EXT_ADDR_EN and EXT_PEND_EN.</li> </ul> Note: Details for SHORT_PEND_EN and EXT_PEND_EN are found in the memory map description (Section 23.4).
0	SRC_MATCH_EN	1	R/W	Source address matching enable (This bit is don't care if FRMFILT0.FRAME_FILTER_EN = 0.)

**SRCSHORTEN0 (0x6183) – Short Address Matching**

Bit No.	Name	Reset	R/W	Description
7:0	SHORT_ADDR_EN[7:0]	0x00	R/W	The 7:0 part of the 24-bit word SHORT_ADDR_EN that enables or disables source address matching for each of the 24 short address table entries.  Optional safety feature: To ensure that an entry in the source-matching table is not used while it is being updated, set the corresponding SHORT_ADDR_EN bit to 0 while updating.

**SRCSHORTEN1 (0x6184) – Short Address Matching**

Bit No.	Name	Reset	R/W	Description
7:0	SHORT_ADDR_EN[15:8]	0x00	R/W	The 15:8 part of the 24-bit word SHORT_ADDR_EN. See previous description of SRCSHORTEN0.

**SRCSHORTEN2 (0x6185) – Short Address Matching**

Bit No.	Name	Reset	R/W	Description
7:0	SHORT_ADDR_EN[23:16]	0x00	R/W	The 23:16 part of the 24-bit word SHORT_ADDR_EN. See previous description of SRCSHORTEN0.

**SRCEXTEN0 (0x6186) – Extended Address Matching**

Bit No.	Name	Reset	R/W	Description
7:0	EXT_ADDR_EN[7:0]	0x00	R/W	The 7:0 part of the 24-bit word EXT_ADDR_EN that enables or disables source address matching for each of the 12 extended address table entries.  Write access: Extended address enable for table entry $n$ (0 to 7) is mapped to EXT_ADDR_EN[2n]. All EXT_ADDR_EN[2n + 1] bits are read-only and don't care when written to.  Read access: Extended address enable for table entry $n$ (0 to 7) is mapped to both EXT_ADDR_EN[2n] and EXT_ADDR_EN[2n + 1].  To ensure that an entry in the source matching table is not used while it is being updated, set the corresponding EXT_ADDR_EN bit to 0 while updating.

**SRCEXTEN1 (0x6187) – Extended Address Matching**

Bit No.	Name	Reset	R/W	Description
7:0	EXT_ADDR_EN[15:8]	0x00	R/W	The 15:8 part of the 24-bit word EXT_ADDR_EN. See previous description of SRCEXTEN0.

**SRCEXTEN2 (0x6188) – Extended Address Matching**

Bit No.	Name	Reset	R/W	Description
7:0	EXT_ADDR_EN[23:16]	0x00	R/W	The 23:16 part of the 24-bit word EXT_ADDR_EN. See previous description of SRCEXTEN0.

**FRMCTRL0 (0x6189) – Frame Handling**

Bit No.	Name	Reset	R/W	Description
7	APPEND_DATA_MODE	0	R/W	When AUTOCRC = 0: Don't care When AUTOCRC = 1: 0: RSSI + The CRC_OK bit and the 7-bit correlation value are appended at the end of each received frame 1: RSSI + The CRC_OK bit and the 7-bit SRCRESINDEX are appended at the end of each received frame. See <a href="#">Table 23-1</a> for details.
6	AUTOCRC	1	R/W	In TX 1: A CRC-16 (ITU-T) is generated in hardware and appended to the transmitted frame. There is no need to write the last 2 bytes to TXBUF. 0: No CRC-16 is appended to the frame. The last 2 bytes of the frame must be generated manually and written to TXBUF (if not, TX_UNDERFLOW occurs).  In RX 1: The CRC-16 is checked in hardware, and replaced in the RXFIFO by a 16-bit status word which contains a CRC OK bit. The status word is controllable through APPEND_DATA_MODE. 0: The last two bytes of the frame (CRC-16 field) are stored in the RXFIFO. The CRC check (if any) must be done manually. Note that this setting does not influence acknowledgment transmission (including AUTOACK).
5	AUTOACK	0	R/W	Defines whether the radio automatically transmits acknowledge frames or not. When autoack is enabled, all frames that are accepted by address filtering, have the acknowledge request flag set, and have a valid CRC are automatically acknowledged 12 symbol periods after being received. 0: Autoack disabled 1: Autoack enabled
4	ENERGY_SCAN	0	R/W	Defines whether the RSSI register contains the most-recent signal strength or the peak signal strength since the energy scan was enabled. 0: Most-recent signal strength 1: Peak signal strength
3:2	RX_MODE[1:0]	00	R/W	Set RX modes 00: Normal operation, use RXFIFO. 01: Reserved 10: RXFIFO looping ignores overflow in RXFIFO; infinite reception. 11: Same as normal operation except that symbol search is disabled. Can be used for RSSI or CCA measurements when it is not desired to find symbol.
1:0	TX_MODE[1:0]	00	R/W	Set test modes for TX 00: Normal operation, transmit TXFIFO 01: Reserved. Should not be used 10: TXFIFO looping ignores underflow in TXFIFO and reads cyclically; infinite transmission. 11: Send pseudorandom data from CRC, infinite transmission.

**FRMCTRL1 (0x618A) – Frame Handling**

Bit No.	Name	Reset	R/W	Description
7:3	–	0000 0	R0	Read as zero
2	PENDING_OR	0	R/W	Defines whether the pending data bit in outgoing acknowledgment frames is always set to 1 or controlled by the main FSM and the address filtering. 0: Pending data bit is controlled by main FSM and address filtering. 1: Pending data bit is always 1.
1	IGNORE_TX_UNDERF	0	R/W	Defines whether TX underflow should be ignored or not. 0: Normal TX operation. TX underflow is detected and TX is aborted if underflow occurs. 1: Ignore TX underflow. Transmit the number of bytes given by the frame-length field.
0	SET_RXENMASK_ON_TX	1	R/W	Defines whether STXON sets bit 6 in the RXENABLE register or leaves it unchanged. 0: Does not affect RXENABLE. 1: Sets bit 6 in RXENABLE. Used for backwards compatibility with the CC2420.

**RXENABLE (0x618B) – RX Enabling**

Bit No.	Name	Reset	R/W	Description
7:0	RXENMASK[7:0]	0x00	R	<p>RXENABLE enables the receiver. A nonzero value in this register causes the main FSM to enable the receiver when in idle, after transmission, and after acknowledgment transmission.</p> <p>The following strobos can modify RXENMASK:</p> <ul style="list-style-type: none"> <li>SRXON: Sets bit 7 in RXENMASK</li> <li>STXON: Sets bit 6 in RXENMASK if SET_RXENMASK_ON_TX = 1</li> <li>SRFOFF: Clears all bits in RXENMASK</li> <li>SRXMASKBITSET: Sets bit 5 in RXENMASK</li> <li>SRXMASKBITCLR: Clears bit 5 in RXENMASK</li> </ul> <p>RXENABLE can be modified directly by the CPU by accessing registers RXMASKSET and RXMASKCLR.</p> <p>There might be conflicts between the CSP and CPU operations if both try to modify RXENMASK simultaneously. To handle the case of simultaneous access to RXENMASK, the following rules apply:</p> <ul style="list-style-type: none"> <li>– If two sources are not in conflict (they modify different parts of the register), both their requests to modify RXENMASK are processed.</li> <li>– If both try to modify the mask simultaneously, bus-write operations to RXMASKSET and RXMASKCLR have priority over the CSP. It is strongly recommended to avoid this situation.</li> </ul>

**RXMASKSET (0x618C) – RX Enabling**

Bit No.	Name	Reset	R/W	Description
7:0	RXENMASKSET[7:0]	0x00	R0/W	When written, the written data is ORed with RXENMASK and stored in RXENMASK.

**RXMASKCLR (0x618D) – RX Disabling**

Bit No.	Name	Reset	R/W	Description
7:0	RXENMASKCLR[7:0]	0x00	R0/W	<p>When written, the written data is inverted and ANDed with RXENMASK and stored in RXENMASK.</p> <p>For example, if a 1 is written to one or more bit positions in this register, the corresponding bits are cleared in RXENMASK.</p>

**RFIRQM0 (0x61A3) – RF Interrupt Masks**

Bit No.	Name	Reset	R/W	Description
7:0	RFIRQM[7:0]	0x00	R/W	Bit mask masking out interrupt sources Bit position 7: RXMASKZERO 6: RXPKTDONE 5: FRAME_ACCEPTED 4: SRC_MATCH_FOUND 3: SRC_MATCH_DONE 2: FIFOP 1: SFD 0: ACT_UNUSED

**RFIRQM1 (0x61A4) – RF Interrupt Masks**

Bit No.	Name	Reset	R/W	Description
7:0	RFIRQM[14:8]	0x00	R/W	Bit mask masking out interrupt sources Bit position 7: Reserved 6: Reserved 5: CSP_WAIT 4: CSP_STOP 3: CSP_MANINT 2: RF_IDLE 1: TXDONE 0: TXACKDONE

**RFERRM (0x61A5) – RF Error Interrupt Mask**

Bit No.	Name	Reset	R/W	Description
7:0	RFERRM[7:0]	0x00	R/W	Bit mask masking out interrupt sources Bit position 7: Reserved 6: STROBE_ERR 5: TXUNDERF 4: TXOVERF 3: RXUNDERF 2: RXOVERF 1: RXABO 0: NLOCK

**FREQCTRL (0x618F) – Controls the RF Frequency**

Bit No.	Name	Reset	R/W	Description
7	–	0	R0	Read as zero
6:0	FREQ[6:0]	0x0B (2405 MHz)	R/W	Frequency control word $f_{RF} = f_{LO} + \text{FREQ}[6:0] \text{ MHz}$ The frequency word in FREQ[6:0] is an offset value from 2394. The device supports the frequency range from 2394 MHz to 2507 MHz. The usable settings for FREQ[6:0] are consequently 0 to 113. Settings outside this range (114–127) give a frequency of 2507 MHz. IEEE 802.15.4-2006 specifies a frequency range from 2405 MHz to 2480 MHz with 16 channels 5 MHz apart. The channels are numbered 11 through 26. For an IEEE 802.15.4-2006 compliant system, the only valid settings are thus $\text{FREQ}[6:0] = 11 + 5 \times (\text{channel number} - 11)$ .

**FREQTUNE (0x618E) – Crystal Oscillator Frequency Tuning**

Bit No.	Name	Reset	R/W	Description
7:4	–	0x0	R0	Read as zero
3:0	XOSC32M_TUNE[3:0]	0xF	R/W	Tune crystal oscillator The default setting, 1111, leaves the XOSC not tuned. Changing the setting from default switches in extra capacitance to the oscillator, effectively lowering the XOSC frequency. Hence, a higher setting gives a higher frequency.

**TXPOWER (0x6190) – Controls the Output Power**

Bit No.	Name	Reset	R/W	Description
7:0	PA_POWER [7:0]	0xF5	R/W	PA power control. <b>NOTE:</b> Before going to TX, this value should be updated. Please consult the data sheet of the device ( <a href="#">Appendix C</a> ) for recommended values; see also <a href="#">Section 23.8.13</a> .

**TXCTRL (0x6191) – Controls the TX Settings**

Bit No.	Name	Reset	R/W	Description
7	–	0	R0	Reserved
6:4	DAC_CURR[2:0]	110	R/W	Change the current in the DAC.
3:2	DAC_DC[1:0]	10	R/W	Adjusts the dc level to the TX mixer.
1:0	TXMIX_CURRENT[1:0]	01	R/W	Transmit mixers core current: current increases with increasing setting.

**FSMSTAT0 (0x6192) – Radio Status Register**

Bit No.	Name	Reset	R/W	Description
7	–	0	R	Reserved
6	CAL_RUNNING	0	R	Frequency synthesizer calibration status 0: Calibration done or not started 1: Calibration in progress
5:0	FSM_FFCTRL_STATE[5:0]	–	R	Gives the current state of the FIFO and frame-control (FFCTRL) finite-state machine.

**FSMSTAT1 (0x6193) – Radio Status Register**

Bit No.	Name	Reset	R/W	Description
7	FIFO	0	R	FIFO is high whenever there is data in the RXFIFO. Low during RXFIFO overflow
6	FIFOP	0	R	FIFOP is set high when there are more than FIFOP_THR bytes of data in the RXFIFO that have passed frame filtering. FIFOP is set high when there is at least one complete frame in the RXFIFO. FIFOP is set low again when a byte is read from the RXFIFO and this leaves FIFOP_THR bytes in the FIFO. FIFOP is high during RXFIFO overflow.
5	SFD	0	R	In TX 0: When a complete frame with SFD has been sent or no SFD has been sent 1: SFD has been sent. In RX 0: When a complete frame has been received or no SFD has been received 1: SFD has been received.
4	CCA	0	R	Clear-channel assessment. Dependent on CCA_MODE settings. See the following description of CCACTRL1.
3	SAMPLED_CCA	0	R	Contains a sampled value of the CCA. The value is updated whenever a SSAMPLECCA or STXONCCA strobe is issued.
2	LOCK_STATUS	0	R	1 when PLL is in lock, otherwise 0.
1	TX_ACTIVE	0	R	Status signal, active when FFCTRL is in one of the transmit states
0	RX_ACTIVE	0	R	Status signal, active when FFCTRL is in one of the receive states

**FIFOPCTRL (0x6194) – FIFOP Threshold**

Bit No.	Name	Reset	R/W	Description
7	–	0	R0	Read as zero
6:0	FIFOP_THR[6:0]	100 0000	R/W	Threshold used when generating FIFOP signal

**FSMCTRL (0x6195) – FSM Options**

Bit No.	Name	Reset	R/W	Description
7:2	–	0000 00	R0	Read as zero
1	SLOTTED_ACK	0	R/W	Controls timing of transmission of acknowledge frames 0: The acknowledge frame is sent 12 symbol periods after the end of the received frame which requests the acknowledge. 1: The acknowledge frame is sent at the first backoff-slot boundary more than 12 symbol periods after the end of the received frame which requests the acknowledge.
0	RX2RX_TIME_OFF	1	R/W	Defines whether or not a 12-symbol time-out should be used after frame reception has ended. 0: No time-out 1: 12-symbol-period time-out

**CCACTRL0 (0x6196) – CCA Threshold**

Bit No.	Name	Reset	R/W	Description
7:0	CCA_THR[7:0]	0xE0	R/W	<p>Clear-channel-assessment threshold value, signed 2s-complement number for comparison with the RSSI.</p> <p>The unit is 1 dB, offset is about 76 dBm. The CCA signal goes high when the received signal is below this value. The CCA signal is available on the CCA pin and in the FSMSTAT1 register.</p> <p>Note that the value should never be set lower than CCA_HYST – 128 in order to avoid erroneous behavior of the CCA signal.</p> <p><b>NOTE:</b> The reset value translates to an input level of approximately <math>-32 - 76 = -108</math> dBm, which is well below the sensitivity limit. That means the CCA signal never indicates a clear channel.</p> <p>This register should be updated to 0xF8, which translates to an input level of about <math>-8 - 76 = -84</math> dBm.</p>

**CCACTRL1 (0x6197) – Other CCA Options**

Bit No.	Name	Reset	R/W	Description
7:5	–	000	R0	Read as zero
4:3	CCA_MODE[1:0]	11	R/W	00: CCA always set to 1 01: CCA = 1 when $RSSI < CCA\_THR - CCA\_HYST$ ; CCA = 0 when $RSSI \geq CCA\_THR$ 10: CCA = 1 when not receiving a frame, else CCA = 0 11: CCA = 1 when $RSSI < CCA\_THR - CCA\_HYST$ and not receiving a frame; CCA = 0 when $RSSI \geq CCA\_THR$ or when receiving a frame
2:0	CCA_HYST[2:0]	010	R/W	Sets the level of CCA hysteresis. Unsigned values given in dB

**RSSI (0x6198) – RSSI Status Register**

Bit No.	Name	Reset	R/W	Description
7:0	RSSI_VAL[7:0]	0x80	R	<p>RSSI estimate on a logarithmic scale, signed number in 2s complement</p> <p>Unit is 1 dB. The offset to use in order to convert the register value into the real RSSI value can be found in the data sheet of the device (<a href="#">Appendix C</a>). The RSSI value is averaged over 8 symbol periods. The RSSI_VALID status bit should be checked before reading RSSI_VAL the first time.</p> <p>The reset value of –128 also indicates that the RSSI value is invalid.</p>

**RSSISTAT (0x6199) – RSSI Valid Status Register**

Bit No.	Name	Reset	R/W	Description
7:1	–	0000 000	R0	Read as zero
0	RSSI_VALID	0	R	RSSI value is valid. Occurs eight symbol periods after entering RX

**RXFIRST (0x619A) – First Byte in RXFIFO**

Bit No.	Name	Reset	R/W	Description
7:0	DATA[7:0]	0x00	R	<p>First byte of the RXFIFO.</p> <p>Note: Reading this register does not modify the contents of the FIFO.</p>

**RXFIFOCNT (0x619B) – Number of Bytes in RXFIFO**

Bit No.	Name	Reset	R/W	Description
7:0	RXFIFOCNT[7:0]	0x00	R	Number of bytes in the RXFIFO. Unsigned integer



**TXFIFOCNT (0x619C) – Number of Bytes in TXFIFO**

Bit No.	Name	Reset	R/W	Description
7:0	TXFIFOCNT[7:0]	0x00	R	Number of bytes in the TXFIFO. Unsigned integer

**RXFIRST\_PTR (0x619D) – RXFIFO Pointer**

Bit No.	Name	Reset	R/W	Description
7	–	0	R	Reserved
6:0	RXFIRST_PTR[6:0]	000 0000	R	RAM address offset of the first byte in the RXFIFO

**RXLAST\_PTR (0x619E) – RXFIFO Pointer**

Bit No.	Name	Reset	R/W	Description
7	–	0	R	Reserved
6:0	RXLAST_PTR[6:0]	000 0000	R	RAM address offset of the last byte +1 byte in the RXFIFO

**RXP1\_PTR (0x619F) – RXFIFO Pointer**

Bit No.	Name	Reset	R/W	Description
7:0	RXP1_PTR[7:0]	0x00	R	RAM address offset of the first byte of the first frame in the RXFIFO

**TXFIRST\_PTR (0x61A1) – TXFIFO Pointer**

Bit No.	Name	Reset	R/W	Description
7:0	TXFIRST_PTR[7:0]	0x00	R	RAM address offset of the next byte to be transmitted from the TXFIFO

**TXLAST\_PTR (0x61A2) – TXFIFO Pointer**

Bit No.	Name	Reset	R/W	Description
7:0	TXLAST_PTR[7:0]	0x00	R	RAM address offset of the last byte +1 byte of the TXFIFO

**MDMCTRL0 (0x61A8) – Controls Modem**

Bit No.	Name	Reset	R/W	Description
7:6	DEM_NUM_ZEROS[1:0]	10	R/W	Sets how many zero symbols must be detected before the sync word when searching for sync. Note that only one is required to have a correlation value above the correlation threshold set in the MDMCTRL1 register. 00: Reserved 01: 1 zero symbol 10: 2 zero symbols 11: 3 zero symbols
5	DEMOD_AVG_MODE	0	R/W	Defines the behavior of the frequency offset averaging filter. 0: Lock average level after preamble match. Restart frequency offset calibration when searching for the next frame. 1: Continuously update average level.
4:1	PREAMBLE_LENGTH [3:0]	0010	R/W	The number of preamble bytes (two zero-symbols) to be sent in TX mode prior to the SFD, encoded in steps of 2 symbols (1 byte). The reset value of 2 is compliant with IEEE 802.15.4. 0000: 2 leading-zero bytes 0001: 3 leading-zero bytes 0010: 4 leading-zero bytes ... .. 1111: 17 leading-zero bytes
0	TX_FILTER	1	R/W	Defines the kind of TX filter that is used. The normal TX filter is as defined by the IEEE802.15.4 standard. Extra filtering may be applied in order to lower the out-of-band emissions. 0: Normal TX filtering 1: Enable extra filtering

**MDMCTRL1 (0x61A9) – Controls Modem**

Bit No.	Name	Reset	R/W	Description
7:6	–	00	R0	Read as zero
5	CORR_THR_SFD	0	R/W	Defines requirements for SFD detection: 0: The correlation value of one of the zero symbols of the preamble must be above the correlation threshold. 1: The correlation value of one zero symbol of the preamble and both symbols in the SFD must be above the correlation threshold.
4:0	CORR_THR[4:0]	1 0100	R/W	Demodulator correlator threshold value, required before SFD search. Threshold value adjusts how the receiver synchronizes to data from the radio. If the threshold is set too low, sync can more easily be found on noise. If set too high, the sensitivity is reduced, but sync is not likely to be found on noise. In combination with DEM_NUM_ZEROS, the system can be tuned so sensitivity is high with less sync found on noise.

**FREQEST (0x61AA) – Estimated RF Frequency Offset**

Bit No.	Name	Reset	R/W	Description
7:0	FREQEST[7:0]	0x00	R	Signed 2s-complement value. Contains an estimate of the frequency offset between carrier and the receiver LO. The offset frequency is $FREQEST \times 7800$ Hz. DEM_AVG_MODE controls when this estimate is updated. If DEM_AVG_MODE = 0, it is updated until sync is found. Then the frequency offset estimate is frozen until the end of the received frame. If DEM_AVG_MODE = 1, it is updated as long as the demodulator is enabled. To calculate the correct value, one must use an offset (FREQEST_offset), which can be found in the data sheet of the device ( <a href="#">Appendix C</a> ). Real FREQEST value = $FREQEST - FREQEST\_offset$ .

**RXCTRL (0x61AB) – Tune Receive Section**

Bit No.	Name	Reset	R/W	Description
7:6	–	00	R0	Reserved
5:4	GBIAS_LNA2_REF[1:0]	11	R/W	Adjusts front-end LNA2 or mixer PTAT current output (from M = 3 to M = 6), default: M = 5
3:2	GBIAS_LNA_REF[1:0]	11	R/W	Adjusts front-end LNA PTAT current output (from M = 3 to M = 6), default: M = 5
1:0	MIX_CURRENT[1:0]	11	R/W	Control of the receiver mixers output current. The current increases with increasing setting.

**FSCTRL (0x61AC) – Tune Frequency Synthesizer**

Bit No.	Name	Reset	R/W	Description
7:6	PRE_CURRENT [1:0]	01	R/W	Prescaler current setting
5:4	LODIV_BUF_CURRENT_TX [1:0]	01	R/W	Adjusts current in mixer and PA buffers. Used when TX_ACTIVE = 1
3:2	LODIV_BUF_CURRENT_RX [1:0]	10	R/W	Adjusts current in mixer and PA buffers. Used when TX_ACTIVE = 0
1:0	LODIV_CURRENT [1:0]	10	R/W	Adjusts divider currents, except mixer and PA buffers.

**FSCAL1 (0x61AE) – Tune Frequency Calibration**

Bit No.	Name	Reset	R/W	Description
7:2	–	0010 10	R/W0	Reserved
1:0	VCO_CURR[1:0]	11	R/W	Defines current in VCO core. Sets the multiplier between calibrated current and VCO current. For the best value to use, see <a href="#">Table 23-6</a> in <a href="#">Section 23.15.1</a> .

**FSCAL2 (0x61AF) – Tune Frequency Calibration**

Bit No.	Name	Reset	R/W	Description
7	–	0	R0	Reserved. Read as 0
6	VCO_CAPARR_OE	0	R/W	Override the calibration result with the value from VCO_CAPARR[5:0].
5:0	VCO_CAPARR[5:0]	10 0000	R*/W	VCO capacitor array setting. Programmed during calibration. Override value when VCO_CAPARR_OE = 1.

**FSCAL3 (0x61B0) – Tune Frequency Calibration**

Bit No.	Name	Reset	R/W	Description
7	–	0	R0	Reserved. Read as 0
6	VCO_DAC_EN_OV	0	R/W	Enables the VCO DAC when 1
5:2	VCO_VC_DAC [3:0]	1010	R/W	Bit vector for programming varactor control voltage from VC DAC.
1:0	VCO_CAPARR_CAL_CTRL[1:0]	10	R/W	Calibration accuracy setting for the capacitor array part of the calibration 00: 80 XOSC periods 01: 100 XOSC periods 10: 125 XOSC periods 11: 250 XOSC periods

**AGCCTRL0 (0x61B1) – AGC Dynamic Range Control**

Bit No.	Name	Reset	R/W	Description
7	–	0	R0	Reserved. Read as 0
6	AGC_DR_XTND_EN	1	R/W	0: The AGC performs no adjustment of attenuation in the AAF. 1: The AGC adjusts the gain in the AAF to achieve extra dynamic range for the receiver.
5:0	AGC_DR_XTND_THR[5:0]	01 1111	R/W	If the measured error between the AGC reference magnitude and the actual magnitude in dB is larger than this threshold, the extra attenuation is enabled in the front end. This threshold should be set higher than 0x0C. This feature is enabled by AGC_DR_XTND_EN.

**AGCCTRL1 (0x61B2) – AGC Reference Level**

Bit No.	Name	Reset	R/W	Description
7:6	–	00	R0	Reserved. Read as 0
5:0	AGC_REF[5:0]	01 0001	R/W	Target value for the AGC control loop, given in 1-dB steps. For the best value to use see <a href="#">Table 23-6</a> in <a href="#">Section 23.15.1</a> .

**AGCCTRL2 (0x61B3) – AGC Gain Override**

Bit No.	Name	Reset	R/W	Description
7:6	LNA1_CURRENT[1:0]	00	R*/W	Override value for LNA 1. Only used when LNA_CURRENT_OE = 1. When read, this register returns the current applied gain setting. 00: 0-dB gain (reference level) 01: 3-dB gain 10: Reserved 11: 6-dB gain
5:3	LNA2_CURRENT[2:0]	000	R*/W	Override value for LNA 2. Only used when LNA_CURRENT_OE = 1. When read, this register returns the current applied gain setting. 000: 0-dB gain (reference level) 001: 3-dB gain 010: 6-dB gain 011: 9-dB gain 100: 12-dB gain 101: 15-dB gain 110: 18-dB gain 111: 21-dB gain
2:1	LNA3_CURRENT[1:0]	00	R*/W	Override value for LNA 3. Only used when LNA_CURRENT_OE = 1. When read, this register returns the current applied gain setting. 00: 0-dB gain (reference level) 01: 3-dB gain 10: 6-dB gain 11: 9-dB gain
0	LNA_CURRENT_OE	0	R/W	Write 1 to override the AGC LNA current setting with the values above (LNA1_CURRENT, LNA2_CURRENT, and LNA3_CURRENT).

**AGCCTRL3 (0x61B4) – AGC Control**

Bit No.	Name	Reset	R/W	Description
7	–	0	R0	Reserved. Read as 0
6:5	AGC_SETTLE_WAIT[1:0]	01	R/W	Time for AGC to wait for analog gain to settle after a gain change. During this period, the energy measurement in the AGC is paused. 00: 15 periods 01: 20 periods 10: 25 periods 11: 30 periods
4:3	AGC_WIN_SIZE[1:0]	01	R/W	Window size for the accumulate and dump function in the AGC 00: 16 samples 01: 32 samples 10: 64 samples 11: 128 samples
2:1	AAF_RP[1:0]	11	R*/W	Overrides the AGC control signals to AAF when AAF_RP_OE = 1. When read, it returns the applied signal to the AAF. 00: 9-dB attenuation in AAF 01: 6-dB attenuation in AAF 10: 3-dB attenuation in AAF 11: 0-dB attenuation in AAF (reference level)
0	AAF_RP_OE	0	R/W	Write 1 to override the AGC AAF control signals with the values stored in AAF_RP.

**ADCTEST0 (0x61B5) – ADC Tuning**

Bit No.	Name	Reset	R/W	Description
7:6	ADC_VREF_ADJ[1:0]	00	R/W	Quantizer threshold control for test and debug
5:4	ADC_QUANT_ADJ[1:0]	01	R/W	Quantizer threshold control for test and debug
3:1	ADC_GM_ADJ[2:0]	000	R/W	Gm control for test and debug
0	ADC_DAC2_EN	0	R/W	Enables DAC2 for enhanced ADC stability

**ADCTEST1 (0x61B6) – ADC Tuning**

Bit No.	Name	Reset	R/W	Description
7:4	ADC_TEST_CTRL[3:0]	0000	R/W	ADC test mode selector
3:2	ADC_C2_ADJ[1:0]	11	R/W	Used to adjust capacitor values in ADC
1:0	ADC_C3_ADJ[1:0]	10	R/W	Used to adjust capacitor values in ADC

**ADCTEST2 (0x61B7) – ADC Tuning**

Bit No.	Name	Reset	R/W	Description
7	–	0	R0	Reserved. Read as 0
6:5	ADC_TEST_MODE	00	R/W	Test mode to enable output of ADC data from demodulator. When enabled, raw ADC data is clocked out on the GPIO pins. 00: Test mode disabled 01: Data from both I and Q ADCs is output, data rate 76 MHz 10: Data from I ADC is output. Two and two ADC samples grouped, data rate 38 MHz 11: Data from Q ADC is output. Two and two ADC samples grouped, data rate 38 MHz
4:3	AAF_RS[1:0]	00	R/W	Controls series resistance of AAF
2:1	ADC_FF_ADJ[1:0]	01	R/W	Adjust feedforward
0	ADC_DAC_ROT	1	R/W	Control of DAC DWA scheme 0: DWA (scrambling) disabled 1: DWA enabled

**MDMTEST0 (0x61B8) – Test Register for Modem**

Bit No.	Name	Reset	R/W	Description
7:4	TX_TONE[3:0]	0111	R/W	<p>Enables the possibility to transmit a baseband tone by picking samples from the sine tables with a controllable phase step between the samples. The step size is controlled by TX_TONE. If MDMTEST1.MOD_IF is 0, the tone is superimposed on the modulated data, effectively giving modulation with an IF. If MDMTEST1.MOD_IF is 1, only the tone is transmitted.</p> <p>0000: –6 MHz            0001: –4 MHz            0010: –3 MHz            0011: –2 MHz            0100: –1 MHz            0101: –500 kHz            0110: –4 kHz            0111: 0            1000: 4 kHz            1001: 500 kHz            1010: 1 MHz            1011: 2 MHz            1100: 3 MHz            1101: 4 MHz            1110: 6 MHz            Others: Reserved</p>
3:2	DC_WIN_SIZE[1:0]	01	R/W	<p>Controls the number of samples to be accumulated between each dump of the accumulate-and-dump filter used in dc removal.</p> <p>00: 32 samples            01: 64 samples            10: 128 samples            11: 256 samples</p>
1:0	DC_BLOCK_MODE[1:0]	01	R/W	<p>Selects the mode of operation:</p> <p>00: The input signal to the dc blocker is passed on to the output without any attempt to remove dc.            01: Enable dc cancellation. Normal operation            10: Freeze estimates of dc when sync is found. Start estimating dc again when searching for the next frame.            11: Reserved</p>

**MDMTEST1 (0x61B9) – Test Register for Modem**

Bit No.	Name	Reset	R/W	Description
7:5	–	000	R0	Reserved. Read as 0
4	MOD_IF	0	R/W	<p>0: Modulation is performed at an IF set by MDMTEST0.TX_TONE.            1: A tone is transmitted with frequency set by MDMTEST0.TX_TONE.</p>
3	RAMP_AMP	1	R/W	<p>1: Enable ramping of DAC output amplitude during startup and finish.            0: Disable ramping of DAC output amplitude</p>
2	RFC_SNIFF_EN	0	R/W	<p>0: Packet sniffer module disabled            1: Packet sniffer module enabled. The received and transmitted data can be observed on GPIO pins.</p>
1	MODULATION_MODE	0	R/W	<p>Set one of two RF modulation modes for RX/TX</p> <p>0: IEEE 802.15.4 compliant mode            1: Reversed phase, non-IEEE compliant</p>
0	RESERVED	0	R/W	Reserved. Do not write.

**DACTEST0 (0x61BA) – DAC Override Value**

Bit No.	Name	Reset	R/W	Description
7	–	0	R0	Reserved. Read as 0
6:0	DAC_Q_O[6:0]	000 0000	R/W	Q-branch DAC override value when DAC_SRC = 001 If DAC_SRC is set to be ADC data, CORDIC magnitude, channel filtered data, then DAC_Q_O controls the part of the word in question that actually is muxed to the DAC as described in the following list. 00 0110 ≥ bits 6:0 00 0111 ≥ bits 7:1 00 1000 ≥ bits 8:2 And so on If an invalid setting is chosen, then the DAC outputs only zeros (minimum value).

**DACTEST1 (0x61BB) – DAC Override Value**

Bit No.	Name	Reset	R/W	Description
7	–	0	R0	Reserved. Read as 0
6:0	DAC_I_O[6:0]	000 0000	R/W	I-branch DAC override value when DAC_SRC = 001 If DAC_SRC is set to be ADC data, CORDIC magnitude, channel filtered data, then DAC_I_O controls the part of the word in question that actually is muxed to the DAC as described in the following list. 00 0110 ≥ bits 6:0 00 0111 ≥ bits 7:1 00 1000 ≥ bits 8:2 And so on If an invalid setting is chosen, then the DAC outputs only zeros (minimum value),

**DACTEST2 (0x61BC) – DAC Test Setting**

Bit No.	Name	Reset	R/W	Description
7:3	–	0010 1	R0	Reserved
2:0	DAC_SRC[2:0]	000	R/W	The data source for the TX DAC is selected by DAC_SRC according to: 000: Normal operation (from modulator). 001: The DAC_I_O and DAC_Q_O override values 010: ADC data after decimation, magnitude-controlled by DAC_I_O and DAC_Q_O 011: I/Q after decimation, channel and dc filtering, magnitude-controlled by DAC_I_O and DAC_Q_O 100: Cordic magnitude output and front-end gain is output, magnitude-controlled by DAC_I_O and DAC_Q_O 101: RSSI I output on the I DAC 111: Reserved

**AATEST (0x61BD) – Analog Test Control**

Bit No.	Name	Reset	R/W	Description
7:6	–	00	R0	Reserved. Read as 0
5:0	AATEST_CTRL[5:0]	00 0000	R/W	Controls the analog test mode: 00 0000: Disabled 00 0001: Enables the temperature sensor (see also the TR0 register description in <a href="#">Section 12.2.10</a> ). 00 0010 : Enables the temperature sensor in the CC2533 (see also the TR0 register description in <a href="#">Section 12.2.10</a> ) Other values reserved.

**RFRND (0x61A7) – Random Data**

Bit No.	Name	Reset	R/W	Description
7:2	–	0000 00	R0	Reserved. Read as 0
1	QRND	0	R0	Random bit from the Q channel of the receiver
0	IRND	0	R0	Random bit from the I channel of the receiver

**PTEST0 (0x61BE) – Override Power-Down Register**

Bit No.	Name	Reset	R/W	Description
7	PRE_PD	0	R/W	Prescaler power-down signal when PD_OVERRIDE = 1
6	CHP_PD	0	R/W	Charge-pump power-down signal when PD_OVERRIDE = 1
5	ADC_PD	0	R/W	Analog-to-digital converter power-down signal when PD_OVERRIDE = 1
4	DAC_PD	0	R/W	Digital-to-analog converter power-down signal when PD_OVERRIDE = 1
3:2	LNA_PD[1:0]	00	R/W	Low-noise amplifier power-down signal. Defines LNA and mixer PD modes. 00: Power up 01: LNA off, mixer and regulator on 10: LNA and mixer off, regulator on 11: Power down When PD_OVERRIDE = 1
1	TXMIX_PD	0	R/W	Transmit mixer power-down signal when PD_OVERRIDE = 1
0	AAF_PD	0	R/W	Antialiasing filter power-down signal when PD_OVERRIDE = 1

**PTEST1 (0x61BF) – Override Power-Down Register**

Bit No.	Name	Reset	R/W	Description
7:4	–	0000	R0	Reserved. Read as 0
3	PD_OVERRIDE	0	R/W	Override enabling and disabling of various modules. For debug and testing only. It is impossible to override hard-coded BIAS_PD[1:0] dependency.
2	PA_PD	0	R/W	Power amplifier power-down signal when PD_OVERRIDE = 1
1	VCO_PD	0	R/W	Voltage-controlled oscillator power-down signal when PD_OVERRIDE = 1
0	LODIV_PD	0	R/W	LO power-down signal when PD_OVERRIDE = 1



**RFC\_OBS\_CTRL0 (0x61EB) – RF Observation Mux Control**

Bit No.	Name	Reset	R/W	Description
7	–	0	R0	Reserved. Read as 0
6	RFC_OBS_POL0	0	R/W	The signal chosen by RFC_OBS_MUX0 is XORed with this bit.
5:0	RFC_OBS_MUX0	00 0000	R/W	Controls which observable signal from RF Core is to be muxed out to rfc_obs_sigs[0] 00 0000: 0 – Constant value 00 0001: 1 – Constant value 00 1000: rfc_sniff_data – Data from packet sniffer. Sample data on rising edges of sniff_clk. 00 1001: rfc_sniff_clk – 250-kHz clock for packet sniffer data 00 1100: rssi_valid – Pin is high when the RSSI value has been updated at least once since RX was started. Cleared when leaving RX. 00 1101: demod_cca – Clear channel assessment. See FSMSTAT1 register for details on how to configure the behavior of this signal. 00 1110: sampled_cca – A sampled version of the CCA bit from demodulator. The value is updated whenever a SSAMPLECCA or STXONCCA strobe is issued. 00 1111: sfd_sync – Pin is high when an SFD has been received or transmitted. Cleared when leaving RX or TX, respectively. Not to be confused with the SFD exception. 01 0000: tx_active – Indicates that FFCTRL is in one of the TX states. Active-high. Note: This signal might have glitches, because it has no output flip-flop and is based on the current state register of the FFCTRL FSM. 01 0001: rx_active – Indicates that FFCTRL is in one of the RX states. Active-high. Note: This signal might have glitches, because it has no output flip-flop and is based on the current state register of the FFCTRL FSM. 01 0010: fctrl_fifo – Pin is high when one or more bytes are in the RXFIFO. Low during RXFIFO overflow. 01 0011: fctrl_fifop – Pin is high when the number of bytes in the RXFIFO exceeds the programmable threshold or at least one complete frame is in the RXFIFO. Also high during RXFIFO overflow. Not to be confused with the FIFOP exception. 01 0100: packet_done – A complete frame has been received. That is, the number of bytes set by the frame-length field has been received. 01 0110: rfc_xor_rand_i_q – XOR between I and Q random outputs. Updated at 8 MHz. 01 0111: rfc_rand_q – Random data output from the Q channel of the receiver. Updated at 8 MHz. 01 1000: rfc_rand_i – Random data output from the I channel of the receiver. Updated at 8 MHz 01 1001: lock_status – 1 when PLL is in lock, otherwise 0 10 1000: pa_pd – Power amplifier power-down signal 10 1010: lna_pd – LNA power-down signal Others: Reserved

**RFC\_OBS\_CTRL1 (0x61EC) – RF Observation Mux Control**

Bit No.	Name	Reset	R/W	Description
7	–	0	R0	Reserved. Read as 0
6	RFC_OBS_POL1	0	R/W	The signal chosen by RFC_OBS_MUX1 is XORed with this bit.
5:0	RFC_OBS_MUX1	00 0000	R/W	Controls which observable signal from RF Core is to be muxed out to rfc_obs_sigs[1]. See description of RFC_OBS_CTRL0 for details.

**RFC\_OBS\_CTRL2 (0x61ED) – RF Observation Mux Control**

Bit No.	Name	Reset	R/W	Description
7	–	0	R0	Reserved. Read as 0
6	RFC_OBS_POL2	0	R/W	The signal chosen by RFC_OBS_MUX2 is XORed with this bit.
5:0	RFC_OBS_MUX2	00 0000	R/W	Controls which observable signal from RF Core is to be muxed out to rfc_obs_sigs[2]. See description of RFC_OBS_CTRL0 for details.

**TXFILTCFG (0x61FA) – TX Filter Configuration**

Bit No.	Name	Reset	R/W	Description
7:4	–	0000	R0	Reserved
3:0	FC	1111	R/W	Sets TX anti-aliasing filter to appropriate bandwidth. Reduces spurious emissions close to signal. For the best value to use, see <a href="#">Table 23-6</a> in <a href="#">Section 23.15.1</a> .

**IVCTRL (0x6265) – Analog control register (CC2533 only)**

Bit No.	Name	Reset	R/W	Description
7:6	–	00	R0	Reserved. Always read as 0.
5:4	DAC_CURR_CTRL	01	R/W	Controls bias current to DAC 00: 100% IVREF, 0% IREF bias 01: 60% IVREF, 40% IREF bias 10: 40% IVREF, 60% IREF bias 11: 0% IVREF, 100% IREF bias
3	LODIV_BIAS_CTRL	0	R/W	Controls bias current to LODIV 1: PTAT bias 0: IVREF bias
2	TXMIX_DC_CTRL	0	R/W	Controls dc bias in TXMIX
1:0	PA_BIAS_CTRL	11	R/W	Controls bias current to PA 00: IREF bias 01: IREF and IVREF bias 10: PTAT bias 11: Increased PTAT slope bias

---

---

## CC2540 and CC2541 Bluetooth low energy Radio

---

---

The CC2540 and CC2541 provide a *Bluetooth* low energy compliant radio transceiver. On the CC2540 and CC2541, radio operation is controlled by the *Bluetooth* low energy stack. The application is not allowed to access the radio directly. The application interacts with the radio by sending API commands to the stack. The TI BLE stack with documentation is available at [www.ti.com/blestack](http://www.ti.com/blestack). The CC2541 may also be run in proprietary mode; see [Chapter 25](#) for a description of the operation in that case.

Topic	Page
24.1 Registers .....	276

## 24.1 Registers

The following status registers are available to the user:

### RFSTAT (0x618D) RF Core Status

Bit	Name	Reset	R/W	Description
7	MOD_UNDERFLOW	0	R/W0	Modulator has underflowed. Must be cleared by software
6:5	DEM_STATUS	00	R	Demodulator status 00: Idle 01: Active 10: Finishing 11: Error
4	SFD	0	R	High when the access address has been sent in TX or when sync has been obtained in RX
3	CAL_RUNNING	0	R	Frequency synth calibration status 0: Calibration done or not started 1: Calibration in progress
2	LOCK_STATUS	0	R	1 when PLL is in lock, otherwise 0
1	TX_ACTIVE	0	R	Status signal, active when radio is in transmit state
0	RX_ACTIVE	0	R	Status signal, active when radio is in receive state

### RFC\_OBS\_CTRL0 (0x61AE) RF Observation Mux Control 0

Bit	Name	Reset	R/W	Description
7	–	0	R0	Reserved. Read as 0
6	RFC_OBS_POL0	0	R/W	The signal chosen by RFC_OBS_MUX0 is XORed with this bit.
5:0	RFC_OBS_MUX0	00 0000	R/W	Controls which observable signal from rf_core is to be muxed out to rfc_obs_sigs(0). 00 0000: 0 – Constant value 00 0001: 1 – Constant value 00 1001: TX_active 00 1010: RX_active 11 0000: High from when receiver has found access address until packet is finished, low otherwise 11 0001: High from when the access address has been transmitted until end of packet, low otherwise Other values reserved

### RFC\_OBS\_CTRL1 (0x61AF) RF Observation Mux Control 1

Bit	Name	Reset	R/W	Description
7	–	0	R0	Reserved. Read as 0
6	RFC_OBS_POL1	0	R/W	The signal chosen by RFC_OBS_MUX1 is XORed with this bit.
5:0	RFC_OBS_MUX1	00 0000	R/W	Controls which observable signal from rf_core is to be muxed out to rfc_obs_sigs(1). 00 0000: 0 – Constant value 00 0001: 1 – Constant value 00 1001: TX_active 00 1010: RX_active 11 0000: High from when receiver has found access address until packet is finished, low otherwise 11 0001: High from when the access address has been transmitted until end of packet, low otherwise Other values reserved

**RFC\_OBS\_CTRL2 (0x61B0) RF Observation Mux Control 2**

Bit	Name	Reset	R/W	Description
7	–	0	R0	Reserved. Read as 0
6	RFC_OBS_POL2	0	R/W	The signal chosen by RFC_OBS_MUX2 is XORed with this bit.
5:0	RFC_OBS_MUX2	00 0000	R/W	Controls which observable signal from rf_core is to be muxed out to rfc_obs_sigs(2). 00 0000: 0 – Constant value 00 0001: 1 – Constant value 00 1001: TX active 00 1010: RX_active 11 0000: High from when receiver has found access address until packet is finished, low otherwise 11 0001: High from when the access address has been transmitted until end of packet, low otherwise Other values reserved

**AATEST (0x61A9) – Analog Test Control**

Bit	Name	Reset	R/W	Description
7:6	–	00	R0	Reserved. Read as 0
5:0	AATEST_CTRL[5:0]	00 0000	R/W	Controls the analog test mode: 00 0000: Disabled 00 0001: Enables the temperature sensor (see also the TR0 register description in <a href="#">Section 12.2.10</a> ). Other values reserved.

## CC2541 Proprietary Mode Radio

In proprietary mode, the CC2541 radio supports data rates up to 2 Mbps, and has extensive baseband automation, including auto-acknowledgment and address decoding. The **RF Core** controls the analog radio module and the RF transceiver state. In addition, it provides an interface between the MCU and the radio which makes it possible to issue commands, read status, and automate and sequence radio events.

It has 1 KB of dedicated RAM, which holds the 128-byte transmit and receive FIFO.

This chapter describes the proprietary mode operation of the CC2541 devices and features in the LLE program. For *Bluetooth* low energy operation, see [Chapter 24](#).

Topic	Page
25.1 RF Core .....	279
25.2 Interrupts .....	279
25.3 RF Core Data Memory .....	280
25.4 Bit-Stream Processor .....	291
25.5 Frequency and Channel Programming .....	296
25.6 Modulation Formats .....	296
25.7 Receiver .....	296
25.8 Packet Format .....	297
25.9 Link Layer Engine.....	301
25.10 Random Number Generation .....	318
25.11 Packet Sniffing.....	319
25.12 Registers.....	320

## 25.1 RF Core

The **RF core** contains several submodules that support and control the analog radio modules. In addition, it provides an interface between the MCU and the radio which makes it possible to issue commands, read status, and automate and sequence radio events.

The **link-layer engine** (LLE) controls the RF transceiver state and most of the dynamically controlled analog signals such as power up and power down of analog modules. The LLE is used to provide the correct sequencing of events (such as performing an FS calibration before enabling the receiver). It handles packet assembly and decoding, including automatic length field handling, address insertion and filtering, and CRC generation and checking.

The **radio data RAM** holds a FIFO for transmit data (TX FIFO) and a FIFO for receive data (RX FIFO). Both FIFOs are 128 bytes long and have hardware control of pointers when data is entered and removed from the FIFOs. In addition, the RAM contains six segments of 128 bytes, one of which is used for communication with the LLE.

The **bit-stream processor** is used for whitening and de-whitening transferred signals and CRC generation and check.

The **modulator** transforms raw data into I/Q signals to the transmitter DAC.

The **demodulator** is responsible for retrieving the over-the-air data from the received signal.

The **frequency synthesizer (FS)** generates the carrier wave for the RF signal.

## 25.2 Interrupts

The radio is associated with two **interrupt** vectors on the CPU. These are the RFERR interrupt (interrupt 0) and the RF interrupt (interrupt 12) with the following functions.

- RFERR: Error situations in the radio are signaled using this interrupt.
- RF: Interrupts coming from normal operation are signaled using this interrupt.

The RF interrupt vector combines the interrupts in `RFIF`. Note that these RF interrupts are rising-edge triggered. Thus, an interrupt is generated when, for example, the TASKDONE status flag in the `RFIRQF1` register goes from 0 to 1. The `RFIF` interrupt flags are described in [Section 25.2.1](#).

### 25.2.1 Interrupt Registers

Two main interrupt-control SFR registers are used to enable the RF and RFERR interrupts. These are the following:

- RFERR: `IEN0.RFERRIE`
- RF: `IEN2.RFIE`

Two main interrupt-flag SFR registers hold the RF and RFERR interrupt flags. These are the following:

- RFERR: `TCON.RFERRIF`
- RF: `S1CON.RFIF`

The two interrupts generated from the RF core are a combination of several sources within the RF core. Each of the individual sources has its own enable and interrupt flags in RF core. Flags can be found in `RFIRQF0`, `RFIRQF1`, and `RFERRF`. Interrupt enable masks can be found in `RFIRQM0`, `RFIRQM1`, and `RFERRM`.

The interrupt enable bits in the mask registers are used to enable individual interrupt sources. Note that masking an interrupt source does not affect the updating of the corresponding status in the flag registers.

Due to the use of individual interrupt masks in the RF core, the interrupts coming from the RF core have two-layered masking, and care must be taken when processing these interrupts. The procedure is described as follows.

To clear an interrupt from the RF core, one must clear two flags, both the flag set in the RF core and the one set in the main interrupt flag SFR registers, `S1CON` or `TCON` (depending on which interrupt is triggered). If a flag is cleared in the RF core and there are other unmasked flags standing, the main interrupt flag is set. Exiting the interrupt service routine with the main interrupt flag set causes the interrupt service routine to be executed again.

TIP: For proper handling of interrupts in ISRs, the following is advised:

- At the start of the ISR, read and store the RF core flags
- Process the interrupts
- Clear the main interrupt flag
- Clear the processed RF core flags. It is important that this is done in a single operation.

### 25.3 RF Core Data Memory

The radio core has 1024 bytes of data RAM divided into eight pages of 128 bytes each. The pages are to be used as shown in [Table 25-1](#).

**Table 25-1. Radio RAM Pages**

Page Number	Assignment
0	RAM-based registers
1	For RX with auto ACK: ACK payload FIFO for addresses 2 and 3
2	For RX with auto ACK: ACK payload FIFO for addresses 4 and 5
3	For RX with auto ACK: ACK payload FIFO for addresses 6 and 7
4	Free for MCU use
5	Additional RAM-based registers. Reserved for LLE
6	RX FIFO
7	TX FIFO; for RX with auto ACK: ACK payload FIFO for addresses 0 and 1

The active memory page is selected in register `RFRAMCFG.PRE`. The selected page is accessible at XDATA addresses 0x6000–0x607F. The RX FIFO page (page 6) is also accessible at XDATA addresses 0x6080–0x60FF. The TX FIFO page (page 7) is also accessible at XDATA addresses 0x6100–0x617F.

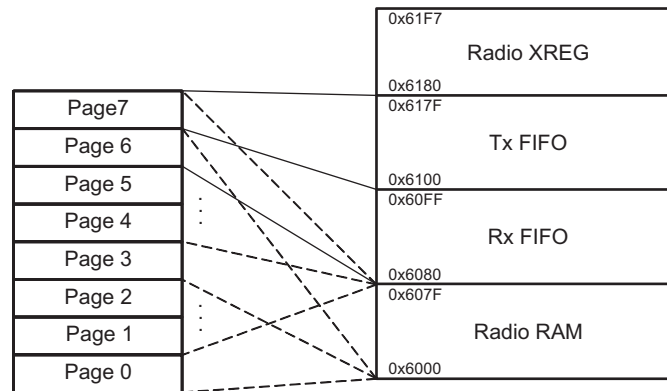
A page is used for transferring parameters to the LLE; see [Section 25.3.3](#).

There is no hardware protection to prevent the MCU from overwriting memory used by the LLE and the FIFO. Thus, the MCU should never write to page 5 (except for special dedicated registers). The MCU should write to pages 0, 1, 2, 3, and 7 only as specified in this chapter. Writes to the FIFO pages should only be done in ways compatible with FIFO operation, except for accessing the TX FIFO page while running an RX task with auto ACK.

Pages 0, 1, 6, and 7 have retention in all power modes, whereas the contents of pages 2–5 are lost in PM2 and PM3.

Radio core hardware registers are located at XDATA addresses 0x6180–0x61F7. [Figure 25-1](#) shows the mapping of radio memory to MCU XDATA memory space.





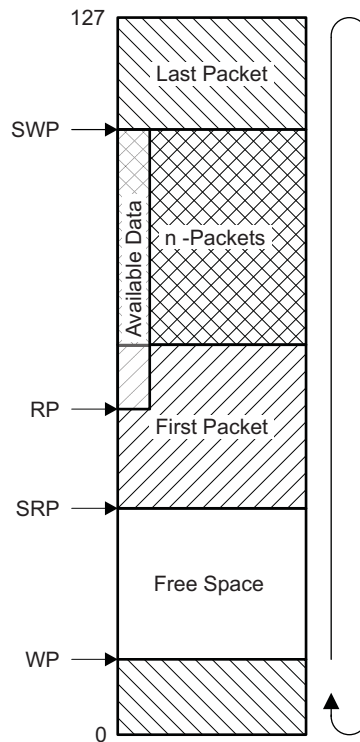
M0219-01

Figure 25-1. Mapping of Radio Memory to MCU XDATA Memory Space

### 25.3.1 FIFOs

The FIFOs are used for transporting data between the MCU and the radio. The FIFOs have hardware support for read and write pointer increment with circular buffering, overflow and underflow detection, and flushing of last entry or the entire FIFO.

The RX and TX FIFOs are fundamentally two similar modules. Each FIFO has four pointers: the write pointer (WP), the read pointer (RP), the start-of-packet write pointer (SWP), and the start-of-packet read pointer (SRP). WP and RP give the index in the FIFO where the next byte is to be written and read, respectively. SWP is used to indicate the start of the current packet being written, and SRP is used to indicate the start of the current packet being read. The use of the pointers is indicated in Figure 25-2.



M0220-01

Figure 25-2. FIFO Pointers

The TX FIFO and RX FIFO may be accessed through the SFR register `RFD` (0xD9). Data is written to the TX FIFO when writing to the `RFD` register. Data is read from the RX FIFO when the `RFD` register is read. In addition, there are separate read and write registers for each FIFO (`RFRXFRD`, `RFRXFWR`, `RFTXFRD`, `RFTXFWR`).

The RX FIFO or TX FIFO can be cleared by issuing `CMD_RXFIFO_RESET` or `CMD_TXFIFO_RESET` (see [Section 25.3.1.2](#)), respectively. The contents of both FIFOs can be cleared by issuing `CMD_FIFO_RESET`.

Four operations are defined to handle the four pointers:

- *Deallocate* is setting `SRP` equal to `RP`. This should be done when the treatment of a packet that has been read from the FIFO is finished.
- *Retry* is setting `RP` equal to `SRP`. This is done to re-read a packet that has been read from the FIFO previously.
- *Discard* is setting `WP` equal to `SWP`. This is done to remove a packet that had been written to the FIFO.
- *Commit* is setting `SWP` equal to `WP`. This is done to confirm the writing of a packet to the FIFO and making it available to be read out.

Using the register `RFFCFG`, it is possible to set up auto-commit and auto-deallocate for each of the FIFOs. If auto-commit is enabled, `SWP` is set equal to `WP` each time a byte is written to the FIFO. If auto-deallocate is enabled, `SRP` is set equal to `RP` each time a byte is read from the FIFO. By default, auto-commit is enabled for the TX FIFO and auto-deallocate is enabled for the RX FIFO. This is also the recommended setting. However, if packets that exceed the FIFO size are to be supported, auto-commit must be enabled for the RX FIFO and auto-deallocate for the TX FIFO; see [Section 25.8.1](#) and [Section 25.8.2](#) for details. If auto-commit is disabled for the TX FIFO, the MCU must issue a commit command after writing a packet to the TX FIFO, and if auto-deallocate is disabled for the RX FIFO, the MCU must issue a deallocate command after reading a packet from the RX FIFO.

### 25.3.1.1 FIFO Status and Interrupts

The XREG registers `RFRXFLEN` and `RFTXFLEN` provide information on the amount of data in the FIFOs. This is the number of bytes between `SRP` and `WP`, that is, the number of bytes that is not free space in [Figure 25-2](#). The register `RFFSTATUS` contains status bits for each of the FIFOs. FIFO empty is defined as the length being 0, and FIFO full is defined as the length being 128. The amount of data between `RP` and `SWP` is known as available data, and there is a status bit in the `RFFSTATUS` register telling whether there is available data for each of the FIFOs.

An attempt to write to a full FIFO results in a FIFO overflow. The data written is then ignored and the `RXOVERF` or `TXOVERF` flag is set in the `RFERRF` register, causing an `RFERR` interrupt. An attempt to read from a FIFO when no data is available results in a FIFO underflow. The value read is then zero, and the `RXUNDERF` or `TXUNDERF` flag is set in the `RFERRF` register, causing an `RFERR` interrupt.

Registers `RFTXFTHRS` and `RFRXFTHRS` are used to set threshold points for the TX and RX FIFOs, respectively. Each FIFO has one status flag and two interrupt flags; when the amount of data in the FIFO crosses the threshold, an interrupt flag is set. The FIFO status flags are available in `RFFSTATUS`, and the interrupt flags are available in `RFIRQF0`.

When the amount of data in the FIFO is above the threshold, that is, `RFxXFLEN` is greater than or equal to `RFxXFTHRS`, the status bit `xXDTHEX` of `RFFSTATUS` is 1, otherwise it is 0.

When data is written to the FIFO causing the FIFO threshold to be crossed, that is, `xXDTHEX` going from 0 to 1, the corresponding interrupt flag is set.

When data is read from the FIFO causing the FIFO threshold to be crossed, that is, `xXDTHEX` going from 1 to 0, the corresponding interrupt flag is set.

### 25.3.1.2 Command Register

The command register `RFST` can be used for sending commands to the FIFO. Commands in the range 0x80–0xFF are commands to the FIFO. Other commands are commands to the LLE; see [Section 25.9.1](#).

The supported FIFO commands are listed in [Table 25-2](#). A command in the range of 0x80–0xFF that does not match any of the listed commands is ignored.

**Table 25-2. Commands to FIFO via RFST Register**

Number	Command Name	Description
0x81	CMD_RXFIFO_RESET	Reset (empty) RX FIFO. Set RFRXF* := 0
0x82	CMD_RXFIFO_DEALLOC	Deallocate RX FIFO. This sets RFRXFSRP := RFRXFRP.
0x83	CMD_RXFIFO_RETRY	Retry RX FIFO. This sets RFRXFRP := RFRXFSRP
0x84	CMD_RXFIFO_DISCARD	Discard RX FIFO. This sets RFRXFWP := RFRXFSWP
0x85	CMD_RXFIFO_COMMIT	Commit RX FIFO. This sets RFRXFSWP := RFRXFWP
0x91	CMD_TXFIFO_RESET	Reset (empty) TX FIFO. Set RFRXF* := 0
0x92	CMD_TXFIFO_DEALLOC	Deallocate TX FIFO. This sets RFTXFSRP := RFTXFRP.
0x93	CMD_TXFIFO_RETRY	Retry TX FIFO. This sets RFTXFRP := RFTXFSRP
0x94	CMD_TXFIFO_DISCARD	Discard TX FIFO. This sets RFTXFWP := RFTXFSWP
0x95	CMD_TXFIFO_COMMIT	Commit TX FIFO. This sets RFTXFSWP := RFTXFWP
0xF1	CMD_FIFO_RESET	Reset both FIFOs
0xF2	CMD_FIFO_DEALLOC	Deallocate both FIFOs
0xF3	CMD_FIFO_RETRY	Retry both FIFOs
0xF4	CMD_FIFO_DISCARD	Discard both FIFOs
0xF5	CMD_FIFO_COMMIT	Commit both FIFOs

### 25.3.1.3 FIFO Pointer Operations

The FIFO pointers can be accessed directly through registers RFFRXP, RFFRXFRP, RFFRXFSWP, RFFRXFSRP, RFTXFWP, RFTXFRP, RFTXFSWP, and RFTXFSRP.

Because the placement of the pointers may be the same for an empty and a full FIFO, there are internal states distinguishing between these situations. This means that although any value can be written to the pointer registers, certain rules must be observed for the FIFO to function reliably after the pointer write.

Any writes to a pointer must be considered to move that pointer up. Hence, writing N to a pointer already holding N is considered equivalent to moving that pointer up 128 places, writing N-1 is equivalent to moving the pointer up 127 places, and so on.

The pointers must maintain a specific ordering: (Going from lowest position to highest) SRP, RP, SWP, WP.

A lower pointer may be moved up to but not past a higher pointer, whereas the highest pointer (WP) may be moved down to, but not past the lower.

### 25.3.1.4 Cooperation With LLE

The LLE performs FIFO operations as part of its operation. In order to avoid conflicts between the LLE and the MCU, access to FIFO registers should be done according to [Table 25-3](#). Read accesses can always be made, except for the data-read registers, which causes the read pointers to be modified. If the MCU reads a register, one must take into account that the value may change at any time due to accesses from the LLE. The reset FIFO commands should only be run by the MCU between LLE tasks. They are marked with an asterisk in [Table 25-3](#).

**Table 25-3. Access to FIFO Registers**

Register	Read Access	Write Access
RFD	MCU	MCU

**Table 25-3. Access to FIFO Registers (continued)**

Register	Read Access	Write Access
RFFST (FIFO commands)	N/A	Depends on command: 0x81 : Reset RX FIFO: MCU* 0x82 : Deallocate RX FIFO: MCU 0x83 : Retry RX FIFO: MCU 0x84 : Discard RX FIFO: LLE 0x85 : Commit RX FIFO: LLE 0x91 : Reset TX FIFO: MCU* 0x92 : Deallocate TX FIFO: LLE <sup>(1)</sup> 0x93 : Retry TX FIFO: LLE <sup>(1)</sup> 0x94 : Discard TX FIFO: MCU 0x95 : Commit TX FIFO: MCU 0xF1 : Reset both FIFOs: MCU* 0xF2 : Deallocate both FIFOs: none 0xF3 : Retry both FIFOs: none 0xF4 : Discard both FIFOs: none 0xF5 : Commit both FIFOs: none
RFFDMA0	Both	MCU
RFFDMA1	Both	MCU
RFFSTATUS	Both	N/A
RFFCFG	Both	MCU
RFRXFLEN	Both	N/A
RFRXFTHRS	Both	MCU
RFRXFWR	N/A	LLE
RFRXFRD	MCU	N/A
RFRXFWP	Both	LLE
RFRXFRP	Both	MCU
RFRXFSWP	Both	LLE
RFRXFSRP	Both	MCU
RFTXFLEN	Both	N/A
RFTXFTHRS	Both	MCU
RFTXFWR	N/A	MCU
RFTXFRD	LLE	N/A
RFTXFWP	Both	MCU
RFTXFRP	Both	LLE
RFTXFSWP	Both	MCU
RFTXFSRP	Both	LLE

<sup>(1)</sup> MCU if PRF\_ADDR\_ENTRY $n$ .CONF.RETRY is 1

### 25.3.2 DMA

It is possible to use direct memory access (DMA) to move data between memory and the radio. See [Chapter 8](#) for a detailed description on how to set up and use DMA transfers.

There are two DMA triggers associated with the radio: the RADIO DMA triggers 0 and 1 (DMA triggers 19 and 11).

The radio DMA trigger source is selected in registers RFFDMA0 and RFFDMA1. See the register descriptions in [Section 25.12](#) for details.

### 25.3.3 RAM-Based Registers

A list of the memory entries of the general radio RAM area used for parameter transfer is shown in [Table 25-4](#). All these registers are in page 0 of the radio RAM. Each memory entry is considered a RAM-based register and has a name. Numeric values that are two bytes long are represented in little-endian format.

The radio RAM registers have no defined reset value and must therefore be initialized by the MCU.

The registers `SEMAPHORE0` and `SEMAPHORE1` can be used to verify data integrity. These registers are changed to 0 when they are read. If a semaphore register is read and the value was 1, the semaphore has been successfully taken, and subsequent reads of the register return 0 until the semaphore is released. If a semaphore register is read as 0, the semaphore was not free. A semaphore can be released by writing 1 to the semaphore register; this should only be done if the semaphore has previously been taken by the MCU. The LLE takes `SEMAPHORE0` when a task starts and `SEMAPHORE1` when the radio has been set up. Both semaphores are released by the LLE at the end of the task. If the LLE is not granted the semaphore, it generates an error. If `SEMAPHORE0` and `SEMAPHORE1` are taken by the MCU before registers protected by these semaphores are modified by the MCU, data integrity is ensured, and an error occurs if the LLE is accidentally started while such an access is going on. `SEMAPHORE2` is not used by the LLE.

Where bit numbering is used, bit 0 is the LSB and bit 7 is the MSB. Multi-byte fields are little-endian.

The detailed breakdown of the address entries `PRF_ADDR_ENTRY0`–`PRF_ADDR_ENTRY7` is shown in [Table 25-5](#) or [Table 25-6](#), depending on the operational mode.

The **Prot** columns of [Table 25-4](#), [Table 25-5](#), and [Table 25-6](#) list the type of protection for each entry:

**Sem0:** Entries protected by `SEMAPHORE0`. Should only be written by the MCU while the LLE does not have `SEMAPHORE0`. Is not modified by the LLE.

**Sem1:** Entries protected by `SEMAPHORE1`. Should only be written by the MCU while the LLE does not have `SEMAPHORE1`. Is not modified by the LLE.

**Sem1/R:** Entries containing state variables and accumulative counters that are updated by the LLE. They may be read by the MCU after a receive or transmit interrupt to see how many packets have been received or transmitted. The MCU must take into account that at the time these values are read, some of them may have been updated for the next interrupt and some not. When the LLE does not have `SEMAPHORE1`, the MCU may write to them to initialize. The counters are not initialized by the LLE.

**None:** No semaphore protection; special rules apply for access.

**Table 25-4. RAM-Based Registers**

Name	Addr	Prot	Description
PRF_CHAN	0x6000	Sem0	<p><b>Bits 0–6:</b> <code>FREQ</code> Frequency to use. 0: 2379 MHz ... 1-MHz steps 116: 2495 MHz 117–126: Reserved 127: The LLE does not program frequency; it is to be set up by the MCU through the <code>FREQCTRL</code> and <code>MDMTEST1</code> registers.</p> <p><b>Bit 7:</b> <code>SYNTH_ON</code> 0: Turn off synthesizer when task is done. 1: Leave synthesizer running after task is done.</p>

**Table 25-4. RAM-Based Registers (continued)**

Name	Addr	Prot	Description
PRF_TASK_CONF	0x6001	Sem0	Configuration of task control <b>Bits 0–1: MODE</b> (operation mode) 00: Basic mode, fixed length 01: Basic mode, variable length 10: Auto mode, 9-bit header 11: Auto mode, 10-bit header <b>Bit 2: REPEAT</b> (repeated operation) 0: Single operation 1: Repeated operation <b>Bit 3: START_CONF</b> (start configuration) 0: Start each receive or transmit immediately 1: Start each receive or transmit on Timer 2 event 1 <b>Bits 4–5: STOP_CONF</b> (stop configuration) 00: No stop based on Timer 2. 01: End task after current packet is done on Timer 2 event 2 (end immediately in sync search or wait) 10: Stop transmit or receive immediately on Timer 2 event 2 11: End task on Timer 2 event 2 in first sync search or clear-channel assessment. No stop after first sync search or clear-channel assessment. <b>Bit 6: TX_ON_CC_CONF</b> 0: Listen until RSSI drops below given level, then start TX. 1: End task if RSSI is above given level <b>Bit 7: REPEAT_CONF</b> For TX_ON_CC with REPEAT = 1: 0: Listen again on repeated operation and retransmissions 1: Listen only before the first transmission, then transmit every time For RX with REPEAT = 1: 0: Recalibrate the synthesizer before listening for new packets 1: Recalibrate the synthesizer only when the task starts

**Table 25-4. RAM-Based Registers (continued)**

Name	Addr	Prot	Description
PRF_FIFO_CONF	0x6002	Sem1	<p>Configure FIFO use</p> <p><b>Bit 0:</b> AUTOFLUSH_IGN Keep received packets with unexpected sequence number in the RX FIFO. 0: Keep 1: Auto-flush</p> <p><b>Bit 1:</b> AUTOFLUSH_CRC Keep received packets with CRC error in the RX FIFO. 0: Keep 1: Auto-flush</p> <p><b>Bit 2:</b> AUTOFLUSH_EMPTY Keep packets with no payload in the RX FIFO. 0: Keep 1: Auto-flush</p> <p><b>Bit 3:</b> RX_STATUS_CONF RX FIFO channel information 0: Do not append RSSI and RES 1: Append RSSI and RES</p> <p><b>Bits 4–5:</b> RX_ADDR_CONF RX FIFO address and config byte configuration 00: Do not include address or config byte in RX FIFO 01: Include received address in RX FIFO (1-byte addresses only), but no config byte 10: Include config byte in RX FIFO, but no address byte 11: Include received address (1-byte addresses only) and config byte in RX FIFO</p> <p><b>Bits 6–7:</b> TX_ADDR_CONF TX FIFO address and config byte configuration 00: No address or config byte; read address from PRF_ADDR_ENTRY0 01: Include address byte in TX FIFO, no config byte 10: Include config byte and use address index in that byte to find address from PRF_ADDR_ENTRYn 11: Read address from TX FIFO followed by config byte (where address information is ignored). Not allowed for PRF_TASK_CONF.MODE = 00 or 01.</p>
PRF_PKT_CONF	0x6003	Sem0	<p>Packet configuration</p> <p><b>Bit 0:</b> ADDR_LEN. Number of address bytes (0 or 1).</p> <p><b>Bit 1:</b> AGC_EN 0: Do not use AGC 1: Use AGC (<a href="#">Section 25.9.2.1</a>)</p> <p><b>Bit 2:</b> START_TONE 0: Ordinary transmission 1: Override extra preamble bytes with tone and reduce synthesizer calibration time accordingly (<a href="#">Section 25.9.2.2</a>)</p> <p><b>Bits 3–7:</b> Reserved, always write 0.</p>
PRF_CRC_LEN	0x6004	Sem1	Number of CRC bytes. Permitted values: 0–4
PRF_RSSI_LIMIT	0x6005	Sem1	For transmit on clear channel. Start a transmit task by listening to the channel; start transmitting if the RSSI drops below the level (signed) given in this register.
PRF_RSSI_COUNT	0x6006–0x6007	Sem1	For transmit on clear channel. Number of additional RSSI measurements that must be below the RSSI limit before transmission takes place.

**Table 25-4. RAM-Based Registers (continued)**

Name	Addr	Prot	Description
PRF_CRC_INIT	0x6008–0x600B	Sem1	Initialization value for CRC. For less than a 4-byte CRC, the first bytes shall be 0 and the last bytes the desired value.
PRF_W_INIT	0x600C	Sem1	Byte to write to register BSP_W before a packet; initializes the PN7 whitener if that is used. If PN9 whitener is used, bit 7 should be 1.
PRF_RETRANS_CNT	0x600D	Sem1	Maximum number of retransmissions in automatic retransmit
PRF_TX_DELAY	0x600E–0x600F	Sem1	Time from end of transmission to new transmission of different payload, given in units of 62.5 ns
PRF_RETRANS_DELAY	0x6010–0x6011	Sem1	Time from end of transmission to retransmission in auto retransmit mode, given in units of 62.5 ns
PRF_SEARCH_TIME	0x6012–0x6013	Sem1	Time to perform search before giving up or retransmitting, given in 31.25-ns units. 0: Never give up. Must be at least 256 if not 0.
PRF_RX_TX_TIME	0x6014–0x6015	Sem1	Time to add to RX-TX turnaround time in RX with auto ACK, given in 31.25-ns units
PRF_TX_RX_TIME	0x6016–0x6017	Sem1	Time to add to TX-RX turnaround time in TX with auto retransmission, given in 31.25-ns units
PRF_ADDR_ENTRY0	0x6018–0x6023		Address structure for address number 0. See <a href="#">Table 25-5</a> and <a href="#">Table 25-6</a> for details.
PRF_ADDR_ENTRY1	0x6024–0x602F		Address structure for address number 1. See <a href="#">Table 25-5</a> and <a href="#">Table 25-6</a> for details.
PRF_ADDR_ENTRY2	0x6030–0x603B		Address structure for address number 2. See <a href="#">Table 25-5</a> and <a href="#">Table 25-6</a> for details.
PRF_ADDR_ENTRY3	0x603C–0x6047		Address structure for address number 3. See <a href="#">Table 25-5</a> and <a href="#">Table 25-6</a> for details.
PRF_ADDR_ENTRY4	0x6048–0x6053		Address structure for address number 4. See <a href="#">Table 25-5</a> and <a href="#">Table 25-6</a> for details.
PRF_ADDR_ENTRY5	0x6054–0x605F		Address structure for address number 5. See <a href="#">Table 25-5</a> and <a href="#">Table 25-6</a> for details.
PRF_ADDR_ENTRY6	0x6060–0x606B		Address structure for address number 6. See <a href="#">Table 25-5</a> and <a href="#">Table 25-6</a> for details.
PRF_ADDR_ENTRY7	0x606C–0x6077		Address structure for address number 7. See <a href="#">Table 25-5</a> and <a href="#">Table 25-6</a> for details.
PRF_N_TX	0x6078	Sem1/R	Total number of packets transmitted
PRF_LAST_RSSI	0x6079	Sem1/R	RSSI of last received packet
PRF_LAST_DCOFF	0x607A–0x607D	Sem1/R	DC offset of last received packet



**Table 25-4. RAM-Based Registers (continued)**

Name	Addr	Prot	Description
PRF_RADIO_CONF	0x607E	Sem0	Configure radio hardware <b>Bits 0–1: RXCAP</b> 00: Do not capture on RX packets 01: Capture start of every RX packet 10: Capture end of every RX packet 11: Capture start of first RX packet only <b>Bits 2–3: TXCAP</b> 00: Do not capture on TX packets 01: Capture start of every TX packet 10: Capture end of every TX packet 11: Capture start of first TX packet only <b>Bits 4–5: TXIF: TX IF configuration (for 2 Mbps only)</b> 00: Zero IF 01: ±1 MHz IF 10: ±2 MHz IF 11: ±3 MHz IF <b>Bit 6: DCOFF: Special dc offset handling</b> 0: Standard dc offset 1: Use special dc offset routine measuring dc offset right after RX start <b>Bit 7: DCWB: Write back dc offset estimate to override registers</b> 0: Do not write back 1: Write back after each received packet with CRC OK
PRF_ENDCAUSE	0x607F	None	Reason why LLE ended task

**Table 25-5. Address Structure for Auto Mode**

Name	Index	Prot	Description
CONF	0x00	Sem1	<b>Bit 0: ENA0</b> (Enable for primary sync word – RX task only) 0: Disable address entry for primary sync word 1: Enable address entry for primary sync word <b>Bit 1: ENA1</b> (Enable for secondary sync word – RX task only) 0: Disable address entry for secondary sync word 1: Enable address entry for secondary sync word <b>Bit 2: REUSE</b> (Allow reuse of transmitted packet) 0: LLE deallocates packet after it has been acknowledged 1: LLE does not deallocate packet after it has been acknowledged (this is up to the MCU) <b>Bit 3: AA</b> (Enable auto acknowledgement or auto retransmission) 0: Disable auto ack (RX) or auto retransmission (TX) for this address 1: Enable auto ack (RX) or auto retransmission (TX) for this address <b>Bit 4: VARLEN</b> (variable length support) 0: Use fixed length given by RXLENGTH in receiver when receiving packets or ACKs 1: Use variable length up to RXLENGTH in receiver when receiving packets or ACKs <b>Bit 5: FIXEDSEQ</b> (fixed sequence number – TX task only) 0: Insert sequence number from SEQSTAT.SEQ 1: Read sequence number from TX FIFO <b>Bit 6: TXLEN</b> 0: Insert packet length in header when transmitting 1: Used fixed-length word when transmitting Note: Must not be set to 1 unless the peer uses fixed length
RXLENGTH	0x01	Sem1	Maximum length of received packet (0–127)
ADDRESS	0x02	Sem1	Address of packet

**Table 25-5. Address Structure for Auto Mode (continued)**

Name	Index	Prot	Description
SEQSTAT	0x03	Sem1/R	<p><b>Bit 0:</b> VALID (RX task only) 0: The status is not valid. Any packet is viewed as new. On successful reception of a packet, the LLE sets this bit. 1: The status is valid. Only packets with a sequence number and CRC different from the previous one are accepted.</p> <p><b>Bits 1–2:</b> SEQ (sequence number). For RX, the sequence number of the last successfully received packet. For TX, the sequence number of the next or current packet to be transmitted</p> <p><b>Bits 3–4:</b> ACKSEQ (ACK sequence number – RX task only) For RX with auto ACK, the sequence number of the next or current ACK to be transmitted</p> <p><b>Bit 5:</b> ACK_PAYLOAD_SENT (RX task only) 0: The last received packet was not acknowledged with payload. 1: The last received packet was acknowledged with payload.</p> <p><b>Bit 6:</b> NEXTACK (next ACK buffer to use – RX task only) 0: Use ACK buffer 0. 1: Use ACK buffer 1.</p>
ACKLENGTH0	0x04	None	For RX with auto ACK: Length of payload to be transmitted from buffer 0. When 0, the buffer is free. After the payload has been transmitted and a packet with a new sequence number is received, the value is set to 0 by the LLE. The MCU only writes to the register when it is zero; the LLE only writes it to zero when it is non-zero.
ACKLENGTH1	0x05	None	For RX with auto ACK: Length of payload to be transmitted from buffer 1. When 0, the buffer is free. After the payload has been transmitted and a packet with a new sequence number is received, the value is set to 0 by the LLE. The MCU only writes to the register when it is zero; the LLE only writes it to zero when it is non-zero.
CRCVAL	0x06–0x07	Sem1/R	CRC value (last two bytes if more than 2 CRC bytes) of last successfully received packet
N_TXDONE	0x08	Sem1/R	Number of packets transmitted. For auto retransmission, only acknowledged packets with new sequence number are counted. For auto ACK, only packets with a new payload are counted when the payload has been confirmed.
N_RXIGNORED	0x09	Sem1/R	Number of retransmitted packets received with CRC OK
N_RXOK	0x0A	Sem1/R	Number of new packets received with CRC OK or ACK packets without payload received
N_RXNOK	0x0B	Sem1/R	Number of packets received with CRC error

**Table 25-6. Address Structure for Basic Mode**

Name	Index	Prot	Description
CONF	0x00	Sem1	<p><b>Bit 0:</b> ENA0 (enable for primary sync word – RX task only) 0: Disable address entry for primary sync word 1: Enable address entry for primary sync word</p> <p><b>Bit 1:</b> ENA1 (enable for secondary sync word – RX task only) 0: Disable address entry for secondary sync word 1: Enable address entry for secondary sync word</p> <p><b>Bit 2:</b> REUSE (allow reuse of transmitted packet) 0: LLE deallocates packet after it has been transmitted 1: LLE does not deallocate packet after it has been transmitted (this is up to the MCU)</p>
RXLENGTH	0x01	Sem1	Maximum length of received packet (0–255)
ADDRESS	0x02	Sem1	Address of packet
	0x03–0x09		Reserved
N_RXOK	0x0A	Sem1/R	Number of packets received with CRC OK
N_RXNOK	0x0B	Sem1/R	Number of packets received with CRC error

### 25.3.4 Variables in RAM Page 5

Some additional RAM registers are placed in page 5 of the RFCORE RAM. These variables have the prefix `PRFX` and are listed in [Table 25-7](#). The addresses overlap other RAM registers, and to access them page 5 must be selected using the `RFRAMCFG` register; see [Section 25.3](#). Some of the registers have a reset value. This value is written by the LLE shortly after it has been taken out of reset by `LLECTRL.LLE_EN` being set to 1. If the MCU must modify these registers, the modification must be done each time the LLE is reset. After taking the LLE out of reset, the MCU may modify the registers after `LLASTAT.LLE_IDLE` has gone high.

**Table 25-7. RAM-Based Registers in RAM Page 5<sup>(1)</sup>**

Name	Addr	Prot	Reset Val	Description
<code>PRFX_LAST_FREQEST</code>	0x6006	Sem1/R	–	Last frequency offset estimate, read from the <code>FREQEST</code> register at the end of receiving each packet
<code>PRFX_RSSI_LIM_LOWER</code>	0x6008	Sem1	0x20	Lower RSSI limit for use in AGC algorithm
<code>PRFX_RSSI_LIM_UPPER</code>	0x6009	Sem1	0x3C	Upper RSSI limit for use in AGC algorithm
<code>PRFX_RSSI_DIFF</code>	0x600A	Sem1	0x14	Difference between high and low RSSI gain
<code>PRFX_LNAGAIN_SAT</code>	0x600B	Sem1	0x4A	<code>LNAGAIN</code> setting to use while close to saturation
<code>PRFX_TONE_DURATION</code>	0x600C–0x600D	Sem1	0x064A	Duration of tone in start of packet if <code>PRF_PKT_CONF.START_TONE = 1</code> , given in 31.25-ns units
<code>PRFX_TONE_OFFSET</code>	0x600E–0x600F	Sem0	0x0600	Time to subtract from TX synthesizer calibration time if <code>PRF_PKT_CONF.START_TONE = 1</code> , given in 31.25-ns units

<sup>(1)</sup> Note that the LLE is reset when the device enters PM2 or PM3. This means that the `PRFX` registers must be re-initialized after coming up from one of these power modes.

The parts of RAM page 5 that are not listed in [Table 25-7](#) are reserved for use by the LLE and should not be written by the MCU.

## 25.4 Bit-Stream Processor

The **bit-stream processor** (BSP) supports automatic insertion of CRC and detection of CRC error with a programmable polynomial of 8, 16, 24, or 32 bits.

The **bit-stream processor** also supports whitening and de-whitening. The whitening sequences supported are a PN7 sequence and a PN 9 sequence compatible with CC2500 and CC2510.

The bit-stream processor is used by the LLE to do the whitening and CRC generation and checking. This operation is based on the configuration set up by the MCU. The BSP can also be run in a coprocessor mode to calculate whitened sequences and CRCs. This must only be done while the LLE is not running.

### 25.4.1 Whitening

The BSP supports two whiteners, a PN7 and a PN9 whitener. The register `BSP_MODE` is used to enable or disable each whitener. When no whitener is enabled, it outputs zero. The whitener sequence is XORed with the transmitted or received signal.

It is possible to enable both whiteners. This is useful, for example, in conjunction with the test command `CMD_TX_TEST` ([#IMPLIED](#)) to transmit a white test signal.

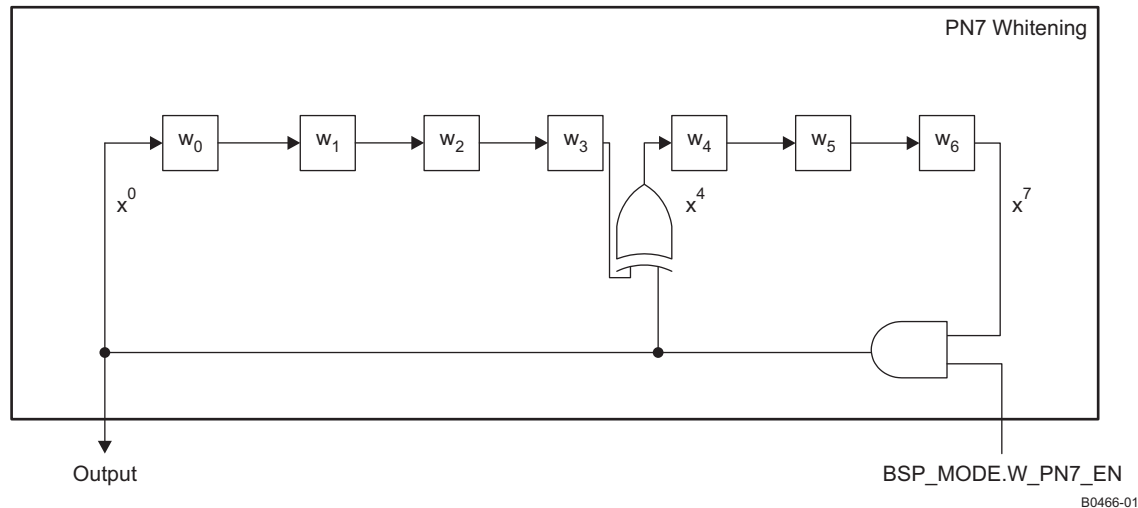
#### 25.4.1.1 PN7 Whitening

The PN7 whitener is shown in [Figure 25-3](#). It has a 7-bit whitening shift register `w` used for calculating the PN sequence given by the polynomial  $x^7 + x^4 + 1$ . The output is the same as the shift register feedback.

The  $w$  register must be initialized by writing  $w$  into register `BSP_W.W` before starting receiving or transmitting a packet. Doing this sets  $w_6$  to `BSP_W[0]`,  $w_5$  to `BSP_W[1]` and so on up to  $w_1$  to `BSP_W[5]`;  $w_0$  is set to 1.

When running normal receive or transmit tasks, writing to `BSP_W` is done by the LLE, which writes the value in `PRF_W_INIT` to this register, but for test commands and co-processor mode, the `BSP_W` register must be written by the MCU.

The PN7 whitener is enabled by the bit `W_PN7_EN` of the `BSP_MODE` register.



**Figure 25-3. PN7 Whitening**

### 25.4.2 CC2500-Compatible PN9 Whitening

The CC2500-compatible PN9 whitener is shown in [Figure 25-4](#). It has a 9-bit whitening shift register  $s$  and an 8-bit output register  $b$ . It produces a whitening sequence compatible with CC2500, CC2510 and other TI devices. These devices use the polynomial  $x^9 + x^4 + 1$ . The whitening sequence is produced one byte at a time, and the byte is bit-reversed before being XORed with a received or transmitted byte. Before starting reception or transmission of a packet, the  $s$  and  $b$  registers must be initialized to all ones by writing a 1 to register `BSP_W.W_PN9_RESET`. As for the PN7 whitener, this is done by the LLE for normal receive and transmit tasks, provided that bit 7 of `PRF_W_INIT` is 1.

In [Figure 25-4](#), the dashed arrows going from the  $s$  blocks to the  $b$  blocks denote a copy that takes place after the whitening of one byte is done. This means that the first byte is whitened by the 8 bits that are in the  $b$  register after initialization (all ones). As this byte is being whitened, the  $s$  register is updated. After the first byte is whitened, the value of the  $s$  register is copied into the  $b$  register and used for whitening the second byte.

The CC2500-compatible whitener is enabled by bit `W_PN9_EN` of the `BSP_MODE` register.

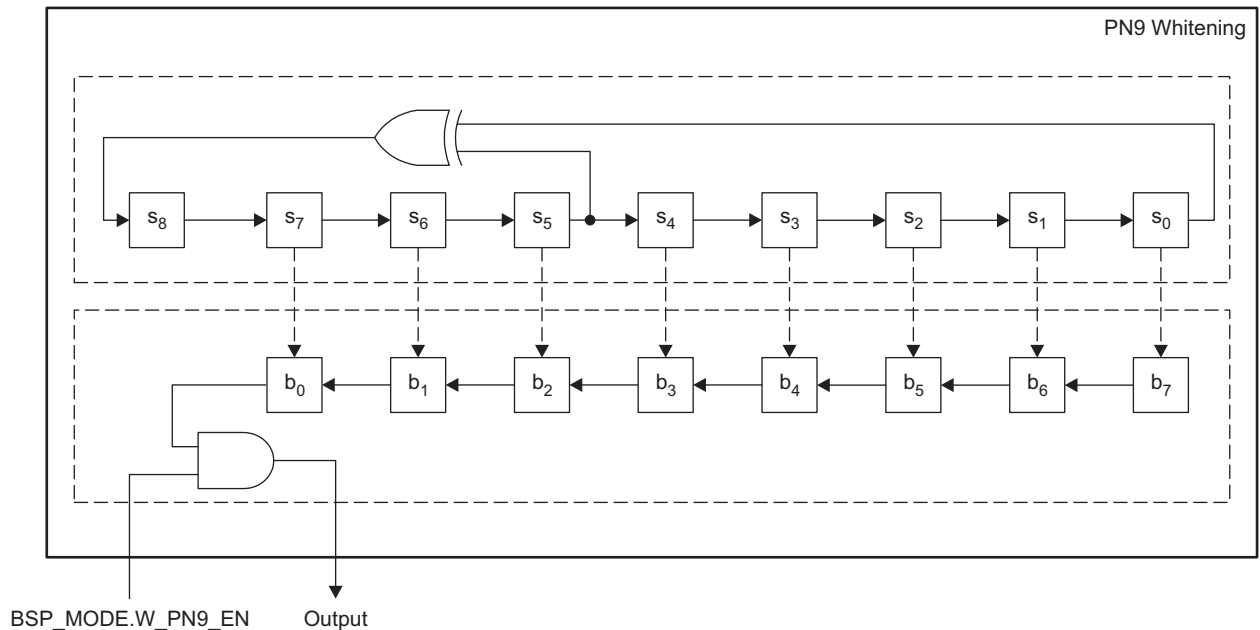


Figure 25-4. CC2500-Compatible Whitening

B0467-01

### 25.4.3 CRC

A block diagram showing the operation of the CRC module is given in Figure 25-5. The CRC sub-module has two registers:

- A 32-bit data shift register **d**
- A 32-bit register **p** for holding the polynomial

The **p** register defines the shift register used for calculating CRC. There is a feedback tap in the locations where the corresponding bit of **p** is set to 1. The module input is XORed by the output of the shift register, and this becomes the feedback of the shift register.

The current value of the data shift register **d** is the CRC value. Prior to the start of CRC calculation, the **d** and **p** registers should be initialized by writing **d** to registers `BSP_D[0-3]` and **p** to registers `BSP_P[0-3]`. The `BSP_P[0-3]` registers only must be set once, whereas the `BSP_D[0-3]` registers should be set again for each packet. In normal transmit and receive modes, this is handled by the LLE, which writes the value of `PRF_CRC_INIT[0-3]` to `BSP_D[0-3]`. At the end of CRC calculation, the value of the register is serially shifted out on the output. When performing CRC checking, all the `BSP_D[0-3]` registers should be 0 for the CRC to be OK after the received CRC has been fed through the shift register.

If whitening is enabled, calculated CRC bytes are whitened before transmission, and received CRC bytes are de-whitened before CRC checking.

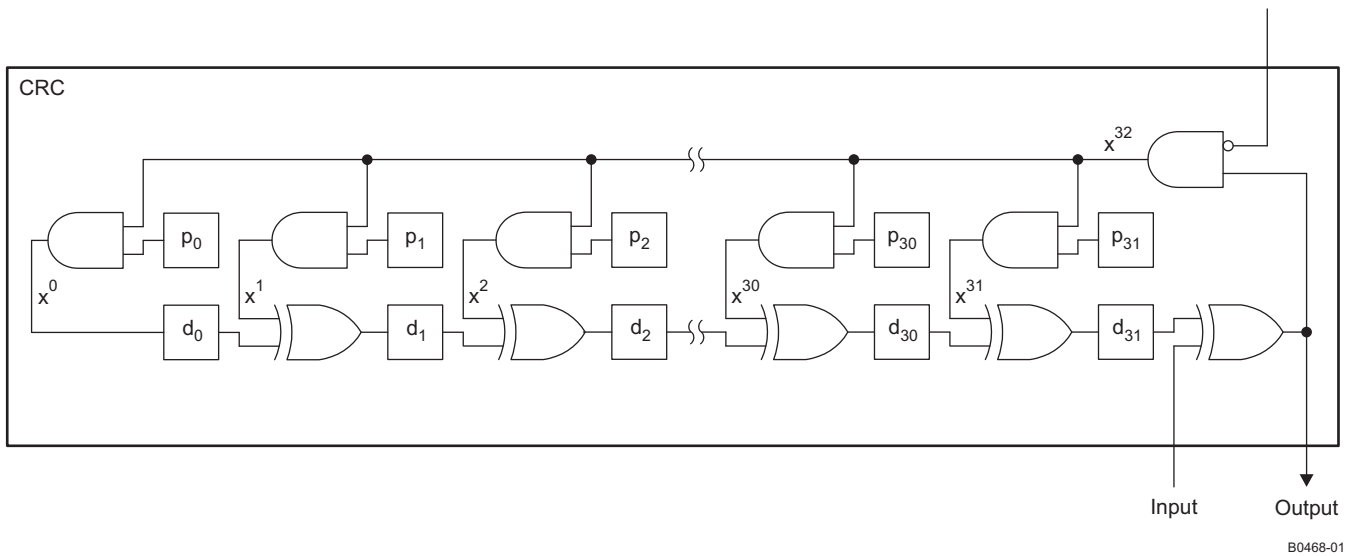


Figure 25-5. CRC Module

A 32-bit CRC polynomial can be described by the equation  $x^{32} + a_{31}x^{31} + \dots + a_1x^1 + 1$ , where all  $a_n$  are 0 or 1. To represent this, each  $P[n]$  bit in the BSP\_P0–BSP\_P3 registers should be set to  $a_n$ , and  $P[0]$  should be set to 1. To reduce the size of the polynomial to  $k$ , set the bits  $P[33 - k:0]$  to 0 and  $P[32 - k]$  to 1. In this case, the initialization value must have zeros at  $D[33 - k:0]$ . In practice, only polynomials of order 8, 16, 24, and 32 are supported, as the number of CRC bits produced in the transmitter and checked in the receiver is always a multiple of 8. The number of CRC bytes produced in normal transmit tasks is given by RAM register PRF\_CRC\_LEN.

This is summarized in Table 25-8 for the four CRC polynomial orders supported. In the BSP\_Px column, the numbers are binary, with the most significant bit at the left. In the PRF\_CRC\_INIT column, an X indicates the initialization value to use (each X does not have to be the same). Some examples are shown in Table 25-9.

Table 25-8. Register Settings for Different CRCs

Order	PRF_CRC_LEN	Polynomial	BSP_Px	PRF_CRC_INIT
8	1	$x^8 + a_7x^7 + \dots + a_1x^1 + 1$	BSP_P0 = 0000 0000 BSP_P1 = 0000 0000 BSP_P2 = 0000 0000 BSP_P3 = $a_7 a_6 a_5 a_4 a_3 a_2 a_1$	PRF_CRC_INIT[0] = 0 PRF_CRC_INIT[1] = 0 PRF_CRC_INIT[2] = 0 PRF_CRC_INIT[3] = X
16	2	$x^{16} + a_{15}x^{15} + \dots + a_1x^1 + 1$	BSP_P0 = 0000 0000 BSP_P1 = 0000 0000 BSP_P2 = $a_7 a_6 a_5 a_4 a_3 a_2 a_1$ BSP_P3 = $a_{15} a_{14} a_{13} a_{12} a_{11} a_{10} a_9 a_8$	PRF_CRC_INIT[0] = 0 PRF_CRC_INIT[1] = 0 PRF_CRC_INIT[2] = X PRF_CRC_INIT[3] = X
24	3	$x^{24} + a_{23}x^{23} + \dots + a_1x^1 + 1$	BSP_P0 = 0000 0000 BSP_P1 = $a_7 a_6 a_5 a_4 a_3 a_2 a_1$ BSP_P2 = $a_{15} a_{14} a_{13} a_{12} a_{11} a_{10} a_9 a_8$ BSP_P3 = $a_{23} a_{22} a_{21} a_{20} a_{19} a_{18} a_{17} a_{16}$	PRF_CRC_INIT[0] = 0 PRF_CRC_INIT[1] = X PRF_CRC_INIT[2] = X PRF_CRC_INIT[3] = X
32	4	$x^{32} + a_{31}x^{31} + \dots + a_1x^1 + 1$	BSP_P0 = $a_7 a_6 a_5 a_4 a_3 a_2 a_1$ BSP_P1 = $a_{15} a_{14} a_{13} a_{12} a_{11} a_{10} a_9 a_8$ BSP_P2 = $a_{23} a_{22} a_{21} a_{20} a_{19} a_{18} a_{17} a_{16}$ BSP_P3 = $a_{31} a_{30} a_{29} a_{28} a_{27} a_{26} a_{25} a_{24}$	PRF_CRC_INIT[0] = X PRF_CRC_INIT[1] = X PRF_CRC_INIT[2] = X PRF_CRC_INIT[3] = X

**Table 25-9. Register Settings for Some Commonly Used CRCs, Assuming Initialization With All 1s**

Order	PRF_CRC_LEN	CRC	BSP_Px	PRF_CRC_INIT
8	1	CRC-8-ATM $x^8 + x^2 + x + 1$	BSP_P0 = 0x00 BSP_P1 = 0x00 BSP_P2 = 0x00 BSP_P3 = 0x07	PRF_CRC_INIT[0] = 0x00 PRF_CRC_INIT[1] = 0x00 PRF_CRC_INIT[2] = 0x00 PRF_CRC_INIT[3] = 0xFF
8	1	CRC-8 $x^8 + x^7 + x^6 + x^4 + x^2 + 1$	BSP_P0 = 0x00 BSP_P1 = 0x00 BSP_P2 = 0x00 BSP_P3 = 0xD3	PRF_CRC_INIT[0] = 0x00 PRF_CRC_INIT[1] = 0x00 PRF_CRC_INIT[2] = 0x00 PRF_CRC_INIT[3] = 0xFF
16	2	CRC-16 (used in CC2500) $x^{16} + x^{15} + x^2 + 1$	BSP_P0 = 0x00 BSP_P1 = 0x00 BSP_P2 = 0x05 BSP_P3 = 0x80	PRF_CRC_INIT[0] = 0x00 PRF_CRC_INIT[1] = 0x00 PRF_CRC_INIT[2] = 0xFF PRF_CRC_INIT[3] = 0xFF
16	2	CRC-16-CCITT $x^{16} + x^{12} + x^5 + 1$	BSP_P0 = 0x00 BSP_P1 = 0x00 BSP_P2 = 0x21 BSP_P3 = 0x10	PRF_CRC_INIT[0] = 0x00 PRF_CRC_INIT[1] = 0x00 PRF_CRC_INIT[2] = 0xFF PRF_CRC_INIT[3] = 0xFF
24	3	CRC-24 $x^{24} + x^{22} + x^{20} + x^{19} + x^{18} + x^{16} + x^{14} + x^{13} + x^{11} + x^{10} + x^8 + x^7 + x^6 + x^3 + x + 1$	BSP_P0 = 0x00 BSP_P1 = 0xCB BSP_P2 = 0x6D BSP_P3 = 0x5D	PRF_CRC_INIT[0] = 0x00 PRF_CRC_INIT[1] = 0xFF PRF_CRC_INIT[2] = 0xFF PRF_CRC_INIT[3] = 0xFF
32	4	CRC-32-IEEE 802.3 $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$	BSP_P0 = 0xB7 BSP_P1 = 0x1D BSP_P2 = 0xC1 BSP_P3 = 0x04	PRF_CRC_INIT[0] = 0xFF PRF_CRC_INIT[1] = 0xFF PRF_CRC_INIT[2] = 0xFF PRF_CRC_INIT[3] = 0xFF

#### 25.4.4 Coprocessor Mode

The coprocessor mode is used to run the BSP as a stand-alone and not part of the signal path. It must not be used while the LLE is running. Coprocessor mode is selected by setting `BSP_MODE.CP_MODE` to 01 or 11. In these modes, one byte to be processed is written to the `BSP_DATA` register, and the result of processing this byte can later be read back from the same register. When `BSP_MODE.CP_MODE` is 01, the coprocessor is in receive mode, where the whitener is applied before the CRC. When `BSP_MODE.CP_MODE` is 11, the coprocessor is in transmit mode, where the whitener is applied after the CRC.

To apply the BSP operations to a byte, write it to the `BSP_DATA` register. When this register is written to, the `BSP_MODE.CP_BUSY` bit goes high.

If `CP_MODE.CP_END` is 0, the first bit provided is the LSB and the last bit is the MSB. If `CP_MODE.CP_END` is 1, the first bit provided is the MSB and the last bit is the LSB.

When `BSP_MODE.CP_BUSY` goes low, the processed data can be read from the `BSP_DATA` register. If one or both whiteners are enabled, this byte is whitened or de-whitened. Otherwise, it is the same as the byte written, except if the CRC is being read as described in the following text.

To read out a CRC in transmit mode, set `BSP_MODE.CP_READOUT` to 1. A zero must be written to the `BSP_DATA` register, and when `BSP_MODE.CP_BUSY` goes low, a CRC byte can be read from `BSP_DATA`. This should be repeated for each CRC byte. If whitening is enabled, the read back CRC bytes are whitened.

The BSP must not be set in coprocessor mode while the LLE is processing a packet.

## 25.5 Frequency and Channel Programming

For normal transmit and receive tasks, the carrier frequency is set by using the `PRF_CHAN.FREQ` register. The carrier frequency is  $2379 + n$  MHz, where  $n$  is the value of this register, and  $n$  can be from 0 to 116. This gives a frequency range from 2379 MHz to 2495 MHz. Note that this frequency range extends beyond the ISM band.

If `PRF_CHAN.FREQ` is set to 127, and for the RX and TX test commands, the frequency must be programmed directly in hardware registers. In this case, the synthesizer frequency is set by programming the 7-bit frequency word located in `FREQCTRL.FREQ[6:0]`. The synthesizer frequency is given by  $2379 + n$  MHz, where  $n$  is the value of `FREQCTRL.FREQ[6:0]`, and is programmable in 1-MHz steps. The device supports synthesizer frequencies in the range from 2379 MHz to 2495 MHz. The usable settings for `FREQ[6:0]` is consequently 0 to 116.

In RX, the system operates on a low intermediate frequency (IF) of 1 MHz for data rates up to 1 Mbps, and on a zero IF for 2 Mbps. In TX, the system supports operating on low IF or zero IF. The IF to be used for TX can be programmed in the register `MDMTEST1.TX_TONE`. The receiver may operate on a positive or negative IF when the data rate is 1 Mbps and lower; this is controlled with `MDMTEST1.RX_IF`.

When the symbol rate is 1 Mbps or lower and the LLE programs the frequency, it uses a  $\pm 1$  MHz IF on TX. For both RX and TX, a negative IF is used when `PRF_CHAN.FREQ < 62`, and a positive IF is used when `PRF_CHAN.FREQ  $\geq$  62`.

When the symbol rate is 2 Mbps and the LLE programs the frequency, it uses an IF on TX as specified in `PRF_RADIO_CONF.TXIF`. This IF may be zero, or  $\pm 1$  MHz,  $\pm 2$  MHz, or  $\pm 3$  MHz. The recommended setting is  $\pm 1$  MHz. A negative IF is used when `PRF_CHAN.FREQ < 62`, and a positive IF is used when `PRF_CHAN.FREQ  $\geq$  62`.

For all data rates, the setting of `MDMCTRL1.PHASE_INVERT` is taken into account by the LLE when finding the setting for `MDMTEST1.TX_TONE`. The `FREQCTRL` register is programmed corresponding to the programmed IF in order to operate on the channel specified by `PRF_CHAN.FREQ`.

## 25.6 Modulation Formats

The CC2541 supports GFSK and MSK modulation formats. For GFSK modulation, the deviation can be set to 160 kHz or 250 kHz (320 kHz or 500 kHz for 2 Mbps). The data rate can be set to 250 kbps, 500 kbps, 1 Mbps, or 2 Mbps. The desired modulation scheme is set in the `MDMCTRL0.MODULATION` register.

Not all combinations of modulation format, data rate and deviation are supported. [Table 25-10](#) gives an overview of supported combinations.

**Table 25-10. Supported Modulation Formats, Data Rates, and Deviations**

Modulation Format	Data Rate	Deviation	MDMCTRL0.MODULATION
GFSK	2 Mbps	500 kHz	0011
GFSK	2 Mbps	320 kHz	0111
GFSK	1 Mbps	250 kHz	0010
GFSK	1 Mbps	160 kHz	0110
GFSK	250 kbps	160 kHz	0100
MSK	500 kbps	–	1001
MSK	250 kbps	–	1000

## 25.7 Receiver

When the receiver is started, it searches for the preamble and the sync word. These are used for frequency offset compensation and bit and byte synchronization. The sync word can be programmed to be from 16 to 32 bits.



Checking the sync word is done in a two-stage process. First, a correlation value is calculated. If this correlation is above a programmable threshold, a data decision of the received sync word is done. It can be programmed in `MDMCTRL3.SYNC_MODE` whether this data decision is to be ignored, no bit errors are to be accepted, or one bit error is to be accepted. The correlation threshold value is programmed in `MDMCTRL1.CORR_THR`. This threshold value should depend on the sync word length. As a rule of thumb, a value of 0.25 times the number of bits (rounded down) can be used.

For the bit synchronization to work well, some guidelines should be followed for the sync word. It should have enough transitions, but not long runs of 10 1010... or other short, repeated patterns. Generally, a longer sync word gives better performance.

The CC2541 devices have support for two independent sync words. The primary and secondary sync words are specified in two sets of registers. The secondary sync word can be enabled by the `SW_CONF.DUAL_RX` bit, and if enabled, the received signal is correlated against both sync words. If the correlation with one of the sync words is above the threshold, data decision is done against that sync word.

While the receiver is running, a received signal strength indicator (RSSI) is updated. The RSSI is available some time after the receiver is started, regardless of whether sync is found. It can be read from the `RSSI` register, which is 0x80 when no RSSI is available. The value given is in the range 0 to approximately 64, with a change of 1 corresponding to a 1-dB change. The offset from a true dBm value depends on the receiver mode and can be found in the device data sheet. For high received signal levels, the reported RSSI saturates at one of the highest possible reported values. The accuracy and update time of the RSSI can be traded off using `MDMTEST0.RSSI_ACC`. The RSSI can be calculated over a window of 5.33  $\mu$ s or 21.3  $\mu$ s, and 1, 2, or 4 such windows can be averaged to give the result. Using a longer average time gives higher accuracy, but it takes longer before a result is ready, and doing the average over a longer time means that the result may be wrong for short packets. An average of  $n$  windows of length  $t_{RSSI}$  should only be used for packets lasting longer than  $(n + 1) t_{RSSI}$  (including preamble, sync word, and CRC).

The receiver must run dc offset estimation and removal. The dc offset estimation mode can be controlled with `MDMTEST0.DC_BLOCK_MODE`. For data rates of 1 Mbps and lower, where the receiver runs on a low IF, it is recommended to use the default setting for this register (continuous estimation). For 2 Mbps, where the receiver runs on zero IF, delayed dc offset estimation should normally be used. This causes the dc offset estimation to be done in front of the packet. The delay can be controlled through `MDMTEST0.DC_BLOCK_LENGTH` and `MDMTEST1.DC_DELAY`. The recommendation is to set `MDMTEST0.DC_BLOCK_LENGTH` to 11 (128 samples) and `MDMTEST1.DC_DELAY` to 00 (5 delays), which allows for up to approximately 105  $\mu$ s of energy in front of the packet payload, including the preamble and sync word. As an alternative for 2 Mbps, dc offset estimation can be turned off, and a previously found value can be used, written into the `DC_I_L`, `DC_I_H`, `DC_Q_L`, and `DC_Q_H` registers. Values can be found in advance, but differ for each frequency. For auto acknowledgments and other packets that are received at a known time, the LLE can perform a special dc offset algorithm as described in [Section 25.9.2](#).

## 25.8 Packet Format

The packet format is configurable. There are two operation modes for radio packet control, basic mode and auto mode. Of these, only auto mode supports automatic acknowledgment and retransmissions. The LLE-controlled part of the packet format is also different for the two modes. In basic mode, there is an optional length field followed by an optional address of 1 byte, as shown in [Figure 25-6](#). In auto mode, there is a 9-bit or 10-bit header field containing length and sequence number information. This format is shown in [Figure 25-7](#). The figures show the packet formats with their configurability. The fields with a header in gray are controlled directly by the modem and are used in the acquisition of received packets. The fields with header in white are controlled by the LLE.

Preamble	Sync word	Length	Address	Payload	CRC
1–16 bytes	16–32 bits	0–1 byte	0–1 byte	0–255 bytes	0–4 bytes

*Handled by modem*

R0009-01

**Figure 25-6. Air Interface Packet Format for Basic Mode**

Preamble	Sync word	Address	Header	Payload	CRC
1–16 bytes	16–32 bits	0–1 byte	9–10 bits	0–127 bytes	0–4 bytes
<i>Handled by modem</i>					

R0010-02

When using a 9 bit header, the payload length is limited to the range 0-63 bytes. Note that the LLE must be reset when the device enters PM2 or PM3. This means that the PRFX registers must be re-initialized after the LLE has been re-enabled after coming up from one of these power modes.

**Figure 25-7. Air Interface Packet Format for Auto Mode**

The preamble is a sequence of 1010 1010 or 0101 0101. It can be from 1 to 16 bytes. The type of preamble and the number of bytes can be set up in the MDMCTRL2 register.

The sync-word field is a synchronization word that can have any length from 16 to 32 bits. The length is programmed in the SW\_CONF.SW\_LEN register. The sync word itself is programmed in the SW0, SW1, SW2, and SW3 registers for the primary sync word, and SW4, SW5, SW6, and SW7 for the secondary sync word. The bit ordering of the sync word is set up with MDMCTRL2.SW\_BIT\_ORDER. If SW\_BIT\_ORDER is 0, the LSB of SW0 (SW4) is transmitted first and the MSB of SW3 (SW7) is transmitted last. If SW\_BIT\_ORDER is 1, the MSB of SW3 (SW7) is transmitted first and the LSB of SW0 (SW4) is transmitted last. The first bit transmitted is always the same regardless of the sync word length; the unused bits for sync word length of less than 32 bits are the ones that would have been transmitted last.

The optional length byte in basic mode (see Figure 25-6) is present if PRF\_TASK\_CONF.MODE = 01. It indicates the number of address and payload bytes following the length byte. If the length field is not present, the length is fixed as described in Section 25.9.2.

The optional address is 1 byte if present; the length is configured with the PRF\_PKT\_CONF.ADDR\_LEN register. In the transmitter, the address can be used for identification or to direct the message to a particular receiver, and in the receiver, the address can be used to filter out messages from unknown or unwanted transmitters and to distinguish between messages from different transmitters. See Section 25.9.2 for details on how the address is used. Note that for the packet format in Figure 25-7 or if a length field is not used, the address field immediately follows the sync word, and can thus be seen as an extension of it.

The 9-bit or 10-bit header shown in Figure 25-7 is shown in more detail in Figure 25-8 and Figure 25-9. This field consists of a 6-bit or 7-bit length followed by a 2-bit sequence number and a flag called NO\_ACK (NOA in Figure 25-8 and Figure 25-9) to inform that acknowledgment of the packet is not expected. If the configuration is to use a fixed length, the value of the length field is ignored in the receiver. It can be configured always to set the length field to 11 0011 in the transmitter for fixed-length packets.

Length						SEQ		NOA
Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	LSB

R0011-01

**Figure 25-8. Bits of 9-Bit Header**

Length							SEQ		NOA
Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	LSB

R0012-01

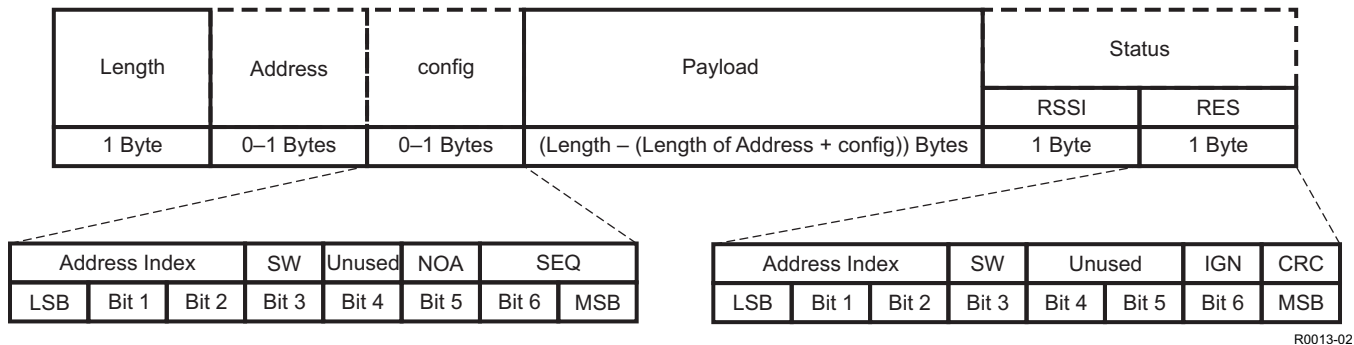
**Figure 25-9. Bits of 10-Bit Header**

The payload can be from zero to 255 bytes in basic mode, but the sum of the number of address and payload bytes must not exceed 255. In auto mode, the payload can be from 0 to 63 bytes with a 9-bit header or 0 to 127 bytes with a 10-bit header. The maximum packet length can be limited, see Section 25.9.2.3.1 and Section 25.9.2.3.2

The bit ordering when transmitting the length, address, payload, and CRC bytes is set up with the `ENDIANNESS` bit of the `FRMCTRL0` register; if 0, the LSB of each byte is transmitted first and if 1, the MSB is transmitted first. Normally, `FRMCTRL0.ENDIANNESS` and `MDMCTRL2.SW_BIT_ORDER` should have the same value. Note that for correct operation in auto mode, `FRMCTRL0.ENDIANNESS` must be set to 1 so that MSB is transmitted first.

The CRC field contains 0 to 4 bytes and is used to check the packet for errors if present. See [Section 25.4.3](#) on how to set up the CRC generation and checking.

### 25.8.1 RX FIFO Packet Organization



R0013-02

Figure 25-10. Structure of Packets in the RX FIFO

The structure of a packet in the RX FIFO is shown in [Figure 25-10](#). All packets start with a length byte, regardless of whether a length byte is present on the air. The length is the number of bytes in the address, config, and payload fields following the length byte, and it may be modified compared to the length received on the air or configured as fixed-length. If packets are longer than what can fit in the FIFO, packets must be read from the FIFO while reception takes place, either by DMA or directly by the MCU. The auto-flush options in `PRF_FIFO_CONF` cannot be used in this case, and auto-commit and auto-deallocate must be enabled for the RX FIFO in `RFFCFG`.

The address byte is placed after the length byte and is present if configured in `PRF_FIFO_CONF.RX_ADDR_CONF`. The address is written in the FIFO as it was received on the air.

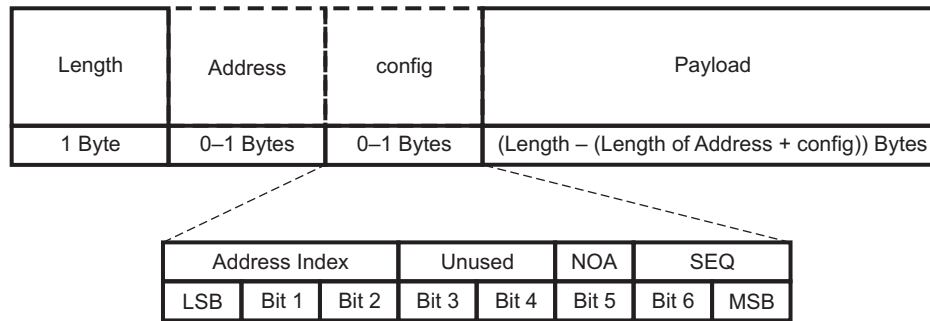
The config byte following the length byte and address byte is present if configured in `PRF_FIFO_CONF.RX_ADDR_CONF`. In this case, the index *n* to the `PRF_ADDR_ENTRYn` containing the received address is present in bits 0–2, and bit 3 is 0 if the primary sync word was received and 1 if the secondary sync word was received. In auto mode, the 3 MSBs of the config byte are set to the 3 LSBs of the received header.

The payload is as received on the air. In case of an empty packet, there is no payload.

The status field consists of 2 bytes appended to the FIFO entry if configured in `PRF_FIFO_CONF.RX_STATUS_CONF`. The presence of a status field is not reflected in the value of the length byte, so if a status field is present, the MCU must read 2 extra bytes. It is possible to configure this even using DMA with automatic length extraction. The status bytes are:

- RSSI is the received signal-strength indication from the demodulator.
- RES contains information on the address and CRC result.
  - The 3 LSBs contain the address index as in the config byte.
  - Bit 3 is 0 if the primary sync word was received and 1 if the secondary sync word was received.
  - IGN is 1 for packets that may be ignored by the MCU due to repeated sequence number and 0 otherwise.
  - CRC is 1 if there was a CRC error and 0 otherwise.

## 25.8.2 TX FIFO Packet Organization



R0014-02

**Figure 25-11. Structure of Packets in the TX FIFO**

The structure of a packet in the TX FIFO is shown in [Figure 25-11](#). All packets start with a length byte, regardless of whether a length byte is present on the air. The length is the number of bytes in the address, config, and payload fields following the length byte, and it may be modified before being transmitted on the air. If a fixed length is used, it is up to the MCU to ensure that the length is correct given the fixed length expected by the receiver. If packets are longer than what can fit in the FIFO, packets must be written to the FIFO while transmission takes place, either by DMA or directly by the MCU. Auto-commit and auto-deallocate must then be enabled for the TX FIFO in `RFFCFG`.

The address byte is placed after the length byte and is present if configured in `PRF_FIFO_CONF.TX_ADDR_CONF`. If it is included, the address is transmitted on the air as it is read from the FIFO. If it is not included, but a config byte is included, the three LSBs of the config byte tell the index  $n$  of `PRF_ADDR_ENTRYn` from which the address is inserted. If neither an address nor a config byte is included, the address is inserted from `PRF_ADDR_ENTRY0.ADDRESS`.

The config byte following the length byte and optional address byte is present if configured in `PRF_FIFO_CONF.TX_ADDR_CONF`. This byte contains an address index which is used to determine the address if no address byte is included as explained previously. If an address byte is included, the address index is used to determine which address entry to read the configuration from, but the `ADDRESS` field in that address entry is ignored. In auto mode, the `NO_ACK` bit (LSB) of the transmitted header is set to bit 5 of the config byte. If `PRF_ADDR_ENTRYn.CONF.FIXEDSEQ`, where  $n$  is the index of the address used, is 1, the `SEQ` field of the transmitted header is taken from the `SEQ` field (bits 6–7) of the config byte; otherwise, the sequence number on the air is inserted from `PRF_ADDR_ENTRYn.SEQSTAT.SEQ`. If the config byte is not included, the `NO_ACK` bit is always sent as 0 and `PRF_ADDR_ENTRYn.CONF.FIXEDSEQ` should be 0 (otherwise the `SEQ` field always remains 0). The payload is transmitted as present in the FIFO.

## 25.8.3 TX Buffers for ACK Payload

The hardware TX FIFO is not used for ACK payload in RX tasks in auto mode. Instead, an acknowledgment packet for each address can be placed in one of two dedicated buffers for that address. These two buffers constitute a FIFO capable of holding two packets. The buffers for the first two addresses are placed in the RAM page normally used for the hardware TX FIFO. These four buffers can either be accessed from the TX FIFO space at `0x6100` or by selecting page 7 through `RFRAMCFG`, but the TX FIFO registers should not be used. The other twelve buffers must be addressed from the configurable radio memory bank through the `RFRAMCFG` register. The mapping of each buffer is shown in [Table 25-11](#).

**Table 25-11. Segments for Holding ACK Payload for Each Address Entry**

Address Entry Number	Buffer Number	Setting of RFRAMCFG	Start Address
0	0	7 or X	0x6000 or 0x6100
0	1	7 or X	0x6020 or 0x6120
1	0	7 or X	0x6040 or 0x6140
1	1	7 or X	0x6060 or 0x6160
2	0	1	0x6000

**Table 25-11. Segments for Holding ACK Payload for Each Address Entry (continued)**

Address Entry Number	Buffer Number	Setting of RFRAMCFG	Start Address
2	1	1	0x6020
3	0	1	0x6040
3	1	1	0x6060
4	0	2	0x6000
4	1	2	0x6020
5	0	2	0x6040
5	1	2	0x6060
6	0	3	0x6000
6	1	3	0x6020
7	0	3	0x6040
7	1	3	0x6060

The status of buffer  $k$  for address  $n$  is contained in the `PRF_ADDR_ENTRYn.ACKLENGTHk` register. If the value is 0, the buffer is free.

In order to enter a payload for address  $n$ , the MCU must follow the following procedure:

1. Read `PRF_ADDR_ENTRYn.ACKLENGTH0` and `PRF_ADDR_ENTRYn.ACKLENGTH1`. Call the values `len_0` and `len_1`, respectively.
2. Read `PRF_ADDR_ENTRYn.SEQSTAT.NEXTACK` and call this value  $k$ . Let  $m$  be NOT  $k$  (that is,  $1 - k$ ).
3. Check if `len_k` is 0. If so, write the payload to buffer  $k$  for address entry  $n$  (see [Table 25-11](#)), then write the payload length to `PRF_ADDR_ENTRYn.ACKLENGTHk`. End the procedure.
4. Otherwise, check whether `len_m` is 0. If so, write the payload to buffer  $m$  for address entry  $n$  (see [Table 25-11](#)), then write the payload length to `PRF_ADDR_ENTRYn.ACKLENGTHm`. End the procedure.
5. Otherwise, no ACK payload buffer for that address is free, and no payload can be entered at this time.

The ACK payload length can be 1–32. When a buffer becomes free, the LLE writes the `PRF_ADDR_ENTRYn.ACKLENGTHk` to 0 and raises a `TXDONE` interrupt.

A buffer contains only the payload to be transmitted. The length is given by `PRF_ADDR_ENTRYn.ACKLENGTHk`, and the address and sequence number are as described in [Section 25.9.2.3.2](#).

In order to flush the buffers for address  $n$ , issue the command `CMD_FLUSH_ACK n` (see [Table 25-12](#)). This causes the LLE to write `PRF_ADDR_ENTRYn.ACKLENGTH0` and `PRF_ADDR_ENTRYn.ACKLENGTH1` to 0 and clear `PRF_ADDR_ENTRYn.SEQSTAT.ACK_PAYLOAD_SENT`. If no task is running, the LLE takes `SEMAPHORE1`; if it fails, it does not write to `PRF_ADDR_ENTRYn.SEQSTAT.ACK_PAYLOAD_SENT`. If the transmission of an acknowledgment with payload had started on that address, flushing happens after the transmission is finished. After the flushing is done, the LLE raises a `TXFLUSHED` interrupt.

## 25.9 Link Layer Engine

The link layer engine controls radio operation. It is started by setting the `LLECTRL.LLE_EN` bit to 1. The LLE must be started before the radio can be operated.

The LLE can be reset by clearing and setting `LLECTRL.LLE_EN`. The LLE should not be reset while the radio is active. The MCU should not enter PM1, PM2, or PM3 while the LLE is running a task. Before entering PM2 or PM3, `LLECTRL.LLE_EN` must be set to 0, otherwise the behavior of the RF core may be unpredictable after waking up. The mode of the LLE is selected with `LLECTRL.LLE_MODE_SEL`. For the proprietary-mode operation described in this chapter, this field must be written to 00. For BLE operation, this field is 01; that value should only be written by the TI BLE stack. In order to switch modes, the LLE must be reset; writing to `LLECTRL.LLE_MODE_SEL` while `LLECTRL.LLE_EN` is 1 has no effect.

### 25.9.1 Command Register

The command register `RFST` can be used for sending commands to the LLE and the FIFOs. Commands in the range `0x80–0xFF` are commands to the FIFOs, see [Section 25.3.1](#). Other commands are commands to the LLE.

The commands are listed in [Table 25-12](#). There are commands for starting receive and transmit modes. In addition, there is a command `CMD_SHUTDOWN` to stop the radio operation and end the task directly. The commands `CMD_SEND_EVENT1` and `CMD_SEND_EVENT2` do the same action as receiving a Timer 2 event 1 or event 2.

If an unknown command is entered, the LLE responds by generating an `LLEERR` interrupt. If a task is running, it stops.

When sending a command to the LLE, the `RFST` register retains its value until the LLE has received the command (but not necessarily executed it) and then is set to 0. Commands should not be sent to the LLE unless `RFST` is 0. FIFO commands may be sent at any time.

**Table 25-12. Commands From MCU to LL Engine via `RFST` Register**

Number	Command Name	Description
0x01	<code>CMD_SHUTDOWN</code>	Stop operation immediately
0x02	<code>CMD_DEMOD_TEST</code>	Start demodulator without sync search
0x03	<code>CMD_RX_TEST</code>	Start demodulator and sync search
0x04	<code>CMD_TX_TEST</code>	Start transmitter and transmit zeros
0x05	<code>CMD_TX_FIFO_TEST</code>	Start transmitter and transmit from TX FIFO
0x06	<code>CMD_PING</code>	Respond with a <code>PINGRSP</code> interrupt
0x08	<code>CMD_RX</code>	Start receive operation
0x09	<code>CMD_TX</code>	Start transmit operation
0x0A	<code>CMD_TX_ON_CC</code>	Start transmit operation on clear channel
0x0B	<code>CMD_STOP</code>	Gracefully stop radio task
0x21	<code>CMD_SEND_EVENT1</code>	Do the same action as if Timer 2 event 1 was observed
0x22	<code>CMD_SEND_EVENT2</code>	Do the same action as if Timer 2 event 2 was observed
0x30	<code>CMD_FLUSH_ACK0</code>	Flush the ACK payload buffers for address 0
0x31	<code>CMD_FLUSH_ACK1</code>	Flush the ACK payload buffers for address 1
0x32	<code>CMD_FLUSH_ACK2</code>	Flush the ACK payload buffers for address 2
0x33	<code>CMD_FLUSH_ACK3</code>	Flush the ACK payload buffers for address 3
0x34	<code>CMD_FLUSH_ACK4</code>	Flush the ACK payload buffers for address 4
0x35	<code>CMD_FLUSH_ACK5</code>	Flush the ACK payload buffers for address 5
0x36	<code>CMD_FLUSH_ACK6</code>	Flush the ACK payload buffers for address 6
0x37	<code>CMD_FLUSH_ACK7</code>	Flush the ACK payload buffers for address 7

### 25.9.2 Radio Tasks

Before starting a task, radio registers should be set up with the desired packet format, and the desired input sensitivity and output power should be programmed. Furthermore, the sync word in use must be programmed in the `SW0`, `SW1`, `SW2`, and `SW3` registers. If a secondary sync word is used, it must be programmed in the `SW4`, `SW5`, `SW6`, and `SW7` registers. The RAM registers must be programmed to configure the task. The way the task runs depends on the `PRF_TASK_CONF` register. The operation mode is set up by the `MODE` bits of this register. A value of 00 or 01 gives basic mode and thus disables auto ACK or auto retransmission. A value of 10 or 11 gives auto mode where auto acknowledgment or auto retransmission can be enabled per the address in `PRF_ADDR_ENTRYn.CONF.AA`.

All tasks start with a start-of-task command from the MCU. The LLE takes `SEMAPHORE0` at this time; if the semaphore is not available, the task ends with an error. Depending on the configuration in `PRF_TASK_CONF.START_CONF`, the LLE either starts the task immediately or waits for a Timer 2 event 1 before starting. Note that a Timer 2 event 1 may be pending from before the LLE starts waiting; in that case, the task starts immediately. To clear a pending Timer 2 event 1, reset the LLE. To prevent

unwanted events from reaching the LLE, Timer 2 event 1 can be disabled in the `T2EVTCFG` register; see [Chapter 22](#). The frequency word is programmed according to the setting of `PRF_CHAN.FREQ`, except if it is 127, in which case no frequency programming is done and any value written by the MCU is retained. When using auto mode on 2 Mbps, the frequency must be programmed through the `PRF_CHAN.FREQ` register. Then the LLE changes the IF frequency automatically (for 2 Mbps, the recommended settings use different IF for transmission and reception) when changing from receive operation to transmit operation (for sending an acknowledgment packet) and vice versa. The LLE starts configuring the transmitter or receiver, depending on the type of task. After the transmitter or receiver has been set up, the LLE takes `SEMAPHORE1` to gain access to the remaining RAM-based registers, read the parameters, and start transmission or reception.

Programming of frequency is done as described in [Section 25.5](#). For symbol rates of 1 Mbps and lower, RX and TX are done on the same synthesizer frequency, whereas for a symbol rate of 2 Mbps, the synthesizer frequency changes between RX and TX. This change is done without a recalibration of the synthesizer.

At the end of a packet, the LLE reads the `RSSI` register and writes the value to the `PRF_LAST_RSSI` register and, if so configured, to the `RSSI` byte of the RX FIFO. This read is done after the next-to-last byte has been obtained from the demodulator. Note that for a bit rate of 2 Mbps and for sync words shorter than 32 bits, `MDMCTRL3.RSSI_MODE` should be set to 11 to ensure a correct reading. Before turning off the demodulator, the LLE reads the dc offset from the `DC_I_L`, `DC_I_H`, `DC_Q_L`, and `DC_Q_H` registers and writes the result to `PRF_LAST_DCOFF` (in the byte order listed for the register read). The LLE also reads the frequency offset from the `FREQEST` register and writes the result to `PRFX_LAST_FREQEST` (see [Table 25-7](#)).

If `PRF_RADIO_CONF.DCOFF` is 1, the LLE runs a procedure that estimates the dc offset right after receiver startup. This mode is suitable for packets that are known to be received at a certain time, such as acknowledgment packets. In this mode, the LLE starts the receiver with normal dc cancellation mode and forces the LNA gain to minimum. After a short time, the LLE reads out the value of the dc offset estimate, writes it into the override registers, and selects manual override mode for dc offset estimation. It sets the LNA gain back to the programmed value and after a waiting time to allow the LNA to stabilize, starts sync search. The time to start RX with this mode is the same as for ordinary start of RX.

If `PRF_RADIO_CONF.DCWB` is 1, the LLE writes the dc offset estimate read out at the end of the packet into the dc offset override register, provided that the received packet did not have a CRC error. This is suited for the delayed dc offset mode, where the override value for dc offset is used before a delayed dc offset is available.

Some of the RAM registers are checked by the LLE to verify that their values are permitted. This applies to `PRF_CHAN.FREQ`, `PRF_FIFO_CONF.TX_ADDR_CONF`, and `PRF_CRC_LEN`. If any of these registers has values that are not permitted, the task ends with an error.

A `CMD_SHUTDOWN` command, undefined command, or any command starting a new task, ends the task immediately. If a packet was being transmitted or received, an `RXTXABO` interrupt to the MCU is raised. This means that to avoid unwanted abort of commands, the CPU should wait for a `TASKDONE` interrupt or check that `LLESTAT.LLE_IDLE` is 1 before starting another command.

If a `CMD_STOP` command is received, the task ends after the current reception or transmission is done. Timer 2 event 2 can be configured to end a task: If `PRF_TASK_CONF.STOP_CONF` is 01, Timer 2 event 2 behaves as a `CMD_STOP`, and if `PRF_TASK_CONF.STOP_CONF` is 10, Timer 2 event 2 behaves as a `CMD_SHUTDOWN`. Setting `PRF_TASK_CONF.STOP_CONF` to 00 disables Timer 2 event 2 as a stop event. With the 11 setting, Timer 2 event 2 only applies to sync search or listen right after a `CMD_RX` or `CMD_TX_ON_CC` (this setting is not meaningful for a `CMD_TX` task) or a start by Timer 2 event 1. This is explained in later subsections.

Timer 2 may capture the time of a packet based on the setting in `PRF_RADIO_CONF`. The fields `TXCAP` and `RXCAP` decide how capture is configured for TX and RX, respectively; see [Table 25-13](#). The captured value can be read from the registers `T2M0`, `T2M1`, `T2MOV0`, `T2MOV1`, and `T2MOV2` when `t2_cap` and `t2ovf_cap` are selected using the `T2MSEL` register; see [Chapter 22](#).

**Table 25-13. Timer 2 Capture Settings**

<b>TXCAP</b>	<b>Description</b>
00	Capture of transmitted packets off
01	Capture the start (after the sync word) of every transmitted packet
10	Capture the end of every transmitted packet
11	Capture the start of the first transmitted packet, that is, capture of transmitted packets is turned off after a packet has been transmitted.
<b>RXCAP</b>	<b>Description</b>
00	Capture of received packets off
01	Capture the start (after the sync word) of every received packet
10	Capture the end of every received packet
11	Capture the start of the first received packet, that is, capture of received packets is turned off after a packet has been fully received.

When capture is done at the beginning of a packet, the time captured is the time right after the sync word has been received or transmitted. Setting `TXCAP` or `RXCAP` to 11 enables capture at the start of a packet, but the capture is turned off after a packet has been transmitted or fully received in a task, so it is the start of the first packet in the task that is captured. The MCU should normally only read the captured value after a task is done; otherwise, the captured value may be overwritten with a new value. The user must take into account that a timer value may be captured on a received packet that does not match the address or that has a length which is not permitted, and that is thus not reported. It is possible to turn on capture for both received and transmitted packets in the same task. If so, it is up to the user to determine if the captured value was from a received or transmitted packet.

When a task is finished, the LLE writes an end-of-task cause in `PRF_ENDCAUSE`, frees the semaphores, raises a `TASKDONE` interrupt, and halts its operation. The possible values of `PRF_ENDCAUSE` are listed in [Table 25-14](#).

If `PRF_CHAN.SYNTH_ON` is 1, the synthesizer is not turned off after the task ends. This can be used to start a new task immediately on the same channel and get faster start of RX or TX. To do so, the next task should be started with `PRF_CHAN.FREQ` set to 127. Note that the synthesizer should not be allowed to run for a long time after a task has ended, as this causes excessive power consumption. The synthesizer can be stopped by sending a `CMD_SHUTDOWN` command.

**Table 25-14. End-of-Task Causes**

<b>Number</b>	<b>Name</b>	<b>Description</b>
<b>Normal task ending</b>		
0	<code>TASK_ENDOK</code>	Task ended normally
1	<code>TASK_RXTIMEOUT</code>	Timer 2 event 2 or <code>CMD_STOP</code> observed while waiting for RX sync
2	<code>TASK_NOSYNC</code>	Sync was not obtained in the specified time
3	<code>TASK_NOCC</code>	<code>TX_ON_CC</code> ended because channel was not clear
4	<code>TASK_MAXRT</code>	Task ended because maximum number of retransmissions was reached
5	<code>TASK_STOP</code>	Task ended after transmission or reception by Timer 2 event 2 or <code>CMD_STOP</code> while transmitting or receiving or with ACK or retransmission in progress
6	<code>TASK_ABORT</code>	Task aborted by command
<b>MCU interface error</b>		
255	<code>TASKERR_INTERNAL</code>	Internal program error
254	<code>TASKERR_CMD</code>	Unknown command
253	<code>TASKERR_SEM</code>	Unable to obtain semaphore
252	<code>TASKERR_PAR</code>	Unpermitted parameter
251	<code>TASKERR_TXFIFO</code>	TX FIFO without available data when not permitted
250	<code>TASKERR_RXFIFO</code>	Overfull RX FIFO in TX task



**Table 25-14. End-of-Task Causes (continued)**

Number	Name	Description
249	TASKERR_MODUNF	Modulator underflow observed

### 25.9.2.1 AGC Algorithm

If `PRF_PKT_CONF.AGC_EN` is 1, an automatic gain control (AGC) algorithm is run while the receiver is looking for sync. The AGC algorithm switches between two different front-end gain settings in the `LNAGAIN` register. It is recommended to use AGC when running on 2 Mbps to improve the saturation performance.

Parameters for control of the AGC algorithm are found in page 5 of the radio RAM; see [Table 25-7](#). This table lists reset values that the LLE sets for these parameters.

The LLE polls the RSSI value at every update and compares it to the values of `PRFX_RSSI_LIM_LOWER` and `PRFX_RSSI_LIM_UPPER`. If the observed RSSI is below `PRFX_RSSI_LIM_LOWER`, the LNA gain is set to the high gain setting. If the observed RSSI is above `PRFX_RSSI_LIM_UPPER`, the LNA gain is set to the low gain setting. If the observed RSSI is between these limits, the LNA gain is not changed.

The high gain to use is the value found in the `LNAGAIN` register when the task is started. The low gain to use is the value found in the `PRFX_LNAGAIN_SAT` RAM register.

The `PRFX_RSSI_LIM_LOWER` and `PRFX_RSSI_LIM_UPPER` values must differ in order to account for the difference that is observed from the RSSI register when the LNA gain is changed and to have hysteresis to avoid too-frequent gain changes.

When sync is obtained on the receiver, the AGC algorithm stops updating the LNA gain, which remains at the value last set. When the receiver is switched off, the `LNAGAIN` register is set back to the value it had when the task started, that is, the high gain setting.

When the gain is reduced during the reception of a packet, the value found in the `PRF_LAST_RSSI` register and (if configured) in the RSSI byte of the RX FIFO is updated to reflect this. This update is done by adding the value of the register `PRFX_RSSI_DIFF` to the value found in the RSSI register. `PRFX_RSSI_DIFF` should therefore contain the difference between the RSSI offset for the two LNA gain settings in use, available from the device data sheet. Note that the hardware RSSI register is not updated this way.

For the AGC algorithm to operate correctly, it requires some signal, having the same power as the packet, transmitted in the band in front of the packet. That signal can be extra preamble bytes or tone. The length required for this signal depends on the RSSI accuracy setting in `MDMTEST0.RSSI_ACC`, see [Section 25.7](#). An average of  $n$  windows of length  $t_{RSSI}$  requires the extra signal to last at least  $(n + 1) t_{RSSI}$ . Extra preamble bytes can be set up using `MDMCTRL2.NUM_PREAM_BYTES`. Note that the extra signal required comes in addition to the 1 preamble byte always used in a packet. When adding extra preamble bytes, this must be accounted for in `PRF_TX_DELAY`, `PRF_RETRANS_DELAY`, and `PRF_RX_TX_TIME`. The RX requires  $n \times t_{RSSI}$  extra time to start when using the AGC. In the dc offset estimation, the extra signal must be accounted for when setting the delay.

### 25.9.2.2 Tone in Front of Packet

In order to get the transmission format to resemble that of other vendors, a tone may be sent in front of the preamble. This tone can be used by the AGC algorithm on the receiver side. If `PRF_PKT_CONF.START_TONE` is 1, such a tone is transmitted as a replacement of the first preamble bytes. This means that this feature must only be used in combination with increasing the number of preamble bytes. The tone lasts for a time given by the RAM register `PRFX_TONE_DURATION`. In order to get a smooth transition from tone to preamble, it is recommended to set `PRFX_TONE_DURATION` as given in [Table 25-15](#).

Tone transmission is allowed to coincide with the synthesizer stabilizing (this may, however, cause the start of the tone to have larger frequency variations than the packet). For this reason, when `PRF_PKT_CONF.START_TONE` is 1, the synthesizer startup time is reduced by the value of the register `PRFX_TONE_OFFSET`. This should normally correspond to the time of the extra preambles, but it must not be larger than 4096 (corresponding to 128  $\mu$ s). `PRFX_TONE_OFFSET` can thus be used to compensate for the extra time added by the extra preamble bytes used for tone generation. However, the duration of the extra preamble bytes configured must be accounted for in `PRF_TX_DELAY`, `PRF_RETRANS_DELAY`, and `PRF_RX_TX_TIME`.

The default values of `PRFX_TONE_DURATION` and `PRFX_TONE_OFFSET` correspond to 48  $\mu$ s and are tuned for using 12 extra preamble bytes (13 in total) on 2 Mbps. When using the reset values, `MDMCTRL2.NUM_PREAM_BYTES` should thus be set to 0x0C.

If `PRFX_TONE_DURATION` is set too large compared to the number of preamble bytes configured, the modulator underflows. If this happens, the task ends with `TASKERR_MODUNF` as end cause.

**Table 25-15. Recommended RAM Register Settings for Start Tone**

Data Rate	PRFX_TONE_DURATION	PRFX_TONE_OFFSET
2 Mbps	$\text{MDMCTRL2.NUM\_PREAM\_BYTES} \times 0x80 + 0x4A$	$\text{MDMCTRL2.NUM\_PREAM\_BYTES} \times 0x80$
1 Mbps	$\text{MDMCTRL2.NUM\_PREAM\_BYTES} \times 0x100 + 0x52$	$\text{MDMCTRL2.NUM\_PREAM\_BYTES} \times 0x100$
500 kbps	$\text{MDMCTRL2.NUM\_PREAM\_BYTES} \times 0x200 + 0x62$	$\min(\text{MDMCTRL2.NUM\_PREAM\_BYTES} \times 0x200, 0x1000)$
250 kbps	$\text{MDMCTRL2.NUM\_PREAM\_BYTES} \times 0x400 + 0x82$	$\min(\text{MDMCTRL2.NUM\_PREAM\_BYTES} \times 0x400, 0x1000)$

### 25.9.2.3 Receive Task

When a `CMD_RX` command is received, the LLE configures the radio on the channel given by `PRF_CHAN.FREQ` and starts listening for a sync word.

The LLE can set up an internal time-out for the sync search in the `PRF_SEARCH_TIME` register. If this register is non-zero and no sync has been obtained in the number of 32 MHz cycles given by this register, the task ends with a `TASK_NOSYNC` end cause. Note that the value of this register must be large enough to have time for the duration of the sync word and 1 preamble byte, in addition to some margin, in order to get sync. The task can also be set up to end on Timer 2 event 2, based on `PRF_TASK_CONF.STOP_CONF`. If this bit field is 11, the Timer 2 event 2 time-out applies only during the first sync search after a `CMD_RX` command has been issued if `PRF_TASK_CONF.START_CONF` is 0. In this case, the time-out in `PRF_SEARCH_TIME` does not apply to the first sync search, but it still applies to subsequent sync searches in the same task. If `PRF_TASK_CONF.STOP_CONF` is 11 and `PRF_TASK_CONF.START_CONF` is 1, the time-out applies to every sync search and `PRF_SEARCH_TIME` never applies, but the Timer 2 event 2 timeout does not apply after sync is obtained or while waiting for Timer 2 event 1 to restart listening. If sync is obtained, the LLE starts reading the packet.

If sync is found on a packet, the time of sync is captured by the Timer 2 capture function (see [Section 22.1.10](#)).

#### 25.9.2.3.1 Basic Mode

This section describes the receive operation if `PRF_TASK_CONF.MODE` is 00 or 01.

If `PRF_TASK_CONF.MODE` is 01, the length byte is read first. It gives the number of bytes between the length byte and the CRC, including the address. If the length is too small to contain the address, the reception of the packet is stopped and the device goes back to sync search (regardless of the setting in `PRF_TASK_CONF.REPEAT`).

Next, the address byte is read if `PRF_PKT_CONF.ADDR_LEN` is 1. It is compared against `PRF_ADDR_ENTRYn.ADDRESS` for the values of  $n$  where the entry is enabled for the received sync word. If there is a matching entry, this entry is used when receiving the packet; otherwise, reception is stopped and the device goes back to sync search. If `PRF_PKT_CONF.ADDR_LEN` is 0, the first entry that is enabled for the received sync word is used. If `PRF_TASK_CONF.MODE` is 00, the packet length is then read from `PRF_ADDR_ENTRYn.RXLENGTH`. This length includes the address, so it must be greater than or equal to the number of address bytes. If `PRF_TASK_CONF.MODE` is 01, the received length byte is compared against `PRF_ADDR_ENTRYn.RXLENGTH`. If it is greater than that value, reception is stopped and the device goes back to sync search.

If reception is stopped due to address mismatch or invalid length, the time-out given by `PRF_SEARCH_TIME` or Timer 2 event 2 still applies. If the first packet of the task is being received and `PRF_TASK_CONF.STOP_CONF` is 11, the next packet still counts as the first packet.

If a CRC field is present, it is checked using the polynomial configured in the BSP and the initialization value from `PRF_CRC_INIT`. The result of the CRC check is written in the MSB of the RES byte in the status field if a status field is configured. If the CRC is not correct and `PRF_FIFO_CONF.AUTOFLUSH_CRC` is 1, the LLE sends a discard RX FIFO command to remove the packet from the RX FIFO.

A packet where the length is equal to the address size contains no payload. Such a packet is known as an empty packet. If `PRF_FIFO_CONF.AUTOFLUSH_EMPTY` is 1 and an empty packet is received, the LLE sends a discard RX FIFO command to remove the empty packet from the RX FIFO.

Note that if the CRC length is 1 byte, the CRC check is not correct for empty packets if fixed length is configured or no address bytes are used.

If the RX FIFO becomes full while receiving a packet, the packet is discarded from the FIFO and no more bytes are stored in the RX FIFO, but the packet is received to its end. After that, it is checked whether the packet would be discarded from the RX FIFO anyway due to the setting of `PRF_FIFO_CONF`. If so, the task proceeds as normally. Otherwise, an `RXFIFOFULL` error interrupt is raised in lieu of the normal interrupt for received packets.

After receiving a packet, the LLE raises an interrupt to the MCU. Depending on CRC result and whether the packet was empty, the interrupts are generated as shown in [Table 25-16](#), provided an `RXFIFOFULL` interrupt is not raised as described previously. The table also shows which of the counters among the RAM registers are to be updated.

**Table 25-16. Interrupt and Counter Operation for Received Messages**

CRC Result	Payload Length > Address Length	Counter Incremented	Interrupt Raised
OK	No	<code>PRF_ADDR_ENTRYn.N_RXOK</code>	RXEMPTY
OK	Yes	<code>PRF_ADDR_ENTRYn.N_RXOK</code>	RXOK
NOK	No	<code>PRF_ADDR_ENTRYn.N_RXNOK</code>	RXNOK
NOK	Yes	<code>PRF_ADDR_ENTRYn.N_RXNOK</code>	RXNOK

An address entry should not be modified while the receiver is running. In order to modify, stop the receiver, modify the entry or entries, and restart the receiver.

### 25.9.2.3.2 Auto Mode

This section describes the receive operation if `PRF_TASK_CONF.MODE` is 10 or 11.

If `PRF_PKT_CONF.ADDR_LEN` is 1, the address byte is compared against `PRF_ADDR_ENTRYn.ADDRESS`, where  $n$  ranges from 0 to 7. It is compared against `PRF_ADDR_ENTRYn.ADDRESS` for the values of  $n$  where the entry is enabled for the received sync word. If there is a matching entry, this entry is used when receiving the packet, otherwise reception is stopped and the device goes back to sync search. If `PRF_PKT_CONF.ADDR_LEN` is 0, the first entry that is enabled for the received sync word is used.

Next, the 9-bit or 10-bit header is read. If `PRF_ADDR_ENTRYn.CONF.VARLEN` is 1, the length is fetched from the header and compared against `PRF_ADDR_ENTRYn.RXLENGTH`. If it is greater than that value, reception is stopped and the device goes back to sync search. If `PRF_ADDR_ENTRYn.CONF.VARLEN` is 0, the length field in the received header is ignored and the packet length is read from `PRF_ADDR_ENTRYn.RXLENGTH`. In both cases, the length is the number of bytes after the header and before the CRC. The length must be less than or equal to 63 for a 9-bit header and 127 for a 10-bit header. When a 10-bit header is used, the MCU must ensure that an entire packet can fit in the RX FIFO for auto ACK to be possible. This limits the maximum packet size based on the settings in `PRF_FIFO_CONF`.

If reception is stopped due to address mismatch or invalid length, the time-out given by `PRF_SEARCH_TIME` or Timer 2 event 2 still applies. If the first packet of the task is being received and `PRF_TASK_CONF.STOP_CONF` is 11, this still counts as the first packet.

If a CRC field is present, it is checked using the polynomial configured in the BSP and the initialization value from `PRF_CRC_INIT`. The result of the CRC is written in the MSB of the RES byte in the status field if a status field is configured. If the CRC is not correct and `PRF_FIFO_CONF.AUTOFLUSH_CRC` is 1, the LLE sends a discard RX FIFO command to remove the packet from the RX FIFO.

If the CRC is correct, the sequence number is checked against the sequence number stored in `PRF_ADDR_ENTRYn.SEQSTAT.SEQ`. If the sequence numbers are equal and `PRF_ADDR_ENTRYn.SEQSTAT.VALID` is 1, the two last received CRC bytes are compared against the 2 bytes in `PRF_ADDR_ENTRYn.LASTCRC`. If they are equal, the packet is determined to be a retransmission which can be ignored. If the CRC is 1 byte only, the received CRC byte is compared to `PRF_ADDR_ENTRYn.LASTCRC[0]` only, and if there is no CRC, the comparison is always viewed as equal. If the packet was a retransmission, the IGN bit of the RES byte in the status field is set if a status field is configured. After reception of a packet with CRC OK and which fits in the RX FIFO, `PRF_ADDR_ENTRYn.SEQSTAT.VALID` is set to 1, `PRF_ADDR_ENTRYn.SEQSTAT.SEQ` is set to the sequence number of the header of the received packet, and `PRF_ADDR_ENTRYn.LASTCRC` is set to the value of the last two received CRC bytes.

If the RX FIFO becomes full while receiving a packet, the packet is discarded from the FIFO and no more bytes are stored in the RX FIFO, but the packet is received to its end. After that, it is checked whether the packet would be discarded from the RX FIFO anyway due to the setting of `PRF_FIFO_CONF`. If so, the task proceeds as normally. Otherwise, an `RXFIFOFULL` error interrupt is raised, and no acknowledgment is transmitted. The sequence number is not updated so that a retransmission of the packet is not ignored.

If the received packet was not a retransmission and `PRF_ADDR_ENTRYn.SEQSTAT.ACK_PAYLOAD_SENT` is 1, the packet is seen as a confirmation of the last transmitted acknowledgment payload. If so, `PRF_ADDR_ENTRYn.SEQSTAT.ACK_PAYLOAD_SENT` is set to 0, a `TXDONE` interrupt is raised, and the `PRF_ADDR_ENTRYn.NTXDONE` counter is incremented. `PRF_ADDR_ENTRYn.ACKLENGTHk` is set to 0 for the `k` found in `PRF_ADDR_ENTRYn.SEQSTAT.NEXTACK`, and `PRF_ADDR_ENTRYn.SEQSTAT.NEXTACK` is inverted.

After receiving a packet, the LLE raises an interrupt to the MCU. Depending on the CRC result, the payload length, and whether the received packet is a retransmission to be ignored, the interrupts are generated as shown in [Table 25-17](#). The table also shows which of the counters among the RAM registers are to be updated.

**Table 25-17. Interrupt and Counter Operation for Received Messages**

CRC Result	Ignore	Length	Counter Incremented	Interrupt Raised
OK	No	> 0	<code>PRF_ADDR_ENTRYn.N_RXOK</code>	RXOK
OK	No	= 0	<code>PRF_ADDR_ENTRYn.N_RXOK</code>	RXEMPTY
OK	Yes	X	<code>PRF_ADDR_ENTRYn.N_RXIGNORED</code>	RXIGNORED
NOK	X	X	<code>PRF_ADDR_ENTRYn.N_RXNOK</code>	RXNOK

After reception of a packet, the next action is determined as follows:

- If the CRC of the received packet was not correct, the treatment of the packet is finished and the next action is as described in [Section 25.9.2.3.3](#).
- If `PRF_ADDR_ENTRYn.CONF.AA` is 0, the treatment of the packet is finished and the next action is as described in [Section 25.9.2.3.3](#).
- If the `NO_ACK` bit of the received header is 1 and the CRC was correct, the treatment of the packet is finished and the next action is as described in [Section 25.9.2.3.3](#).
- If the packet did not fit in the RX FIFO and was not otherwise to be discarded, the treatment of the packet is finished and the next action is as described in [Section 25.9.2.3.3](#).
- Otherwise, an acknowledgment is transmitted as described in the following text.

After receiving a packet where the CRC is correct and where an acknowledgment is supposed to be sent, the transmitter is configured. The transmission starts at a time given by the `PRF_RX_TX_TIME` register. Synthesizer recalibration is performed only if there is time. The LLE checks

`PRF_ADDR_ENTRYn.SEQSTAT.NEXTACK` to find  $k$ . If `PRF_ADDR_ENTRYn.ACKLENGTHk` is nonzero, payload is included the packet. In this case, `PRF_ADDR_ENTRYn.SEQSTAT.ACK_PAYLOAD_SENT` is set to 1 by the LLE. The transmitted packet has the same sync word and address as the received packet. If `PRF_ADDR_ENTRYn.CONF.TXLEN` is 0, the length field in the header is set equal to `PRF_ADDR_ENTRYn.ACKLENGTHk`. If `PRF_ADDR_ENTRYn.CONF.TXLEN` is 1, the length field is set to 11 0011 for a 9-bit header and to 011 0011 for a 10-bit header. Note that a value of 0 for `PRF_ADDR_ENTRYn.CONF.TXLEN` may be used regardless of the `VARLEN` setting in the peer device, as the length field is ignored for fixed length. A value of 1 must only be used if the peer is configured to use fixed length for the ACK payload, and should only be used with ACKs without payload. The sequence number is set to the value of `PRF_ADDR_ENTRYn.SEQSTAT.ACKSEQ`, and `NO_ACK` is set to 0. If there is payload, it is read from the buffer as described in [Section 25.8.3](#).

After the acknowledgement has been transmitted, `PRF_ADDR_ENTRYn.SEQSTAT.ACKSEQ` is incremented modulo 4, the `PRF_N_TX` counter is incremented, and the next action is as described in [Section 25.9.2.3.3](#).

### 25.9.2.3.3 Continuation and Ending of Receive Tasks

When a task ends, a `TASKDONE` interrupt is raised and an end cause is then available in `PRF_ENDCAUSE`.

After a packet has been received and potentially an acknowledgment has been transmitted, the next action depends on `PRF_TASK_CONF.REPEAT`. If this value is 0, the task ends. In this case, the `PRF_ENDCAUSE` register is set to `TASK_ENDOK`.

If `PRF_TASK_CONF.REPEAT` is 1, reception restarts. If `PRF_TASK_CONF.START_CONF` is 1, the LLE behaves as if the task was started again, with the LLE waiting for Timer 2 event 1; then starting to listen. If `PRF_TASK_CONF.START_CONF` is 0, the receiver restarts as soon as possible, as starting a new task (except for the behavior of Timer 2 event 2 if `PRF_TASK_CONF.STOP_CONF` is 11). In both cases, synthesizer recalibration is done if `PRF_TASK_CONF.REPEAT_CONF` is 0, otherwise not. Skipping synthesizer recalibration reduces the time before listening is restarted.

If a `CMD_SHUTDOWN` or a command starting a new task is observed while the task is running, it ends immediately with `TASK_ABORT` as the end cause. If the receiver or transmitter was running, an `RXTXABO` interrupt is also raised.

If `CMD_STOP` is received while in sync search, the task ends immediately with `TASK_RXTIMEOUT` as the end cause. If `CMD_STOP` is received while receiving or while transmitting an ACK or in the transition between those, the task ends with `TASK_STOP` as the end cause after the packet is fully received and (if ACK is to be sent) the ACK is sent. If `CMD_STOP` is received while waiting for Timer 2 event 1 to restart reception, the task ends immediately with `TASK_STOP` as the end cause.

If Timer 2 event 2 (either from Timer 2 or from `CMD_SEND_EVENT2`) is observed during the task, the behavior depends on `PRF_TASK_CONF.STOP_CONF`:

- 00: Nothing happens
- 01: Behaves as if a `CMD_STOP` was received
- 10: Behaves as if a `CMD_SHUTDOWN` was received

- 11: If received while in sync search for the first packet after the task was started, or if `PRF_TASK_CONF.START_CONF` is 1 while in sync search for any packet, the task ends immediately with `TASK_RXTIMEOUT` as the end cause. Otherwise, nothing happens.

In addition, the task can end due to an internal time-out as described in the beginning of [Section 25.9.2.1](#), or it can end due to an error condition. The full list of possible end causes is summarized in [Table 25-18](#).

**Table 25-18. End-of-Receive Tasks**

Condition	End-of-Task Cause	Comment
Received packet (and potentially sent ACK) with <code>PRF_TASK_CONF.REPEAT = 0</code>	<code>TASK_ENDOK</code>	
Received packet (and potentially sent ACK) with <code>PRF_TASK_CONF.REPEAT = 1</code> after having observed <code>CMD_STOP</code> or Timer 2 event 2 with <code>PRF_TASK_CONF.STOP_CONF = 01</code> and <code>PRF_TASK_CONF.REPEAT = 1</code>	<code>TASK_STOP</code>	
While in sync search, observed <code>CMD_STOP</code> or Timer 2 event 2 with <code>PRF_TASK_CONF.STOP_CONF = 01</code>	<code>TASK_RXTIMEOUT</code>	
Observed Timer 2 event 2 while in sync search of the first packet with <code>PRF_TASK_CONF.STOP_CONF = 11</code>	<code>TASK_RXTIMEOUT</code>	
Did not get sync in the time specified by <code>PRF_SEARCH_TIME</code>	<code>TASK_NOSYNC</code>	
Received command for starting new task or <code>CMD_SHUTDOWN</code> or observed Timer 2 event 2 with <code>PRF_TASK_CONF.STOP_CONF = 10</code>	<code>TASK_ABORT</code>	If transmitter was running or receiver was running and had obtained sync, an <code>RXTXABO</code> interrupt is also raised.
Received unknown command	<code>TASKERR_CMD</code>	<code>LLEERR</code> interrupt is also raised. If transmitter was running or receiver was running and had obtained sync, an <code>RXTXABO</code> interrupt is also raised.
Semaphore is not free when expected	<code>TASKERR_SEM</code>	Task ends without any radio operation. <code>LLEERR</code> interrupt is also raised.
Unpermitted value of RAM register	<code>TASKERR_PAR</code>	<code>LLEERR</code> interrupt is also raised.
For <code>PRF_TASK_CONF.MODE = 00</code> : <code>PRF_ADDR_ENTRYn.RXLENGTH</code> corresponding to the received address is smaller than address length	<code>TASKERR_PAR</code>	<code>LLEERR</code> interrupt is also raised.
For auto mode: <code>PRF_ADDR_ENTRYn.ACKLENGTGHm</code> for the ACK payload to be transmitted exceeded 32	<code>TASKERR_PAR</code>	<code>LLEERR</code> interrupt is also raised.

#### 25.9.2.4 Transmit Task

When a `CMD_TX` command is received, the LLE configures the radio on the channel given by `PRF_CHAN.FREQ` and starts transmitting the packet from the TX FIFO.

If the TX FIFO has no available data, the task ends with `TASKERR_TXFIFO` as the end cause. Otherwise, the number of bytes given by the length byte in the TX FIFO is read from the TX FIFO and transmitted or otherwise handled as described in following sections. No check of data availability is done after the length byte is read, so if the FIFO contains fewer bytes than indicated in the length field, a TX FIFO underflow interrupt is raised by the FIFO hardware.

##### 25.9.2.4.1 Basic Mode

This section describes the transmit operation if `PRF_TASK_CONF.MODE` is 00 or 01.

If `PRF_TASK_CONF.MODE` is 01, the length field is calculated from the length field in the FIFO and transmitted. It is up to the MCU to ensure that the calculated length field does not exceed 255. If `PRF_TASK_CONF.MODE` is 00, no length field is transmitted.

If an address is configured, it is found based on the setting in `PRF_FIFO_CONF.TX_ADDR_CONF`. It can be set to take the address from `PRF_ADDR_ENTRY0`, to read it from the TX FIFO (which for the transmitter is equivalent to not having an address configured), or to read an index  $n$  from the config byte in the FIFO and read the address from `PRF_ADDR_ENTRYn.ADDRESS`. The values of `ENA0` and `ENA1` in `PRF_ADDR_ENTRYn.CONF` are ignored for the transmitter; the primary sync word is always transmitted.

The payload (if any) is transmitted as given in the FIFO.

If configured, a CRC with the number of bytes given by `PRF_CRC_LEN` is transmitted at the end.

When a packet has been transmitted, the LLE sends a deallocate TX FIFO command if `PRF_ADDR_ENTRYn.REUSE` is 0. Otherwise, the MCU must issue either a deallocate TX FIFO (to send a new packet) or a retry TX FIFO (to reuse) before sending again. The `PRF_N_TX` counter is incremented. A TXDONE interrupt to the MCU is raised when the packet has been completely read out of the TX FIFO by the LLE. Note that due to modulator delay, CRC transmission and ramp-down, this will happen before the packet transmission is finished. The next action is as given in [Section 25.9.2.4.3](#).

#### 25.9.2.4.2 Auto Mode

This section describes the transmit operation if `PRF_TASK_CONF.MODE` is 10 or 11.

When a 10-bit header is used, the MCU must ensure that an entire packet can fit in the TX FIFO for auto retransmission to be possible. This limits the maximum packet size based on the settings in `PRF_FIFO_CONF`.

If an address is configured, it is the first byte transmitted. It is found based on the setting in `PRF_FIFO_CONF.TX_ADDR_CONF`. It can be set to take the address from `PRF_ADDR_ENTRY0`, to read it from the TX FIFO, or to read an index  $n$  from the config byte in the FIFO and read the address from `PRF_ADDR_ENTRYn.ADDRESS`. In other cases,  $n$  is always assumed to be 0 in the following text. The values of `ENA0` and `ENA1` in `PRF_ADDR_ENTRYn.CONF` are ignored for the transmitter; the primary sync word is always transmitted.

The 9-bit or 10-bit header is transmitted next. If `PRF_ADDR_ENTRYn.CONF.TXLEN` is 0, the length field is set to the number of payload bytes after the header, which is calculated from the length byte in the TX FIFO. If `PRF_ADDR_ENTRYn.CONF.TXLEN` is 1, the length field is set to 11 0011 for a 9-bit header and to 011 0011 for a 10-bit header. Note that a value of 0 for `PRF_ADDR_ENTRYn.CONF.TXLEN` may be used regardless of the `VARLEN` setting in the receiver, as a receiver configured to use fixed length ignores the length field. A value of 1 must only be used if the receiver is configured to use fixed length. The `NO_ACK` bit transmitted is set according to bit 5 of the config byte read from the TX FIFO if present, otherwise to 0. If `PRF_ADDR_ENTRYn.CONF.FIXEDSEQ` is 1, the `SEQ` bits transmitted are set equal to bits 6 and 7 of the config byte read from the FIFO. Otherwise, the `SEQ` bits are set to the value of `PRF_ADDR_ENTRYn.SEQSTAT.SEQ`.

The payload (if any) is transmitted as given in the FIFO.

If configured, a CRC with the number of bytes given by `PRF_CRC_LEN` is transmitted at the end.

When a packet has been transmitted, the `N_TX` counter is incremented.

After transmission of a packet, the action depends on `PRF_ADDR_ENTRYn.CONF.AA` and the `NO_ACK` bit in the transmitted header. If `PRF_ADDR_ENTRYn.CONF.AA = 0` or `NO_ACK = 1`, no acknowledgment is expected, and the action is as if a valid acknowledgment had been received.

If `PRF_ADDR_ENTRYn.CONF.AA` is 1 and the transmitted `NO_ACK` bit was 0, the LLE configures RX to listen for an acknowledgment. To listen for acknowledgment, the receiver is configured at a time given by the `PRF_TX_RX_TIME` register. Synthesizer recalibration is performed only if there is time. The unit looks for sync. The sync search times out at the time given by `PRF_SEARCH_TIME`. If sync is found, the packet is received into the RX FIFO. If `PRF_PKT_CONF.ADDR_LEN` is 1, the address byte is compared against `PRF_ADDR_ENTRYn.ADDRESS` for the  $n$  that was used in transmission. If not matching, reception is stopped.

Next, the 9-bit or 10-bit header is read. If `PRF_ADDR_ENTRYn.CONF.VARLEN` is 1, the length is fetched from the header and compared against `PRF_ADDR_ENTRYn.RXLENGTH`. The maximum allowed value of this register is 32. If the received length is greater than `PRF_ADDR_ENTRYn.RXLENGTH`, reception is stopped and the device goes back to sync search. If `PRF_ADDR_ENTRYn.CONF.VARLEN` is 0, the length field in the received header is ignored and the packet length is read from `PRF_ADDR_ENTRYn.RXLENGTH`, which should normally be 0 in this case. The length is the number of bytes after the header and before the CRC.

If a CRC field is present, it is checked using the polynomial configured in the BSP and the initialization value from `PRF_CRC_INIT`. The result of the CRC is written in the MSB of the RES byte in the status field if a status field is configured. If the CRC is not correct and `PRF_FIFO_CONF.AUTOFLUSH_CRC` is set, the LLE sends a discard RX FIFO command to remove the packet from the RX FIFO.

The received sequence number is written to the config byte of the RX FIFO if configured, but is otherwise ignored.

If the RX FIFO becomes full while receiving an acknowledgment packet, the packet is discarded from the FIFO and no more bytes are stored in the RX FIFO, but the packet is received to its end. After that, it is checked to see whether the packet would be discarded from the RX FIFO anyway due to the setting of `PRF_FIFO_CONF`. If so, the task proceeds as normally. Otherwise, the task ends after the packet is received and an `RXFIFOFULL` error interrupt is raised. In this case, the treatment of the packet is as if the acknowledgment were not successfully received. This means that the next time a transmit task is started, the packet is retransmitted so that the receiver retransmits the ACK payload.

After receiving an acknowledgment, the LLE raises an interrupt to the MCU. Depending on the CRC result, the payload length, and whether the received packet had the same sequence number as the transmitted one, the interrupts are generated as shown in [Table 25-19](#). It also shows which of the counters among the RAM registers are to be updated.

**Table 25-19. Interrupt and Counter Operation for Received ACK Packets**

CRC Result	Length	Counter Incremented	Interrupt Raised
OK	> 0	<code>PRF_ADDR_ENTRYn.N_RXOK</code>	RXOK
OK	= 0	<code>PRF_ADDR_ENTRYn.N_RXIGNORED</code>	RXEMPTY
NOK	X	<code>PRF_ADDR_ENTRYn.N_RXNOK</code>	RXNOK

If an acknowledgment was not received (because no sync was obtained in time, the address did not match, the sequence number was wrong, the CRC check failed, or the ACK did not fit in the RX FIFO and was not otherwise to be discarded) the LLE sends a retry TX FIFO command. If the number of retransmissions already performed (not including the original transmission) is equal to `PRF_RETRANS_CNT`, the task ends. Otherwise, the packet is retransmitted. The time from the end of the previous transmission to the start of the retransmission is given in units of 62.5 ns by `PRF_RETRANS_DELAY`.

If the received packet was a valid acknowledgment, or if a packet was completely read out of the TX FIFO and no acknowledgment was expected, the LLE sends a deallocate TX FIFO command if `PRF_ADDR_ENTRYn.REUSE` is 0. Otherwise, the MCU must issue either a deallocate TX FIFO (to send a new packet) or a retry TX FIFO (to reuse) before sending again. The `PRF_ADDR_ENTRYn.NTXDONE` counter is incremented. A `TXDONE` interrupt to the MCU is raised. If `PRF_ADDR_ENTRYn.CONF.FIXEDSEQ = 0`, `PRF_ADDR_ENTRYn.SEQSTAT.SEQ` is incremented by 1 modulo 4. The next action is as given in [Section 25.9.2.3.3](#).

If the task ends because of a maximum number of retransmissions, a retry TX FIFO command is sent before the task ends, and `PRF_ADDR_ENTRYn.SEQSTAT.SEQ` is not incremented. This means that by default, the packet retransmission is attempted in the next task. If this is not desired, the packet must be removed from the FIFO. This can be done either by issuing a `CMD_TXFIFO_RESET` (this also removes any subsequent packets in the TX FIFO), by reading out the packet using the `RFTXFRD` register and issuing a `CMD_TX_FIFO_DEALLOC` command, or by TX FIFO pointer manipulation ([Section 25.3.1.3](#)). `PRF_ADDR_ENTRYn.SEQSTAT.SEQ` should then be incremented by one. These operations should only take place between tasks (that is, while the LLE does not have `SEMAPHORE1`).



### 25.9.2.4.3 Continuation and Ending of Transmit Tasks

When a task ends, a TASKDONE interrupt is raised and an end cause is then available in PRF\_ENDCAUSE.

After a packet has been transmitted and potentially a valid acknowledgment has been received, the next action depends on PRF\_TASK\_CONF.REPEAT. If this value is 0, the task ends. In this case, the PRF\_ENDCAUSE register is set to TASK\_ENDOK.

If PRF\_TASK\_CONF.REPEAT is 1, the TX FIFO status is checked. If the TX FIFO has no available data, the task ends with TASK\_ENDOK as the end cause. Otherwise, transmission restarts. If PRF\_TASK\_CONF.START\_CONF is 1, it behaves as if the task was started again with the LLE waiting for Timer 2 event 1, then performing a synthesizer calibration and starting to transmit. If PRF\_TASK\_CONF.START\_CONF is 0, the transmitter restarts PRF\_TX\_DELAY after the end of the previously transmitted packet, with synthesizer recalibration only if there is enough time, but in other respects as starting a new task. The PRF\_TX\_DELAY register gives the wait time in units of 62.5 ns. If the value is too small to fulfill, the transmission starts as soon as possible.

If a CMD\_SHUTDOWN or a command starting a new task is observed while the task is running, it ends immediately with TASK\_ABORT as the end cause. If the transmitter or receiver was running, an RXTXABO interrupt is also raised.

If CMD\_STOP is received while transmitting a packet, waiting for transmission of another packet, waiting for an ACK, receiving an ACK, or in the transition between those, the task ends with TASK\_STOP as the end cause after the packet is fully transmitted and (if ACK is expected) the ACK is received or given up. If CMD\_STOP is received while waiting for Timer 2 event 1 to restart reception, the task ends immediately with TASK\_STOP as the end cause.

If Timer 2 event 2 (either from Timer 2 or from CMD\_SEND\_EVENT2) is observed during the task, the behavior depends on PRF\_TASK\_CONF.STOP\_CONF.

- 00: Nothing happens.
- 01: Behaves as if a CMD\_STOP was received
- 10: Behaves as if a CMD\_SHUTDOWN was received
- 11: Nothing happens.

In addition, the task can end for reasons described earlier, or it can end due to an error condition. The full list of possible end causes is summarized in [Table 25-20](#).

**Table 25-20. End-of-Transmit Tasks**

Condition	End-of-Task Cause	Comment
Transmitted packet (and potentially received ACK) with PRF_TASK_CONF.REPEAT = 0	TASK_ENDOK	
Transmitted packet (and potentially received ACK) and observed TX FIFO with no available data	TASK_ENDOK	
Transmitted packet (and potentially received ACK) after having observed CMD_STOP or Timer 2 event 2 with PRF_TASK_CONF.STOP_CONF = 01	TASK_STOP	
Did not get valid acknowledgment after having retransmitted the number of times given by PRF_RETRANS_CNT	TASK_MAXRT	
Observed empty TX FIFO when packet transmission is supposed to start, or TX FIFO is in an invalid state	TASKERR_TXFIFO	LLEERR and RXTXABO interrupts are also raised.
RX FIFO went overfull while receiving an ACK that was not otherwise to be discarded	TASKERR_RXFIFO	RXFIFOFULL interrupt is also raised.
Received command for starting new task or CMD_SHUTDOWN or observed Timer 2 event 2 with PRF_TASK_CONF.STOP_CONF = 10	TASK_ABORT	If transmitter was running, an RXTXABO interrupt is also raised.
Received unknown command	TASKERR_CMD	LLEERR interrupt is also raised. If transmitter or receiver was running and had obtained sync, an RXTXABO interrupt is also raised.

**Table 25-20. End-of-Transmit Tasks (continued)**

Condition	End-of-Task Cause	Comment
Semaphore is not free when expected	TASKERR_SEM	Task ends without any radio operation. LLEERR interrupt is also raised.
Invalid value of RAM register	TASKERR_PAR	LLEERR interrupt is also raised.
Length field to be transmitted exceeded maximum allowed value (255 in basic mode, 127 or 63 in auto mode with variable length)	TASKERR_PAR	LLEERR interrupt is also raised.

### 25.9.2.5 Transmit on Clear-Channel Task

When a `CMD_TX_ON_CC` command is received, the LLE configures the receiver on the channel given by `PRF_CHAN.FREQ`, but sync search is not enabled. The LLE polls the `RSSI` register every  $5.33\ \mu\text{s}$  and compares it to the value of `PRF_RSSI_LIMIT`. If a valid `RSSI` below the value of `PRF_RSSI_LIMIT` is observed more than `PRF_RSSI_COUNT` times in a row, the system starts transmitting. From there, the operation is as a normal transmit task, see [Section 25.9.2.4](#), except for the operation after a packet has been transmitted and potentially acknowledged, which is described in [Section 25.9.2.5.1](#).

#### 25.9.2.5.1 Continuation and Ending of Transmit on Clear-Channel Tasks

If `PRF_TASK_CONF.TX_ON_CC_CONF` is 1, the task ends if a valid `RSSI` value is not below the limit. If `PRF_TASK_CONF.TX_ON_CC_CONF` is 0, the device keeps listening until an `RSSI` below the given value is found.

If `PRF_TASK_CONF.STOP_CONF` is 11, Timer 2 event 2 may give a time-out while listening for a clear channel the first time, but not after the first transmission has been started.

If `PRF_TASK_CONF.TX_ON_CC_CONF` is 0, the clear-channel assessment must not be used as the only medium access control scheme in a multiuser environment, as this may cause all the devices to start transmission at the same time.

If retransmission is enabled, the LLE listens for acknowledgment and retransmits if needed as for normal TX tasks. However, if `PRF_TASK_CONF.REPEAT_CONF` is 0 after applying the retransmit delay, the device returns to listening, performing the same operation as when the task started, before possibly retransmitting.

If `PRF_TASK_CONF.REPEAT` is 0, the task ends after transmission as for normal TX tasks.

If `PRF_TASK_CONF.REPEAT` is 1, the TX FIFO status is checked. If the TX FIFO has no available data, the task ends with `TASK_ENDOK` as the end cause. Otherwise, transmission restarts. If `PRF_TASK_CONF.REPEAT_CONF` is 0, the task returns to listening, whereas if it is 1, the task restarts as if it were a standard transmit task. If `PRF_TASK_CONF.START_CONF` is 1, it behaves as if the task was started again with the LLE waiting for Timer 2 event 1, then performing a synthesizer calibration and starting to listen or transmit. If `PRF_TASK_CONF.START_CONF` is 0, the listening or transmission restarts `PRF_TX_DELAY` after the end of the previously transmitted packet, with synthesizer recalibration only if there is enough time, but in other respects as starting a new task.

If Timer 2 event 2 (either from Timer 2 or from `CMD_SEND_EVENT2`) is observed during the task, the behavior depends on `PRF_TASK_CONF.STOP_CONF`.

- 00: Nothing happens.
- 01: If received while transmitting a packet or waiting for or receiving an ACK or in the transition between those, the task ends with `TASK_STOP` as the end cause after the packet is fully transmitted and (if ACK is expected) the ACK is received or given up. If received while waiting for Timer 2 event 1 to restart reception, the task ends immediately with `TASK_STOP` as the end cause.
- 10: Behaves as if a `CMD_SHUTDOWN` was received
- 11: If received while listening for `RSSI` below the level before the first packet after the task was started, or if `PRF_TASK_CONF.START_CONF` is 1 while listening before any packet, the task ends immediately with `TASK_NOCC` as the end cause. Otherwise, nothing happens.

The task can end for all the same reasons as a normal transmit task summarized in [Table 25-20](#). In addition it can end for the reasons listed in [Table 25-21](#).

**Table 25-21. Additional Reasons for End-of-Transmit on Clear-Channel Tasks**

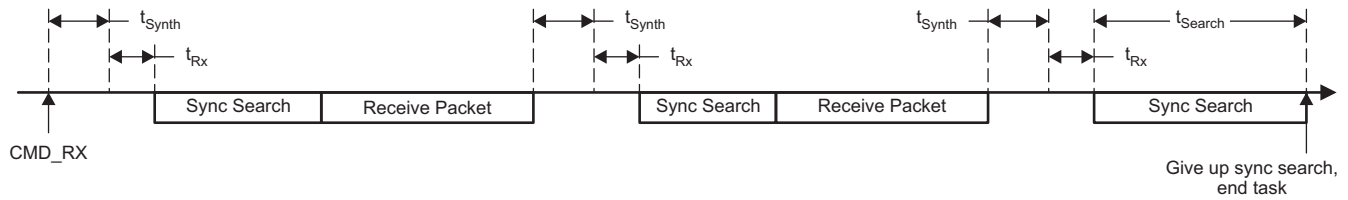
Condition	End-of-Task Cause	Comment
Observed valid RSSI above the threshold with <code>PRF_TASK_CONF.TX_ON_CC_CONF = 1</code>	TASK_NOCC	
Observed <code>CMD_STOP</code> or Timer 2 event 2 with <code>PRF_TASK_CONF.STOP_CONF = 01</code> or <code>11</code> while listening for RSSI below the threshold	TASK_NOCC	

### 25.9.2.6 Timing

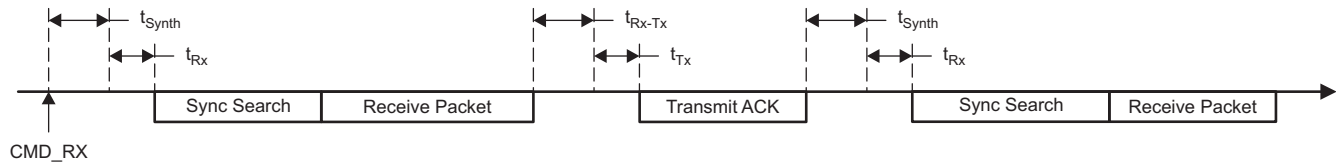
The timing in tasks is given in registers `PRF_TX_DELAY`, `PRF_RETRANS_DELAY`, `PRF_SEARCH_TIME`, `PRF_RX_TX_TIME`, and `PRF_TX_RX_TIME`. The first two of these registers are multiplied by 2 and then represent the number of 32-MHz samples, while the rest directly give the number of 32-MHz samples. `PRF_TX_DELAY` gives the time from the end of the previous transmission in the task to the start of the next transmission. Some examples of these delays are shown in [Figure 25-12](#) and [Figure 25-13](#) for RX and TX tasks, respectively. The time from the end of a received packet to the beginning of a transmitted packet is 130  $\mu$ s in an RX task with auto ACK.

When sync search takes place, either for receiving a normal packet or for receiving ACK, a time-out can be set up for when to give up the search. This time-out, given in 32-MHz cycles, is set up in the `PRF_SEARCH_TIME` register. Setting this register to 0 disables the time-out. In case of a time-out, the task ends for a normal sync search, or a packet is retransmitted in case of an ACK sync search.

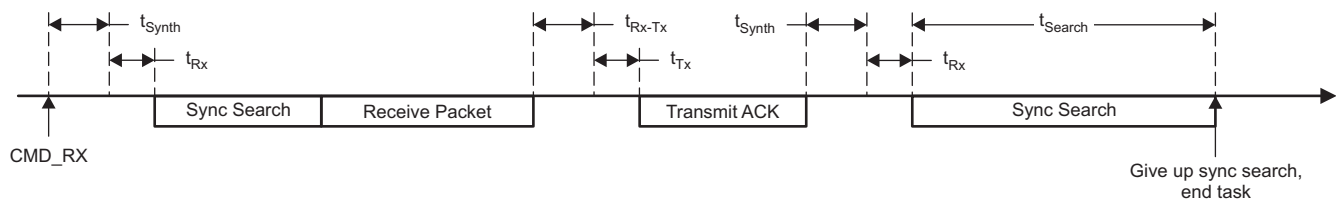
Receive task, PRF\_TASK\_CONF.MODE = 0X, .START\_CONF = 0, .REPEAT = 1:



Receive task, PRF\_TASK\_CONF.MODE = 1X, .START\_CONF = 0, .REPEAT = 1:



Receive task, PRF\_TASK\_CONF.MODE = 1X, .START\_CONF = 0, .REPEAT = 1:



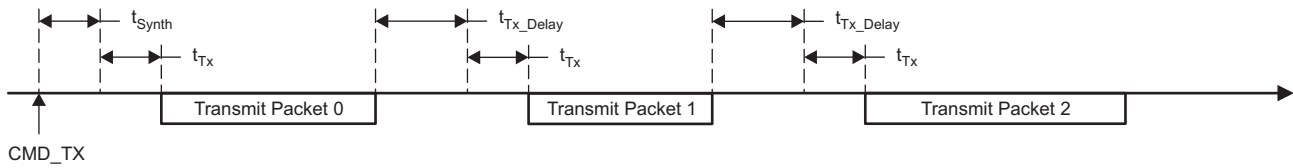
T0536-01

NOTE: The time given by PRF\_SEARCH\_TIME is denoted  $t_{Search}$  and the time given by PRF\_RX\_TX is denoted  $t_{Rx-Tx}$ . The setup and wait time for the synthesizer, receiver, and transmitter are denoted  $t_{Synth}$ ,  $t_{Tx}$ , and  $t_{Rx}$ , respectively.

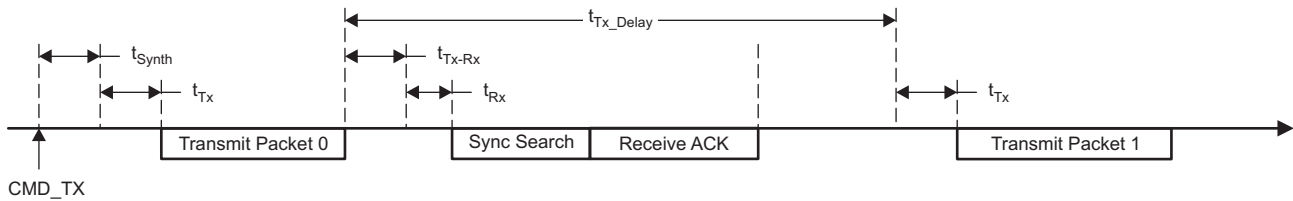
**Figure 25-12. Timing of Packets in RX Tasks**

For auto retransmit tasks, the time PRF\_RETRANS\_DELAY is the time from the end of a transmission to the retransmission of the packet in case an ACK is not found or there is a CRC error; see [Figure 25-13](#). The values of PRF\_SEARCH\_TIME and the maximum packet length in PRF\_PAYLOAD\_LEN should be set such that this time can always be achieved. If it is not possible to achieve the retransmission time, the packet is retransmitted as early as possible.

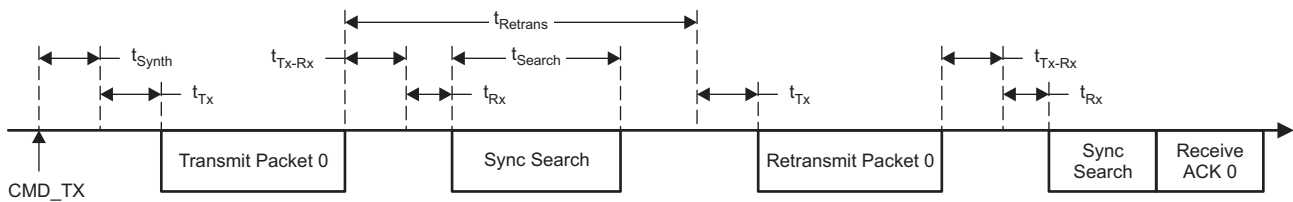
Transmit task, PRF\_TASK\_CONF.MODE = 0X, .START\_CONF = 0, .REPEAT = 1:



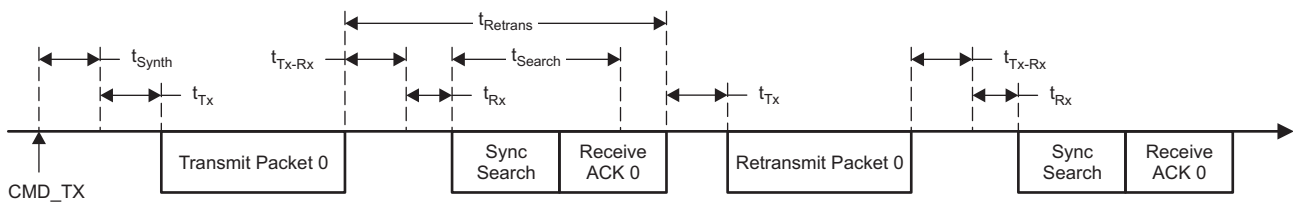
Transmit task, PRF\_TASK\_CONF.MODE = 0X, .START\_CONF = 0, .REPEAT = 1:



Transmit task, PRF\_TASK\_CONF.MODE = 1X, .START\_CONF = 0, .REPEAT = 1, no ACK found first time:



Transmit task, PRF\_TASK\_CONF.MODE = 1X, .START\_CONF = 0, .REPEAT = 1, CRC error on ACK:



T0537-01

NOTE: The time given by PRF\_TX\_DELAY is denoted  $t_{TX\_Delay}$ , the time given by PRF\_SEARCH\_TIME is denoted  $t_{search}$ , the time given by PRF\_RETRANS\_DELAY is denoted  $t_{Retrans}$ , and the time given by PRF\_TX\_RX is denoted  $t_{TX-RX}$ . The setup and wait times for the synthesizer, receiver, and transmitter are denoted  $t_{Synth}$ ,  $t_{TX}$ , and  $t_{RX}$ , respectively.

Figure 25-13. Timing of Packets in TX Tasks

### 25.9.3 RF Test Commands

In Table 25-12, there are listed some commands for modem test purposes. No registers are programmed by the LLE based on these commands, so all registers must be set up by the MCU. This includes the frequency word, which is otherwise written by the LLE; see Section 25.5 on how to program this. These commands take no parameters, and do not cause the LLE to create any interrupts or to write any end cause.

When receiving `CMD_DEMOD_TEST`, the LLE starts the receiver, but does not start sync search. The receiver runs until a `CMD_SHUTDOWN` command (or a command starting another task) is sent.

When receiving `CMD_RX_TEST`, the LLE starts the receiver and starts sync search. Any received data is discarded. The receiver runs until a `CMD_SHUTDOWN` command (or a command starting another task) is sent.

When receiving a `CMD_TX_TEST`, the LLE starts the transmitter and starts sending all zeros. The TX test command is normally combined with configuration of the modem to send a tone or by the BSP to send a whitening sequence. The transmitter runs until a `CMD_SHUTDOWN` command (or a command starting another task) is sent. In order to send random modulated data for test purposes, it is recommended to set `BSP_MODE` to `0x03` to enable both whiteners.

In order to send a continuous wave (CW), `MDMCTRL0.TX_IF` can be set to 1 before the `CMD_TX_TEST` command is issued. In this case, the radio outputs a tone with the frequency given in `MDMTEST1.TX_TONE`. In most cases, a tone at the synthesizer frequency is wanted (for example, to measure phase noise), in which case `MDMTEST.TX_TONE` should be set to `0x0A`. The frequency synthesizer must be programmed to the carrier frequency with no offset in this case; see [Section 25.5](#).

When receiving a `CMD_TX_FIFO_TEST`, the LLE starts the transmitter and starts sending from the TX FIFO; otherwise, the command is as `CMD_TX_TEST`. The MCU must feed data into the TX FIFO to avoid underflow, and the TX FIFO must be set up with auto commit and auto deallocate.

When receiving a `CMD_PING` command, the LLE responds with a `PINGRSP`. This can be used for checking that the LLE code is running.

## 25.10 Random Number Generation

The CC2541 has a hardware pseudo-random register, as explained in [Section 14.1](#). The RF core register bank provides a second interface to this register. Reading the `RFPSRND` register is equivalent to reading `RNDL`, then writing `01` to `ADCCON1.RCTL`.

For seeding the pseudo-random number generator and for tasks where higher entropy of the random numbers is needed, the radio can be used as a true-random generator. The register `RFRND` provides access to the least-significant bits of the radio ADC, which is random when noise is received. In order to get values on this register, the receiver must be turned on. The value in `RFRND` is updated every `0.17 μs`, so the sampling of that register must be slower than that in order to get a new value with every sample.

To get true random numbers, the following procedure can be followed:

1. Program `FREQCTRL` to a channel that is not likely to contain a narrow-band signal. A frequency outside the ISM band, such as a setting of `0`, is recommended.
2. Program `LNAGAIN` to `0` to have minimum reception of a signal on the air.
3. Start the receiver in test mode by issuing a `CMD_DEMOD_TEST` command.
4. Wait until ADC data are ready. This can be seen by the `RFRND` register having a value different from `0`.
5. Read the number of values needed from `RFRND`. Make sure that there is at least `0.17 μs` between each read (that is, at least 6 cycles if running on 32 MHz). For instance, to seed the pseudo-random generator, two values are needed.
6. Shut down the receiver by issuing a `CMD_SHUTDOWN` command.

The values read from the `RFRND` register do not have a perfectly uniform distribution. In order to improve this, several random numbers can be combined to produce one random number. One way of doing this is to use the pseudo-random generator in CRC mode and combine eight numbers into one. An example of how this can be done is given in the C code below:

```
// Store LNA gain setting and set minimum LNA gain
lnagain_stored = LNAGAIN;
LNAGAIN = 0x00;

// Set lowest possible frequency to avoid signals in ISM band
FREQCTRL = 0x00;

// Enable radio in RX without sync search
while (RFST != 0);
```

```

RFST = CMD_DEMOD_TEST;

// Wait for modem to be running
while (RFRND == 0);

// Seed RNG
RNDL = RFRND;
RNDL = RFRND;

for (j=0; j<ARRAY_SZ; j++) {
    // Read 8 random bytes into CRC generation
    RNDH = RFRND;
    RNDH = RFRND;
    RNDH = RFRND;
    RNDH = RFRND;
    RNDH = RFRND;
    RNDH = RFRND;
    RNDH = RFRND;
    RNDH = RFRND;
    // Read out LSB of CRC state
    rndarray[j] = RNDL;
}

// Shut down radio
while (RFST != 0);
RFST = CMD_SHUTDOWN;
// Restore LNA gain setting
LNAGAIN = lnagain_stored;

```

### 25.11 Packet Sniffing

Packet sniffing is a nonintrusive way of observing data that is either being transmitted or received. The packet sniffer outputs a clock and a data signal, which should be sampled on the rising edges of the clock. The two packet-sniffer signals are observable as GPIO outputs. For accurate time stamping, the SFD signal should also be output. The packet sniffer does not work for the 2 Mbps data rate.

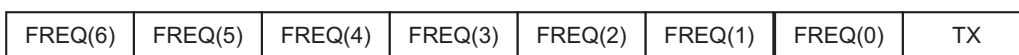
The packet-sniffer mode is selected in register `MDMCTRL3.RFC_SNIFF_CTRL`; see [Table 25-22](#) for a description of the different modes of operation.

**Table 25-22. Packet-Sniffer Modes of Operation**

MDMCTRL3.RFC_SNIFF_CTRL	Description
00	Packet sniffer disabled
01	Data output from BSP. TX data, including CRC, is whitened if the whitener is enabled. RX data, including CRC, is always de-whitened.
10	Data output from modulator. Only TX data before whitening is output. Any CRC bytes are 0.
11	Data output from the demodulator. Only RX data before de-whitening is output.

The packet-sniffer clock frequency is equal to the RF data rate. The data is output serially, in the received or transmitted order. It is possible to use a SPI slave to receive the data stream.

When a complete packet has been transferred, the packet sniffer appends a status byte that tells which value of the `FREQCTRL` register was used and if it was a TX or RX packet (bit 0 is high if it was a TX packet). The appended byte is formatted as follows (first transmitted bit to the left):



R0015-01

**Figure 25-14. Complete Appended Packet**

This allows for the external receiver to differentiate between RX and TX packets.

To set up the packet-sniffer signals or some of the other RF core observation outputs (in total maximum 3; rfc\_obs\_sig0, rfc\_obs\_sig1, and rfc\_obs\_sig2), the user must perform the following steps:

**Step1:** Determine which signal (RFC\_OBS\_CTRL[0-2]) to output on which GPIO pin (P1[0:5]). This is done using the OBSSELx control registers (OBSSEL0-OBSSEL5) that control the observation output to the pins P1[0:5] (overriding the standard GPIO behavior for those pins).

**Step2:** Set the (RFC\_OBS\_CTRL[0-2]) control registers to select the correct signals (rfc\_obs\_sig); for example, for packet sniffing one needs the rfc\_sniff\_data for the packet-sniffer data signal and rfc\_sniff\_clk for the corresponding clock signal.

**Step3:** Enable the packet-sniffer module in the MDMCTRL3 register.

## 25.12 Registers

### 25.12.1 Register Overview

#### 25.12.1.1 SFR Registers

- 1 - RFIRQF0 (0xE9) RF interrupt flags
- 2 - RFIRQF1 (0x91) RF interrupt flags
- 3 - RFERRF (0xBF) RF error interrupt flags
- 4 - RFD (0xD9) RF data
- 5 - RFST (0x6189) LLE and FIFO commands

#### 25.12.1.2 XREG Registers

**Table 25-23. XREG Register Overview**

Address (Hex)	+ 0x0000	+ 0x001	+ 0x002	+ 0x003
0x6180	FRMCTRL0	RFIRQM0	RFIRQM1	RFERRM
0x6184	FREQCTRL	FREQTUNE	TXPOWER	TXCTRL
0x6188	LLESTAT		SEMAPHORE0	SEMAPHORE1
0x618C	SEMAPHORE2	RFSTAT	RSSI	RFPSRND
0x6190	MDMCTRL0	MDMCTRL1	MDMCTRL2	MDMCTRL3
0x6194	SW_CONF	SW0	SW1	SW2
0x6198	SW3	FREQEST	RXCTRL	FSCTRL
0x619C				
0x61A0	LNAGAIN	AAFGAIN	ADCTEST0	
0x61A4		MDMTEST0	MDMTEST1	
0x61A8		ATEST		
0x61AC			RFC_OBS_CTRL0	RFC_OBS_CTRL1
0x61B0	RFC_OBS_CTRL2	LLECTRL		
0x61BC	TXFILTCFG			RFRND
0x61C0	RFRAMCFG			RFFDMA0
0x61C4	RFFDMA1	RFFSTATUS	RFFCFG	
0x61C8	RFRXFLEN	RFRXFTHRS	RFRXFWR	RFRXFRD
0x61CC	RFRXFWP	RFRXFRP	RFRXFSWP	RFRXFSRP
0x61D0	RFTXFLEN	RFTXFTHRS	RFTXFWR	RFTXFRD
0x61D4	RFTXFWP	RFTXFRP	RFTXFSWP	RFTXFSRP



**Table 25-23. XREG Register Overview (continued)**

Address (Hex)	+ 0x0000	+ 0x001	+ 0x002	+ 0x003
0x61E0	BSP_P0	BSP_P1	BSP_P2	BSP_P3
0x61E4	BSP_D0	BSP_D1	BSP_D2	BSP_D3
0x61E8	BSP_W	BSP_MODE	BSP_DATA	
0x61F8	SW4	SW5	SW6	SW7
0x61FC	DC_I_L	DC_I_H	DC_Q_L	DC_Q_H

### 25.12.2 Register Settings Update

This section contains a summary of the register settings that should be updated from their default value to have optimal performance. For some of the registers, the setting depends on the required gain in the receiver chain for bit rates of 1 Mbps and lower. For 2 Mbps, other values are needed, and different values should be used for RX and TX tasks. Note that registers that are listed only in the 2-Mbps table should have their default values when operating at 1 Mbps and lower, and vice versa.

**Table 25-24. Registers That Should Be Updated From Their Default Value, Bit Rates 1 Mbps and Lower**

Register Name	Address (Hex)	Standard Gain: New Value (Hex) <sup>(1)</sup>	High Gain: New Value (Hex)	Description
FRMCTRL0	6180	43	43	Amplitude weight in frequency offset compensation (assuming sync word included in CRC and MSB first)
MDMCTRL1	6191	48	48	Sync word correlation threshold (32-bit sync word)
MDMCTRL2	6192	C0	C0	Use inverse of preamble for frequency offset estimation (assuming MSB first)
MDMCTRL3	6193	63	63	Set RSSI mode to peak detect after sync
RXCTRL	619A	33	3F	Receiver currents
FSCTRL	619B	55	5A	Prescaler and mixer currents
LNAGAIN	61A0	3A	7F	LNA gain
TXFILTCFG	61BC	03	03	Sets TX anti-aliasing filter to appropriate bandwidth
TXPOWER	6186	E1	E1	TX power (0 dBm)
TXCTRL	6187	19	19	DAC current
IVCTRL	6265	1B	1B	PA, mixer, and DAC bias

<sup>(1)</sup> Not all modulation types are characterized for the standard gain setting; see the *CC2541 2.4-GHz Bluetooth low energy and Proprietary System-on-Chip* data sheet ([SWRS110](#)).

**Table 25-25. Registers That Should Be Updated From Their Default Value, Bit Rate 2 Mbps**

Register Name	Address (Hex)	RX Tasks	TX Tasks	Description
FRMCTRL0	6180	43	43	Amplitude weight in frequency offset compensation (assuming sync word is included in CRC and MSB first)
MDMCTRL1	6191	48	48	Sync word correlation threshold (32-bit sync word)
MDMCTRL2	6192	CC	CC	Use inverse of preamble for frequency offset estimation (assuming MSB first); set extra preamble bytes
MDMCTRL3	6193	63	63	Set RSSI mode to peak detect after sync
RXCTRL	619A	29	29	Receiver currents

**Table 25-25. Registers That Should Be Updated From Their Default Value, Bit Rate 2 Mbps (continued)**

Register Name	Address (Hex)	RX Tasks	TX Tasks	Description
FSCTRL	619B	5A	5A	Prescaler and mixer currents
ADCTEST0	61A2	66	66	Reduce ADC gain
MDMTEST0	61A5	0F	01	Select dc offset compensation method; change RSSI averaging
TXFILTCFG	61BC	03	03	Transmit filter bandwidth
PRF_PKT_CONF	6003	06	06	Enable AGC and start tone
PRF_RADIO_CONF	607E	90	D0	Set 1-MHz TX IF and dc write-back; for TX tasks also special dc offset compensation
TXPOWER	6186	E1	E1	TX power (0 dBm)
TXCTRL	6187	19	19	DAC current
IVCTRL	6265	1B	1B	PA, mixer, and DAC bias

The values for FRMCTRL0, MDMCTRL2, and PRF\_PKT\_CONF may require further adjustment based on the frame format, and the correlation threshold in MDMCTRL1 should be adjusted according to the sync word length, see [Section 25.7](#).

In addition to these modifications, registers must be set in order to set up the modulation format, packet handling, and so forth, as explained throughout this chapter.

### 25.12.3 SFR Register Descriptions

#### RFIRQF0 (0xE9) – RF Interrupt Flags

Bit No.	Name	Reset	R/W	Description
7:4	–	0000	R0	Reserved
3	RXTHSHUP	0	R/W0	RX FIFO goes above its upper threshold. 0: No interrupt pending 1: Interrupt pending
2	TXTHSHUP	0	R/W0	TX FIFO goes above its upper threshold. 0: No interrupt pending 1: Interrupt pending
1	RXTHSHDN	0	R/W0	RX FIFO goes below its lower threshold. 0: No interrupt pending 1: Interrupt pending
0	TXTHSHDN	0	R/W0	TX FIFO goes below its lower threshold. 0: No interrupt pending 1: Interrupt pending

**RFIRQF1 (0x91) – RF Interrupt Flags**

Bit No.	Name	Reset	R/W	Description
7	PINGRSP	0	R/W0	When receiving a CMD_PING command, the LLE responds with a PINGRSP. This can be used for checking that the LLE is running. 0: No interrupt pending 1: Interrupt pending
6	TASKDONE	0	R/W0	TX FIFO packet completed 0: No interrupt pending 1: Interrupt pending
5	TXDONE	0	R/W0	Task ended 0: No interrupt pending 1: Interrupt pending
4	RXEMPTY	0	R/W0	Empty packet received 0: No interrupt pending 1: Interrupt pending
3	RXIGNORED	0	R/W0	Packet received with unexpected sequence number 0: No interrupt pending 1: Interrupt pending
2	RXNOK	0	R/W0	Packet received with CRC error 0: No interrupt pending 1: Interrupt pending
1	TXFLUSHED	0	R/W0	TX ACK buffer flushed 0: No interrupt pending 1: Interrupt pending
0	RXOK	0	R/W0	Packet received correctly 0: No interrupt pending 1: Interrupt pending

**RFERRF (0xBF) – RF Error Interrupt Flags**

Bit No.	Name	Reset	R/W	Description
7	–	0	R/W0	Reserved
6	RXFIFOFULL	0	R/W0	RX FIFO is full when trying to store received data 0: No interrupt pending 1: Interrupt pending
5	LLEERR	0	R/W0	LLE command or parameter error 0: No interrupt pending 1: Interrupt pending
4	RXTXABO	0	R/W0	Receive or transmit operation aborted 0: No interrupt pending 1: Interrupt pending
3	RXOVERF	0	R/W0	RX FIFO overflow 0: No interrupt pending 1: Interrupt pending
2	TXOVERF	0	R/W0	TX FIFO overflow 0: No interrupt pending 1: Interrupt pending
1	RXUNDERF	0	R/W0	RX FIFO underflow 0: No interrupt pending 1: Interrupt pending
0	TXUNDERF	0	R/W0	TX FIFO underflow 0: No interrupt pending 1: Interrupt pending

**RFD (0xD9) – RF data**

Bit No.	Name	Reset	R/W	Description
7:0	RFD [ 7 : 0 ]	0x00	R/W	Data written to the register is written to the TX FIFO. When reading this register, data from the RX FIFO is read.

**RFST (0xE1) – LLE and FIFO commands**

Bit No.	Name	Reset	R/W	Description
7:0	RFST[7:0]	0x00	R/WH0	Commands to radio are written to this register. The register is cleared (set to 0x00) when the radio is ready for a new command.

**25.12.3.1 XREG Register Descriptions**
**FRMCTRL0 (0x6180) – Frame Control**

Bit No.	Name	Reset	R/W	Description
7:5	FOC_MAGN_CONT[2:0]	000	R/W	Controls how signal amplitude is weighted into the frequency offset compensation scheme. 000: Magnitude has no effect 001 to 111: Low-to-high weighting of the magnitude
4:2	–	000	R/W	Reserved always write 0.
1	SW_CRC_MODE	0	R/W	0: The sync word is not included in the CRC calculation. 1: The sync word is included in the CRC calculation. Only to be used with whitening disabled.
0	ENDIANNESS	0	R/W	0: The data goes LSB-first over the air. 1: The data goes MSB-first over the air.

**RFIRQM0 (0x6181) – RF Interrupt Masks**

Bit No.	Name	Reset	R/W	Description
7:4	–	0000	R0	Reserved
3	RXTHSHUP	0	R/W	RX FIFO goes above its upper threshold. 0: Interrupt disabled 1: Interrupt enabled
2	TXTHSHUP	0	R/W	TX FIFO goes above its upper threshold. 0: Interrupt disabled 1: Interrupt enabled
1	RXTHSHDN	0	R/W	RX FIFO goes below its lower threshold. 0: Interrupt disabled 1: Interrupt enabled
0	TXTHSHDN	0	R/W	TX FIFO goes below its lower threshold. 0: Interrupt disabled 1: Interrupt enabled

**RFIRQM1 (0x6182) – RF Interrupt Masks**

Bit No.	Name	Reset	R/W	Description
7	PINGRSP	0	R/W	When receiving a CMD_PING command, the LLE responds with a PINGRSP. This can be used for checking that the LLE is running. 0: Interrupt disabled 1: Interrupt enabled
6	TASKDONE	0	R/W	TX FIFO packet completed 0: Interrupt disabled 1: Interrupt enabled
5	TXDONE	0	R/W	Task ended 0: Interrupt disabled 1: Interrupt enabled
4	RXEMPTY	0	R/W	Empty packet received 0: Interrupt disabled 1: Interrupt enabled
3	RXIGNORED	0	R/W	Packet received with unexpected sequence number 0: Interrupt disabled 1: Interrupt enabled
2	RXNOK	0	R/W	Packet received with CRC error 0: Interrupt disabled 1: Interrupt enabled
1	TXFLUSHED	0	R/W	TX ACK buffer flushed 0: Interrupt disabled 1: Interrupt enabled
0	RXOK	0	R/W	Packet received correctly 0: Interrupt disabled 1: Interrupt enabled

**RFERRM (0x6183) – RF Error Interrupt Masks**

Bit No.	Name	Reset	R/W	Description
7	–	0	R/W	Reserved
6	RXFIFOFULL	0	R/W	RX FIFO is full when trying to store received data. 0: Interrupt disabled 1: Interrupt enabled
5	LLEERR	0	R/W	LLE command or parameter error 0: Interrupt disabled 1: Interrupt enabled
4	RXTXABO	0	R/W	Receive or transmit operation aborted 0: Interrupt disabled 1: Interrupt enabled
3	RXOVERF	0	R/W	RX FIFO overflow 0: Interrupt disabled 1: Interrupt enabled
2	TXOVERF	0	R/W	TX FIFO overflow 0: Interrupt disabled 1: Interrupt enabled
1	RXUNDERF	0	R/W	RX FIFO underflow 0: Interrupt disabled 1: Interrupt enabled
0	TXUNDERF	0	R/W	TX FIFO underflow 0: Interrupt disabled 1: Interrupt enabled

**FREQCTRL (0x6184) – Synthesizer Frequency Control**

Bit No.	Name	Reset	R/W	Description
7	–	0	R0	Reserved. Read as zero
6:0	FREQ[6:0]	0x16	R/W	Frequency control word. Controls frequency of local oscillator. See <a href="#">Section 25.5</a> for the relation between the LO frequency and the RF frequency. $f_{LO} = (2379 + \text{FREQ}[6:0]) \text{ MHz}$ The frequency word in FREQ[6:0] is an offset value from 2379 MHz. The device supports the frequency range from 2379 MHz to 2495 MHz. The usable settings for FREQ[6:0] are consequently 0 to 116. Settings outside this range (117–127) give a frequency of 2495 MHz.

**FREQTUNE (0x6185) – Crystal Oscillator Frequency Tuning**

Bit No.	Name	Reset	R/W	Description
7:4	–	0x0	R0	Reserved. Read as zero
3:0	XOSC32M_TUNE[3:0]	0xF	R/W	Tune crystal oscillator. The default setting of 1111 leaves the XOSC not tuned. Changing setting from default switches in extra capacitance to the oscillator, effectively lowering the XOSC frequency. Hence, a higher setting gives a higher frequency.

**TXPOWER (0x6186) – Output Power Control**

Bit No.	Name	Reset	R/W	Description
7:0	PA_POWER[7:0]	0xF5	R/W	PA power control. NOTE: This value should be updated. See the device data sheet (Appendix C) for recommended values.

**TXCTRL (0x6187) – TX Settings**

Bit No.	Name	Reset	R/W	Description
7	–	0	R0	Reserved
6	–	1	R/W	Reserved
5:4	DAC_CURR[1:0]	10	R/W	Change the current in the DAC to change the current steps
3:2	DAC_DC[1:0]	10	R/W	Adjusts the dc level to the TX mixer.
1:0	TXMIX_CURRENT[1:0]	01	R/W	Transmit mixers core current: Current increases with increasing setting.

**LLESTAT (0x6188) – LLE Status**

Bit No.	Name	Reset	R/W	Description
7:5	–	000	R0	Reserved
4	AGC_LOWGAIN	0	R	1 if the AGC algorithm has reduced the front-end gain; 0 otherwise
3	WAIT_T2E1	0	R	Indication on the LLE waiting for Timer 2 event 1 to start a task 0: Not waiting for Timer 2 event 1 1: Command processed, event 1 not yet received
2	LLE_IDLE	0	R	Link-layer engine idle 0: The LLE is busy processing or finishing a command, or in reset. 1: The LLE is idle waiting for a command to start a new task.
1	SYNC_SEARCH	0	R	RX search for sync 0: The modem is not ready to receive a packet 1: The modem is in search for a sync word or is receiving a packet
0	VCO_ON	0	R	VCO on 0: The VCO is powered down, so the next RX or TX operation must start and calibrate the synthesizer before transmitting or receiving 1: The VCO is powered up. If the LLE is idle, it means the next task starts quickly if frequency programming is disabled (PRF_CHAN . FREQ = 127)

**SEMAPHORE0 (0x618A) – Semaphore for Accessing RF Data Memory**

Bit No.	Name	Reset	R/W	Description
7:1	–	0000 000	R0	Reserved, read as 0
0	SEMAPHORE	1	R/W1	When SEMAPHORE = 1 and SEMAPHORE0 is read, SEMAPHORE is set to 0. SEMAPHORE can only be set to 1 by a reset or by writing 1 to it.

**SEMAPHORE1 (0x618B) – Semaphore for Accessing RF Data Memory**

Bit No.	Name	Reset	R/W	Description
7:1	–	0000 000	R0	Reserved, read as 0
0	SEMAPHORE	1	R/W1	When SEMAPHORE = 1 and SEMAPHORE1 is read, SEMAPHORE is set to 0. SEMAPHORE can only be set to 1 by a reset or by writing 1 to it.

**SEMAPHORE2 (0x618C) – Semaphore**

Bit No.	Name	Reset	R/W	Description
7:1	–	0000 000	R0	Reserved, read as 0
0	SEMAPHORE	1	R/W1	When SEMAPHORE = 1 and SEMAPHORE2 is read, SEMAPHORE is set to 0. SEMAPHORE can only be set to 1 by a reset or by writing 1 to it.

**RFSTAT (0x618D) – RF Core Status**

Bit No.	Name	Reset	R/W	Description
7	MOD_UNDERFLOW	0	R/W0	Modulator has underflowed. Must be cleared by software
6:5	DEM_STATUS	00	R	Demodulator status 00: Idle 01: Active 10: Finishing 11: Error
4	SFD	0	R	High when the sync word has been sent in TX or when sync has been obtained in RX
3	CAL_RUNNING	0	R	Frequency synthesizer calibration status. 0: Calibration done or not started 1: Calibration in progress
2	LOCK_STATUS	0	R	1 when PLL is in lock; 0 otherwise
1	TX_ACTIVE	0	R	Status signal, active when the LLE is in one of the transmit states
0	RX_ACTIVE	0	R	Status signal, active when the LLE is in one of the receive states

**RSSI (0x618E) – Received Signal Strength Indicator**

Bit No.	Name	Reset	R/W	Description
7:0	RSSI_VAL[7:0]	0x80	R	RSSI estimate on a logarithmic scale. Unit is 1 dB; offset depends on the gain of the RX chain, including external components; see the device data sheet. The reset value of 0x80 also indicates that the RSSI value is invalid or the measurement is not yet complete.

**RFPSRND (0x618F) – Pseudorandom Number Generator**

Bit No.	Name	Reset	R/W	Description
7:0	RNG_DOUT[7:0]	0x00	R	The value read from the pseudorandom number generator, see <a href="#">Chapter 14</a> . Reading this register causes the shift register to be updated with 13 times rollout.

**MDMCTRL0 (0x6190) – Modem Configuration**

Bit No.	Name	Reset	R/W	Description
7:6	FOC_DECAY[1:0]	00	R/W	Controls decay ratio of frequency offset compensation mechanism. Value by which to increment preamble cost function at each decay 00: 8 01: 16 10: 32 11: 64
5	TX_IF	0	R/W	0: Modulation is done at an IF set by rfr_tx_tone. 1: Modulator outputs tone set by rfr_tx_tone.
4:1	MODULATION[3:0]	0010	R/W	Modulation scheme 0010: GFSK 250-kHz deviation, 1-Mbps data rate 0011: GFSK 500-kHz deviation, 2-Mbps data rate 0100: GFSK 160-kHz deviation, 250-kbps data rate 0110: GFSK 160-kHz deviation, 1-Mbps data rate 0111: GFSK 320-kHz deviation, 2-Mbps data rate 1000: MSK, 250-kbps data rate 1001: MSK, 500-kbps data rate Others: Reserved
0	PHASE_INVERT	0	R/W	Set one of two RF modulation modes for RX or TX 0: Normal (binary 0 represented with negative frequency deviation, binary 1 represented with positive frequency deviation) 1: Inverted phase (binary 0 represented with positive frequency deviation, binary 1 represented with negative frequency deviation)

**MDMCTRL1 (0x6191) – Modem Configuration**

Bit No.	Name	Reset	R/W	Description
7:6	FOC_MODE	01	R/W	Frequency-offset average filter behavior 00: No frequency-offset compensation done 01: Freeze frequency-offset estimate after sync 10: Continuously estimate and remove frequency offset 11: Freeze the frequency-offset estimate after sync, use double decay rate
5	–	0	R0	Reserved
4:0	CORR_THR[4:0]	0 1111	R/W	Demodulator correlator threshold value, used in sync search. Optimal threshold value depends on SW_CONF . SW_LEN. CORR_THR adjusts how the receiver synchronizes to data from the radio. If threshold is set too low, sync can more easily be found on noise. If set too high, the sensitivity is reduced but sync is not likely to be found on noise.



**MDMCTRL2 (0x6192) – Modem Configuration**

Bit No.	Name	Reset	R/W	Description
7	SW_BIT_ORDER	0	R/W	0: The sync word is transmitted LSB to MSB (from SYNC_WORD[ 0 ] to SYNC_WORD[ 31 ]), and in receive the correlator expects this bit ordering. 1: The sync word is transmitted MSB to LSB (from SYNC_WORD[ 31 ] to SYNC_WORD[ 0 ]), and in receive the correlator expects this bit ordering.
6	DEM_PREAM_MODE	0	R/W	Use PREAM_SEL[1:0] or 1s complement of PREAM_SEL[1:0] for frequency offset estimation. 0: PREAM_SEL[1:0] 1: 1s complement of PREAM_SEL[1:0]
5:4	PREAM_SEL[1:0]	00	R/W	00: Select preamble based on first bit of sync word; last bit of preamble is inverse of first bit of sync word. 01: Select preamble based on first bit of sync word; last bit of preamble is same as first bit of sync word. 10: Use preamble 0101 0101 11: Use preamble 1010 1010
3:0	NUM_PREAM_BYTES[3:0]	0000	R/W	The number of preamble bytes to be sent in TX mode prior to the sync word 0000: 1 leading preamble byte 0001: 2 leading preamble bytes 0010: 3 leading preamble bytes 0011: 4 leading preamble bytes ... 1111: 16 leading preamble bytes

**MDMCTRL3 (0x6193) – Modem Configuration**

Bit No.	Name	Reset	R/W	Description
7:6	SYNC_MODE[1:0]	01	R/W	00: Correlation value above threshold is sufficient as sync criterion. 01: Correlation value above threshold and data decision on all symbols of sync word is used as sync criterion. 10: Correlation value above threshold and data decision on all symbols of sync word is used as sync criterion. Accept one bit error in sync word 11: Reserved
5	RAMP_AMP	1	R/W	1: Enable ramping of DAC output amplitude during start-up and finish. 0: Disable ramping of DAC output amplitude.
4:3	RFC_SNIFF_CTRL[1:0]	00	R/W	Enable and disable <i>rfc_sniff</i> . 00: Sniffer disabled 01: Output data out of the BSP 10: Output data out of the modulator before the BSP 11: Output data out of the demodulator before the BSP
2	–	0	R0	Reserved. Read as 0.
1:0	RSSI_MODE[1:0]	00	R/W	Controls mode of RSSI 00 : Continuous mode 01 : Freeze estimate at sync 10 : Peak detect 11 : Continuous before sync, peak detect after sync

**SW\_CONF (0x6194) – Sync Word Configuration**

Bit No.	Name	Reset	R/W	Description
7	DUAL_RX	0	R/W	0: Only search for primary SW 1: Search for both primary and secondary SW
6	–	0	R/W	Reserved. Always write 0.
5	SW_RX	0	R	0: Primary SW received 1: Secondary SW received Valid only when RFSTAT . SFD is 1
4:0	SW_LEN[ 4 : 0 ]	0 0000	R/W	Determines how many of the bits in SYNC_WORD are to be used. This allows for arbitrary sync word lengths. 0 0000: 32-bit SW 0 0001 to 0 11111: Reserved 1 0000: 16-bit SW 1 0001: 17-bit SW 1 0010: 18-bit SW 1 0011: 19-bit SW ... 1 1111: 31-bit SW

**SW0 (0x6195) – Primary Sync Word Byte 0**

Bit No.	Name	Reset	R/W	Description
7:0	SYNC_WORD[ 7 : 0 ]	0x00	R/W	Contains bits 7:0 of the primary synchronization word

**SW1 (0x6196) – Primary Sync Word Byte 1**

Bit No.	Name	Reset	R/W	Description
7:0	SYNC_WORD[ 15 : 8 ]	0x00	R/W	Contains bits 15:8 of the primary synchronization word

**SW2 (0x6197) – Primary Sync Word Byte 2**

Bit No.	Name	Reset	R/W	Description
7:0	SYNC_WORD[ 23 : 16 ]	0x00	R/W	Contains bits 23:16 of the primary synchronization word

**SW3 (0x6198) – Primary Sync Word Byte 3**

Bit No.	Name	Reset	R/W	Description
7:0	SYNC_WORD[ 31 : 24 ]	0x00	R/W	Contains bits 31:24 of the primary synchronization word

**SW4 (0x61F8) – Secondary Sync Word Byte 0**

Bit No.	Name	Reset	R/W	Description
7:0	SYNC_WORD2[ 7 : 0 ]	0x00	R/W	Contains bits 7:0 of the secondary synchronization word

**SW5 (0x61F9) – Secondary Sync Word Byte 1**

Bit No.	Name	Reset	R/W	Description
7:0	SYNC_WORD2[ 15 : 8 ]	0x00	R/W	Contains bits 15:8 of the secondary synchronization word

**SW6 (0x61FA) – Secondary Sync Word Byte 2**

Bit No.	Name	Reset	R/W	Description
7:0	SYNC_WORD2[ 23 : 16 ]	0x00	R/W	Contains bits 23:16 of the secondary synchronization word

**SW7 (0x61FB) – Secondary Sync Word Byte 3**

Bit No.	Name	Reset	R/W	Description
7:0	SYNC_WORD2[31:24]	0x00	R/W	Contains bits 31:24 of the secondary synchronization word

**FREQEST (0x6199) – Estimated RF Frequency Offset**

Bit No.	Name	Reset	R/W	Description
7:0	FREQEST[7:0]	0x00	R	Signed value. Contains an estimate of the frequency offset between carrier and the receiver frequency. FOC_MODE controls when this estimate is updated.

**RXCTRL (0x619A) – Receive Section Tuning**

Bit No.	Name	Reset	R/W	Description
7:6	–	00	R0	Reserved
5:4	GBIAS_LNA2_REF[1:0]	10	R/W	Adjusts front-end LNA2/mixer PTAT current output ( $M = \text{GBIAS\_LNA2\_REF}[1:0] + 3$ ), default: $M = 5$ .
3:2	GBIAS_LNA_REF[1:0]	10	R/W	Adjusts front-end LNA PTAT current output ( $M = \text{GBIAS\_LNA\_REF}[1:0] + 3$ ), default: $M = 5$ .
1:0	MIX_CURRENT[1:0]	01	R/W	Control of the output-current consumption of the receiver mixers. The current increases with increasing setting.

**FSCTRL (0x619B) – Frequency Synthesizer Tuning**

Bit No.	Name	Reset	R/W	Description
7:6	PRE_CURRENT [1:0]	01	R/W	Prescaler current setting
5:4	LODIV_BUF_CURRENT_TX [1:0]	01	R/W	Adjusts current in mixer and PA buffers (lodiv_buf_current). Used when lle_tx_active = 1
3:2	LODIV_BUF_CURRENT_RX [1:0]	01	R/W	Adjusts current in mixer and PA buffers (lodiv_buf_current). Used when lle_tx_active = 0
1:0	LODIV_CURRENT [1:0]	01	R/W	Adjusts divider currents, except mixer and PA buffers.

**LNAGAIN (0x61A0) – LNA Gain Setting**

Bit No.	Name	Reset	R/W	Description
7	–	0	R0	Reserved, read as 0
6:5	LNA1_CURRENT[1:0]	11	R/W	Gain setting, LNA1 00: 0-dB gain (reference level) 01: 3-dB gain 10: Reserved 11: 6-dB gain
4:2	LNA2_CURRENT[2:0]	111	R/W	Gain setting, LNA2 000: 0-dB gain (reference level) 001: 3-dB gain 010: 6-dB gain 011: 9-dB gain 100: 12-dB gain 101: 15-dB gain 110: 18-dB gain 111: 21-dB gain
1:0	LNA3_CURRENT[1:0]	11	R/W	Gain setting, LNA3 00: 0-dB gain (reference level) 01: 3-dB gain 10: 6-dB gain 11: 9-dB gain

**AAFGAIN (0x61A1) – AAF Gain Setting**

Bit No.	Name	Reset	R/W	Description
7:2	–	0000 00	R0	Reserved. Read as zero
1:0	AAF_GAIN[1:0]	11	R/W	Controls attenuation in AAF 00: 9-dB attenuation in AAF 01: 6-dB attenuation in AAF 10: 3-dB attenuation in AAF 11: 0-dB attenuation in AAF (reference level)

**ADCTEST0 (0x61A2) – ADC Tuning**

Bit No.	Name	Reset	R/W	Description
7:0	ADC_ADJ[7:0]	0x10	R/W	Adjust ADC gain

**MDMTEST0 (0x61A5) – Modem Configuration**

Bit No.	Name	Reset	R/W	Description
7:5	RSSI_ACC[2:0]	101	R/W	RSSI accuracy 000: 5.33- $\mu$ s average window 001: Mean of two 5.33- $\mu$ s average windows 010: Reserved 011: Mean of four 5.33- $\mu$ s average windows 100: 21.3- $\mu$ s average window 101: Mean of two 21.3- $\mu$ s average windows 110: Reserved 111: Mean of four 21.3- $\mu$ s average windows
4	–	0	R/W	Reserved, always write 0.
3:2	DC_BLOCK_LENGTH[1:0]	00	R/W	Controls the number of samples to be accumulated between each dump of the accumulate-and-dump filter used in dc removal. 00: 16 samples 01: 32 samples 10: 64 samples 11: 128 samples
1:0	DC_BLOCK_MODE[1:0]	01	R/W	Selects the mode of operation: 00 : Manual override mode 01 : Enable dc cancellation. Normal operation 10 : Freeze estimates of dc when sync is found. Start estimating dc again when searching for the next frame. 11 : Delayed dc offset estimate used. Delay set by MDMTEST1.DC_DELAY. Until the first estimate is ready, the manual override value is used.

**MDMTEST1 (0x61A6) – Modem Configuration**

Bit No.	Name	Reset	R/W	Description
7:6	DC_DELAY	00	R/W	Controls delay of dc estimate delayed dc block mode. Delay unit is set by MDMTEST0.DC_BLOCK_LENGTH 00: 5 delays 01: 6 delays 10: 7 delays 11: 8 delays
5	RX_IF	0	R/W	Controls mixer frequency in demodulator (not 2 Mbps) 0: 1 MHz 1: –1 MHz For 2 Mbps, always write 0. The receiver then operates at zero IF.
4:0	TX_TONE[4:0]	0 0000	R/W	Controls baseband frequency of transmission Note: If MDMCTRL0.PHASE_INVERT is 1, the sign of the frequency is inverted 0: –8 MHz 1: –6 MHz 2: –4 MHz 3: –3 MHz 4: –2 MHz 5: –1 MHz 6: –500 kHz 7: –250 kHz 8: –125 kHz 9: –4 kHz 10: 0 Hz 11: 4 kHz 12: 125 kHz 13: 250 kHz 14: 500 kHz 15: 1 MHz 16: 2 MHz 17: 3 MHz 18: 4 MHz 19: 6 MHz 20: 8 MHz

**AATEST (0x61A9) – Analog Test Control**

Bit No.	Name	Reset	R/W	Description
7:6	–	00	R0	Reserved. Read as zero
5:0	AATEST_CTRL[5:0]	00 0000	R/W	Controls the analog test mode: 00 0000: Disabled 00 0001: Enables the temperature sensor (see also the TR0 register description in <a href="#">#IMPLIED</a> ). Other values reserved.

**RFC\_OBS\_CTRL0 (0x61AE) – RF Observation Mux Control 0**

Bit No.	Name	Reset	R/W	Description
7	–	0	R0	Reserved. Read as 0
6	RFC_OBS_POL0	0	R/W	The signal chosen by RFC_OBS_MUX0 is XORed with this bit
5:0	RFC_OBS_MUX0	00 0000	R/W	<p>Controls which observable signal from rf_core is to be muxed out to rfc_obs_sigs(0); see <a href="#">Section 7.9</a>.</p> <p>00 0111: rfc_sniff_data – Data from packet sniffer, see <a href="#">Section 25.11</a></p> <p>00 1000: rfc_sniff_clk – Clock for packet sniffer data, see <a href="#">Section 25.11</a></p> <p>00 1001: tx_active</p> <p>00 1010: rx_active</p> <p>00 1011: vco_on – VCO on</p> <p>Low: The VCO is powered down, so the next RX or TX operation must start and calibrate the synthesizer before transmitting or receiving.</p> <p>High: The VCO is powered up. If the LLE is idle, it means the next task starts quickly if frequency programming is disabled (PRF_CHAN.FREQ = 127).</p> <p>00 1100: sync_search – RX search for sync</p> <p>Low: The modem is not ready to receive a packet.</p> <p>High: The modem is in search for a sync word or receiving a packet.</p> <p>00 1101: lle_idle – Link-layer engine idle</p> <p>Low: The LLE is busy processing or finishing a command, or in reset.</p> <p>High: The LLE is idle waiting for a command to start a new task.</p> <p>00 1110: wait_t2e1 – Indication on the LLE waiting for Timer 2 event 1 to start a task</p> <p>Low: Not waiting for Timer 2 event 1</p> <p>High: Command processed, event 1 not yet received</p> <p>00 1111: agc_lowgain – High if the AGC algorithm has reduced the front-end gain; low otherwise</p> <p>01 0011: fsc_lock – High when PLL is in lock; low otherwise</p> <p>01 1011: pa_pd - Power amplifier power-down signal</p> <p>10 1100: Inamix_pd - Low-noise amplifier power-down signal</p> <p>11 0000: dem_sync_found - High when demodulator has detected a sync word. Stays high until end of packet.</p> <p>11 0001: mod_sync_sent - High when modulator has sent a sync word. Stays high until end of packet.</p> <p>Others: Reserved</p>

**RFC\_OBS\_CTRL1 (0x61AF) – RF Observation Mux Control 1**

Bit No.	Name	Reset	R/W	Description
7	–	0	R0	Reserved. Read as 0
6	RFC_OBS_POL1	0	R/W	The signal chosen by RFC_OBS_MUX1 is XORed with this bit.
5:0	RFC_OBS_MUX1	00 0000	R/W	Controls which observable signal from rf_core is to be muxed out to rfc_obs_sigs(1). See description of RFC_OBS_CTRL0.

**RFC\_OBS\_CTRL2 (0x61B0) – RF Observation Mux Control 2**

Bit No.	Name	Reset	R/W	Description
7	–	0	R0	Reserved. Read as 0
6	RFC_OBS_POL2	0	R/W	The signal chosen by RFC_OBS_MUX2 is XORed with this bit.
5:0	RFC_OBS_MUX2	00 0000	R/W	Controls which observable signal from rf_core is to be muxed out to rfc_obs_sigs(2). See description of RFC_OBS_CTRL0.

**LLECTRL (0x61B1) – LLE Control**

Bit No.	Name	Reset	R/W	Description
7:3	–	0000 0	R0	Reserved. Read as 0
2:1	LLE_MODE_SEL	00	R/W	LLE mode. Changing this field has no effect unless LLE_EN is changed from 0 to 1. 00: Proprietary mode (described in this chapter) 01: BLE mode (only for use by the BLE stack) Others: Reserved
0	LLE_EN	0	R/W	Must be set to 0 before entering PM2 or PM3, otherwise the behavior of the RF core after waking up may be unpredictable. 0: LLE held in reset 1: LLE enabled

**TXFILTCFG (0x61BC) – TX Filter Configuration**

Bit No.	Name	Reset	R/W	Description
7:4	–	0000	R0	Reserved
3:2	–	11	R/W	Reserved
1:0	FC	11	R/W	Sets TX anti-aliasing filter to appropriate bandwidth. Reduces spurious emissions close to signal. For the best value to use, see <a href="#">Table 25-24</a> and <a href="#">Table 25-25</a> .

**RFRND (0x61BF) – Random Data**

Bit No.	Name	Reset	R/W	Description
7:0	RND	0x00	R	Random bits, provided analog part is in random number generation mode (receiver running without sync)

**RFRAMCFG (0x61C0) – Radio RAM Configuration**

Bit No.	Name	Reset	R/W	Description
7:3	–	0000 1	R	Reserved
2:0	PRE	000	R/W	Selects active memory page for RF core data memory

**RFFDMA0, (0x61C3) – Radio DMA Trigger 0 Control**

Bit No.	Name	Reset	R/W	Description
7:5	–	000	R	Reserved
4:0	DMA0	0 0000	R/W	Generate a pulse on radio DMA trigger 0 (DMA trigger 19) when: 0 0000: Never 0 0001: A byte is read from RX FIFO and more bytes remain or when a byte arrives in RX FIFO and it was previously empty. 0 0010: A byte is written to RX FIFO and there is available space left or when there becomes available space when the RX FIFO was full. 0 0011: RX FIFO is empty. 0 0100: RX FIFO is full. 0 0101: RX FIFO length equals RFRXFTHRS after a write to RX FIFO. 0 0110: RX FIFO is read when its size equals RFRXFTHRS . 0 0111: RX FIFO is reset (see <a href="#">Table 25-2</a> ). 0 1000: RX FIFO is deallocated (see <a href="#">Table 25-2</a> ). 0 1001: RX FIFO is retried (see <a href="#">Table 25-2</a> ). 0 1010: RX FIFO is discarded (see <a href="#">Table 25-2</a> ). 0 1011: RX FIFO is committed (see <a href="#">Table 25-2</a> ). 0 1100–0 1111: Reserved (never) 1 0000: Never 1 0001: A byte is read from TX FIFO and more bytes remain or when a byte arrives in TX FIFO and it was previously empty. 1 0010: A byte is written to TX FIFO and there is available space left or when there becomes available space when the TX FIFO was full. 1 0011: TX FIFO is empty. 1 0100: TX FIFO is full. 1 0101: TX FIFO length equals RFTXFTHRS after a write to TX FIFO. 1 0110: TX FIFO is read when its size equals RFTXFTHRS . 1 0111: TX FIFO is reset (see <a href="#">Table 25-2</a> ). 1 1000: TX FIFO is deallocated (see <a href="#">Table 25-2</a> ). 1 1001: TX FIFO is retried (see <a href="#">Table 25-2</a> ). 1 1010: TX FIFO is discarded (see <a href="#">Table 25-2</a> ). 1 1011: TX FIFO is committed (see <a href="#">Table 25-2</a> ). 1 1100–1 1111: Reserved (never)

**RFFDMA1, (0x61C4) – Radio DMA Trigger 1 Control**

Bit No.	Name	Reset	R/W	Description
7:5	–	000	R	Reserved
4:0	DMA1	0 0000	R/W	Condition for generating a pulse on radio DMA trigger 1 (DMA trigger 11). See RFFDMA0 for the list of conditions.



**RFFSTATUS (0x61C5) – FIFO Status**

Bit No.	Name	Reset	R/W	Description
7	TXAVAIL	0	R	0: No readable data in TX FIFO 1: Readable data present in TX FIFO
6	TXFEMPTY	1	R	0: Data present in TX FIFO 1: TX FIFO is empty
5	TXDTHRESH	1	R	0: There is less data in TX FIFO than the threshold amount given by RFTXFTHRS. 1: There is more than or equal amount of data in TX FIFO than the threshold amount given by RFTXFTHRS
4	TXFFULL	0	R	0: TX FIFO has available space 1: TX FIFO is full
3	RXAVAIL	0	R	0: No readable data in RX FIFO 1: Readable data present in RX FIFO
2	RXFEMPTY	1	R	0: Data present in RX FIFO 1: RX FIFO is empty
1	RXDTHRESH	1	R	0: There is less data in RX FIFO than the threshold amount given by RFRXFTHRS. 1: There is more than or equal amount of data in RX FIFO than the threshold amount given by RFRXFTHRS
0	RXFFULL	0	R	0: RX FIFO has available space 1: RX FIFO is full

**RFFCFG (0x61C6) – FIFO Configuration**

Bit No.	Name	Reset	R/W	Description
7:6	–	00	R	Reserved
5	TXAUTOCOMMIT	1	R/W	0: Commit TX FIFO only on command 0x95 1: Always set RFTXSWP = RFTXWP
4	TXFAUTODEALLOC	0	R/W	0: Deallocate TX FIFO only on command 0x92 1: Always set RFTXFSRP = RFTXFRP.
3:2	–	00	R	Reserved
1	RXAUTOCOMMIT	0	R/W	0: Commit RX FIFO only on command 0x85 1: Always set RFRXSWP = RFRXWP
0	RXFAUTODEALLOC	1	R/W	0: Deallocate RX FIFO only on command 0x82 1: Always set RFRXFSRP = RFRXFRP.

**RFRXFLEN (0x61C8) – RX FIFO Length**

Bit No.	Name	Reset	R/W	Description
7:0	D	0x00	R	Amount of data present in RX FIFO

**RFRXFTHRS (0x61C9) – RX FIFO Threshold**

Bit No.	Name	Reset	R/W	Description
7	–	0	R	Reserved
6:0	D	000 0000	R/W	Threshold value for RX FIFO

**RFRXFWR (0x61CA) – RX FIFO Write Register**

Bit No.	Name	Reset	R/W	Description
7:0	D	0x00	W	Data written to this register is written to the RX FIFO address at offset RFRXFWR from the start of the RX FIFO area (see <a href="#">Figure 25-1</a> ). RFRXFWR (and RFRXFSWP if RFFCFG.RXAUTODEALLOC = 1) is incremented by 1 modulo 0x80 unless the write fails.

**RFRXFRD (0x61CB) – RX FIFO Read Register**

Bit No.	Name	Reset	R/W	Description
7:0	D	0x00	R	When this register is read, the data in RX FIFO address offset RFRXFRP from the start of the RX FIFO area is returned (see <a href="#">Figure 25-1</a> ). RFRXFRP (and RFRXFSRP if RFFCFG.RXAUTO COMMIT = 1) is incremented by 1 modulo 0x80 unless the read fails.

**RFRXFWP (0x61CC) – RX FIFO Write Pointer**

Bit No.	Name	Reset	R/W	Description
7	–	0	R	Reserved
6:0	D	000 0000	R/W	RX FIFO write pointer. This is the offset into the RX FIFO to which the next write operation writes.

**RFRXFRP (0x61CD) – RX FIFO Read Pointer**

Bit No.	Name	Reset	R/W	Description
7	–	0	R	Reserved
6:0	D	000 0000	R/W	RX FIFO read pointer. This is the offset into the RX FIFO from which the next read operation reads.

**RFRXFSWP (0x61CE) – RX FIFO Start-of-Frame Write Pointer**

Bit No.	Name	Reset	R/W	Description
7	–	0	R	Reserved
6:0	D	000 0000	R/W	RX FIFO start of written package. This is the point to which the write pointer can be reset if a discard command is issued.

**RFRXFSRP (0x61CF) – RX FIFO Start-of-Frame Read Pointer**

Bit No.	Name	Reset	R/W	Description
7	–	0	R	Reserved
6:0	D	000 0000	R/W	RX FIFO start of read package. This is the start of the allocated part of the RX FIFO.

**RFTXFLEN (0x61D0) – TX FIFO Length**

Bit No.	Name	Reset	R/W	Description
7:0	D	0x00	R	Amount of data present in TX FIFO

**RFTXFTHRS (0x61D1) – TX FIFO Threshold**

Bit No.	Name	Reset	R/W	Description
7	–	0	R	Reserved
6:0	D	000 0000	R/W	Threshold value for TX FIFO

**RFTXFWR (0x61D2) – TX FIFO Write Register**

Bit No.	Name	Reset	R/W	Description
7:0	D	0x00	W	Data written to this register is written to the TX FIFO address at offset RFTXFWR from the start of the TX FIFO area (see <a href="#">Figure 25-1</a> ) is returned. RFTXFWR (and RFTXFSPW if RFFCFG.TXAUTO DEALLOC = 1) is incremented by 1 modulo 0x80 unless the write fails.

**RFTXFRD (0x61D3) – TX FIFO Read Register**

Bit No.	Name	Reset	R/W	Description
7:0	D	0x00	R	When this register is read, the data in TX FIFO address offset RFTXFRD from the start of the TX FIFO area is returned (see <a href="#">Figure 25-1</a> ). RFTXFRD (and RFTXFSRP if RFFCFG.TXAUTO COMMIT = 1) is incremented by 1 modulo 0x80 unless the read fails.

**RFTXFWP (0x61D4) – TX FIFO Write Pointer**

Bit No.	Name	Reset	R/W	Description
7	–	0	R	Reserved
6:0	D	000 0000	R/W	TX FIFO write pointer. This is the offset into TX FIFO the next write operation writes to.

**RFTXFRP (0x61D5) – TX FIFO Read Pointer**

Bit No.	Name	Reset	R/W	Description
7	–	0	R	Reserved
6:0	D	000 0000	R/W	TX FIFO read pointer. This is the offset into TX FIFO the next read operation reads from.

**RFTXFSWP (0x61D6) – TX FIFO Start-of-Frame Write Pointer**

Bit No.	Name	Reset	R/W	Description
7	–	0	R	Reserved
6:0	D	000 0000	R/W	TX FIFO start of written package. This is the point to which the write pointer can be reset if a discard command is issued.

**RFTXFSRP (0x61D7) – TX FIFO Start-of-Frame Read Pointer**

Bit No.	Name	Reset	R/W	Description
7	–	0	R	Reserved
6:0	D	0x00	R/W	TX FIFO start-of-read package. This is the start of the allocated part of the TX FIFO.

**BSP\_P0 (0x61E0) – CRC Polynomial Byte 0**

Bit No.	Name	Reset	R/W	Description
7:0	P[7:0]	0x00	R/W	Bits 7:0 of p register in CRC sub-module

**BSP\_P1 (0x61E1) – CRC Polynomial Byte 1**

Bit No.	Name	Reset	R/W	Description
7:0	P[15:8]	0x5B	R/W	Bits 15:8 of p register in CRC sub-module

**BSP\_P2 (0x61E2) – CRC Polynomial Byte 2**

Bit No.	Name	Reset	R/W	Description
7:0	P[23:16]	0x06	R/W	Bits 23:16 of p register in CRC sub-module

**BSP\_P3 (0x61E3) – CRC Polynomial Byte 3**

Bit No.	Name	Reset	R/W	Description
7:0	P[31:24]	0x00	R/W	Bits 31:24 of p register in CRC sub-module

**BSP\_D0 (0x61E4) – CRC Value Byte 0**

Bit No.	Name	Reset	R/W	Description
7:0	D[7:0]	0x00	R/W	Bits 7:0 of d register in CRC sub-module

**BSP\_D1 (0x61E5) – CRC Value Byte 1**

Bit No.	Name	Reset	R/W	Description
7:0	D[15:8]	0x5B	R/W	Bits 15:8 of d register in CRC sub-module

**BSP\_D2 (0x61E6) – CRC Value Byte 2**

Bit No.	Name	Reset	R/W	Description
7:0	D[23:16]	0x06	R/W	Bits 23:16 of d register in CRC sub-module

**BSP\_D3 (0x61E7) – CRC Value Byte 3**

Bit No.	Name	Reset	R/W	Description
7:0	D[31:24]	0x00	R/W	Bits 31:24 of d register in CRC sub-module

**BSP\_W (0x61E8) – Whitener Value**

Bit No.	Name	Reset	R/W	Description
7	W_PN9_RESET	0	R0	When a 1 is written to this bit, the CC2500-compatible whitener is reset, and all bits in the s and b registers are set to 1.
6:0	W[6:0]	110 0101	R/W	Write: Writes all whitening registers. $w_6$ is set to BSP_W[0], $w_5$ is set to BSP_W[1] and so on up to $w_1$ is set to BSP_W[5]. $w_0$ is set to 1. Read: Reads back w register. BSP_W[0] is set to $w_6$ , BSP_W[1] is set to $w_5$ and so on up to BSP_W[6] is set to $w_0$ .

**BSP\_MODE (0x61E9) – Bit Stream Processor Configuration**

Bit No.	Name	Reset	R/W	Description
7	–	0	R0	Reserved. Read as zero
6	CP_BUSY	0	R	Coprocessor mode busy. Goes to 1 after a byte has been written to BSP_DATA. Goes to 0 when a byte is ready to be read back from BSP_DATA
5	CP_READOUT	0	R/W	Coprocessor mode readout
4	CP_END	0	R/W	Endianness of data in coprocessor mode. 0: LSB processed first 1: MSB processed first
3:2	CP_MODE[1:0]	00	R/W	Coprocessor mode 00: Coprocessor disabled 01: Coprocessor receive mode 10: Reserved 11: Coprocessor transmit mode
1	W_PN9_EN	0	R/W	Enable CC2500-compatible PN9 whitener
0	W_PN7_EN	1	R/W	Enable PN7 whitener

**BSP\_DATA (0x61EA) – Bit Stream Processor Coprocessor Data**

Bit No.	Name	Reset	R/W	Description
7:0	BSP_DATA[7:0]	0x00	R/W	When BSP_MODE . CP_BUSY = 0: Write: Provide byte to be processed in coprocessor mode Read: Read processed byte

**DC\_I\_L (0x61FC) – In-Phase DC Offset Estimate, Low Byte**

Bit No.	Name	Reset	R/W	Description
7:0	DC_I[7:0]	0x00	R*/W	When running dc estimation, this register reflects the 8 LSBs of the dc estimate in the I channel. When manual dc override is selected, the override value is written to this register.

**DC\_I\_H (0x61FD) – In-Phase DC Offset Estimate, High Byte**

Bit No.	Name	Reset	R/W	Description
7:0	DC_I[15:8]	0x00	R*/W	When running dc estimation, this register reflects the 8 MSBs of the dc estimate in the I channel. When manual dc override is selected, the override value is written to this register.

**DC\_Q\_L (0x61FE) – Quadrature-Phase DC Offset Estimate Low Byte**

Bit No.	Name	Reset	R/W	Description
7:0	DC_Q[7:0]	0x00	R*/W	When running dc estimation, this register reflects the 8 LSBs of the dc estimate in the Q channel. When manual dc override is selected, the override value is written to this register.

**DC\_Q\_H (0x61FF) – Quadrature-Phase DC Offset Estimate High Byte**

Bit No.	Name	Reset	R/W	Description
7:0	DC_Q[15:8]	0x00	R*/W	When running dc estimation, this register reflects the 8 MSBs of the dc estimate in the Q channel. When manual dc override is selected, the override value is written to this register.

**IVCTRL (0x6265) – Analog Control Register**

Bit No.	Name	Reset	R/W	Description
7:6	–	00	R0	Reserved
5:4	TX_MIX_LOAD	01	R/W	Controls load capacitor in TX mixer 00: Minimum load ... (Intermediate loads) 11: Maximum load
3	LODIV_BIAS_CTRL	0	R/W	Controls bias current to LODIV 0: IVREF bias 1: PTAT bias
2	–	0	R/W	Reserved
1:0	PA_BIAS_CTRL	11	R/W	Controls bias current to PA 00: IREF bias 01: IREF and IVREF bias 10: PTAT bias 11: Increased PTAT slope bias

## Voltage Regulator

---

---

---

The digital voltage regulator is used to power the digital core. The output of this regulator is available on the `DCOUPLE` pin and requires capacitive decoupling to function properly (see, for example, the CC2530 reference design).

The voltage regulator is disabled in power modes PM2 and PM3 (see [Chapter 4](#)). When the voltage regulator is disabled, register and RAM contents are retained while the unregulated 2 V to 3.6 V power supply is present

---

**NOTE:** The voltage regulator should not be used to provide power to external circuits.

---

## Available Software

---



---

This chapter presents the various available software solutions relevant to the CC253x, CC2540, and CC2541 family. They are all available free of charge on the TI Web site at [www.ti.com/lprf](http://www.ti.com/lprf) when used with TI LPRF devices.

As shown in [Table 0-1](#) in the Preface, the members of the CC253x, CC2540, and CC2541 family have different flash and RAM sizes; hence, they are not equally well suited for the different software offerings in the sections below. For example, a user designing a ZigBee device should use the CC2530F256, as the Z-Stack™ requires in most cases more than 128 KB of flash and needs the 8-KB RAM.

Topic	Page
<b>27.1 SmartRF™ Software for Evaluation (<a href="http://www.ti.com/smartrfstudio">www.ti.com/smartrfstudio</a>)</b> .....	<b>344</b>
<b>27.2 RemoTI™ Network Protocol (<a href="http://www.ti.com/remoti">www.ti.com/remoti</a>)</b> .....	<b>344</b>
<b>27.3 SimpliciTI™ Network Protocol (<a href="http://www.ti.com/simpliciti">www.ti.com/simpliciti</a>)</b> .....	<b>345</b>
<b>27.4 TIMAC Software (<a href="http://www.ti.com/timac">www.ti.com/timac</a>)</b> .....	<b>345</b>
<b>27.5 Z-Stack™ Software (<a href="http://www.ti.com/z-stack">www.ti.com/z-stack</a>)</b> .....	<b>346</b>
<b>27.6 BLE Stack Software</b> .....	<b>346</b>

## 27.1 SmartRF™ Software for Evaluation ([www.ti.com/smarrfstudio](http://www.ti.com/smarrfstudio))

Texas Instruments' SmartRF Studio can be used for radio performance and functionality evaluation and is great for exploring and gaining knowledge about the RF-IC products. This software helps the designers of radio systems to evaluate the RF-ICs easily at an early stage in the design process. It is especially useful for generation of the configuration data and for finding optimized external component values.

SmartRF Studio software runs on Microsoft™ Windows™ 98, Windows 2000, Windows XP, Windows Vista (32 bit) and Windows 7 (32 bit). SmartRF Studio software can be downloaded from the Texas Instruments Web page: [www.ti.com/smarrfstudio](http://www.ti.com/smarrfstudio) (<http://www.ti.com/litv/zip/swrc046m>).

### Features

- Link tests. Send and receive packets between nodes
- Packet error-rate (PER) tests
- Communication with evaluation boards through the USB port or the parallel port
- Up to eight USB devices are supported on a single computer.
- Normal view with preferred register settings
- Register view with possibilities to read and write each individual register. Each register given with detailed information
- Save or Open configuration data from file
- Save or Load register settings from file
- Export or Import register values from text file
- Exports register settings into a C-compatible software structure

## 27.2 RemoTI™ Network Protocol ([www.ti.com/remoti](http://www.ti.com/remoti))

Most existing remote controls use infrared technology to communicate commands to consumer electronics devices. However, radio frequency (RF) remote controls enable non-line-of-sight operation and provide more advanced features based on bidirectional RF communication.

ZigBee Radio Frequency for Consumer Electronics (RF4CE) is the result of a recent agreement between the ZigBee Alliance and the RF4CE Consortium (<http://www.zigbee.org/rf4ce>) and has been designed to be deployed in a wide range of remotely-controlled audio and visual consumer electronics products, such as TVs and set-top boxes. ZigBee RF4CE promises you:

- Richer communication and increased reliability
- Enhanced features and flexibility
- Interoperability
- No line-of-sight barrier

The RemoTI network protocol is Texas Instruments' implementation of the ZigBee RF4CE standard. It is a complete solution offering hardware and software support for TI's low-power RF product portfolio. With the RemoTI network protocol we provide:

- The industry-leading RF4CE-compliant stack featuring the interoperable CERC profile support, a simple API, easy-to-understand sample application code, full development kits and reference designs, and much more.
- Operation on our best-in-class IEEE 802.15.4 compliant System-on-Chip, the CC2530, with excellent RF co-existence and RF performance. The four flexible power modes include the lowest-current-consumption power-down mode for long battery in life low-duty-cycle applications.
- Extensive worldwide support and tools to ensure that development of ZigBee RF4CE-based products is simple, fast, and can be completed at minimal cost.
- The RemoTI network protocol is a Golden Unit platform; that is, it is used for testing other implementations of the ZigBee RF4CE standard for standard compliance.

For more information on TI's RemoTI network protocol, see the Texas Instruments RemoTI network protocol Web site [www.ti.com/remoti](http://www.ti.com/remoti).



### 27.3 SimpliciTI™ Network Protocol ([www.ti.com/simpliciti](http://www.ti.com/simpliciti))

The SimpliciTI network protocol is a low-power RF protocol (for sub-1 GHz, 2.4 GHz, and IEEE 802.15.4 RF ICs) targeting simple, small RF networks. This open-source software is an excellent start for building a network with battery-operated devices using a TI low-power RF System-on-Chip (SoC). The SimpliciTI network protocol was designed for easy implementation and deployment out-of-the-box on several TI RF platforms. It provides several sample applications.

#### Key Applications

- Alarm and security: occupancy sensors, light sensors, carbon monoxide sensors, glass-breakage detectors
- Smoke detectors
- Automatic meter reading: gas meters, water meters, e-meters
- Active RFID applications

#### Key Features

- Low power: A TI-proprietary low-power network protocol
- Flexible:
  - Direct device-to-device communication
  - Simple star with access point for store and forward to end device
  - Range extenders to increase range to four hops
- Simple: uses a five-command API
- Low data rate and low duty cycle
- Ease of use

For more information about the SimpliciTI network protocol, see the Texas Instruments SimpliciTI network protocol Web site, [www.ti.com/simpliciti](http://www.ti.com/simpliciti).

### 27.4 TIMAC Software ([www.ti.com/timac](http://www.ti.com/timac))

TIMAC software is an IEEE 802.15.4 medium-access-control software stack for TI's IEEE 802.15.4 transceivers and System-on-Chips.

You can use TIMAC when you:

- Need a wireless point-to-point or point-to-multipoint solution; for example, multiple sensors reporting directly to a master
- Need a standardized wireless protocol
- Have battery-powered and/or mains-powered nodes
- Need support for acknowledgment and retransmission
- Have low data-rate requirements (around 100-kbps effective data rate)

#### Features

- Support for IEEE 802.15.4 standard
- Support for beacon-enabled and non-beaconing systems
- Multiple platforms
- Easy application development

The TIMAC software stack is certified to be compliant with the IEEE 802.15.4 standard. TIMAC software is distributed as object code free of charge. There are no royalties for using TIMAC software.

For more information about TIMAC software, see the Texas Instruments TIMAC Web site [www.ti.com/timac](http://www.ti.com/timac).

## 27.5 Z-Stack™ Software ([www.ti.com/z-stack](http://www.ti.com/z-stack))

The Z-Stack software is TI's ZigBee-compliant protocol stack for a growing portfolio of IEEE 802.15.4 products and platforms. The Z-Stack software stack is compliant with the ZigBee-2007 specification, supporting both the ZigBee and ZigBee PRO features sets. The Z-Stack software includes implementation of two ZigBee application profiles – SmartEnergy and HomeAutomation. Other application profiles can easily be implemented by the user.

Z-Stack software notables include:

- A fully compliant ZigBee and ZigBee PRO feature set
- A range of sample applications including support for the ZigBee Smart Energy and ZigBee Home Automation profiles
- Over-the-air download and serial boot loader support
- Can be used together with the RF front ends, [CC2590](#) and [CC2591](#), which support 10 dBm and 20 dBm output power, respectively, and improved receive sensitivity.

The Z-Stack software has been awarded the ZigBee Alliance's golden-unit status for both the ZigBee and ZigBee PRO stack profiles and is used by ZigBee developers worldwide.

Z-Stack software is well suited for:

- Smart energy (AMI)
- Home automation
- Commercial building automation
- Medical, assisted living, or personal health and hospital care
- Monitoring and control applications
- Wireless sensor networks
- Alarm and security
- Asset tracking
- Applications that require interoperability

For more information about Z-Stack software, see the Texas Instruments Z-Stack software Web site [www.ti.com/z-stack](http://www.ti.com/z-stack).

## 27.6 BLE Stack Software

TI's single-mode *Bluetooth* low energy stack has been certified according to the *Bluetooth* 4.0 low energy specification. Key features:

- Supports all BLE roles
- Range of example applications
- Multi-role capabilities

For more information about TI's BLE stack software, visit Texas Instruments *Bluetooth* low energy stack software Web site at [www.ti.com/blestack](http://www.ti.com/blestack).

## Abbreviations

Abbreviations used in this user's guide:

AAF	Anti-aliasing filter
ACK	Acknowledge
ADC	Analog-to-digital converter
AES	Advanced Encryption Standard
AGC	Automatic gain control
ARIB	Association of Radio Industries and Businesses
BCD	Binary-coded decimal
BER	Bit error rate
BLE	Bluetooth low-energy
BOD	Brownout detector
BOM	Bill of materials
BSP	Bit-stream process
CBC	Cipher block chaining
CBC-MAC	Cipher block chaining message authentication code
CCA	Clear channel assessment
CCM	Counter mode + CBC-MAC
CFB	Cipher feedback
CFR	Code of Federal Regulations
CMRR	Common-mode rejection ratio
CPU	Central processing unit
CRC	Cyclic redundancy check
CSMA-CA	Carrier sense multiple access with collision avoidance
CSP	Command strobe processor
CTR	Counter mode (encryption)
CW	Continuous wave
DAC	Digital-to-analog converter
DC	Direct current
DMA	Direct memory access
DSM	Delta-sigma modulator
DSSS	Direct-sequence spread spectrum
ECB	Electronic code book (encryption)
EM	Evaluation module
ENOB	Effective number of bits
ETSI	European Telecommunications Standards Institute
EVM	Error vector magnitude
FCC	Federal Communications Commission
FCF	Frame control field
FCS	Frame check sequence
FFCTRL	FIFO and frame control
FIFO	First in, first out

FS	Full scale
GPIO	General-purpose input/output
HF	High frequency
HSSD	High-speed serial data
I/O	Input/output
I/Q	In-phase/quadrature-phase
IEEE	Institute of Electrical and Electronics Engineers
IF	Intermediate frequency
IOC	I/O controller
IRQ	Interrupt request
IR	Infrared
ISM	Industrial, scientific and medical
ITU-T	International Telecommunication Union – Telecommunication
IV	Initialization vector
KB	1024 bytes
kbps	Kilobits per second
LFSR	Linear feedback shift register
LLE	Link-layer engine
LNA	Low-noise amplifier
LO	Local oscillator
LQI	Link quality indication
LSB	Least-significant bit/byte
MAC	Medium access control
MAC	Message authentication code
MCU	Microcontroller unit
MFR	MAC footer
MHR	MAC header
MIC	Message integrity code
MISO	Master in, slave out
MOSI	Master out, slave in
MPDU	MAC protocol data unit
MSB	Most-significant bit/byte
MSDU	MAC service data unit
MUX	Multiplexer
NA	Not applicable/available
NC	Not connected
OFB	Output feedback (encryption)
O-QPSK	Offset – quadrature phase-shift keying
PA	Power amplifier
PC	Program counter
PCB	Printed circuit board
PER	Packet error rate
PHR	PHY header
PHY	Physical layer
PLL	Phase-locked loop
PM1, PM2, PM3	Power mode 1, 2, and 3
PMC	Power management controller
PN7, PN9	7-bit or 9-bit pseudo-random sequence
POR	Power-on reset

PSDU	PHY service data unit
PWM	Pulse-width modulator
RAM	Random access memory
RBW	Resolution bandwidth
RC	Resistor-capacitor
RCOSC	RC oscillator
RF	Radio frequency
RSSI	Receive signal strength indicator
RTC	Real-time clock
RX	Receive
SCK	Serial clock
SFD	Start of frame delimiter
SFR	Special function register
SHR	Synchronization header
SINAD	Signal-to-noise and distortion ratio
SPI	Serial peripheral interface
SRAM	Static random-access memory
ST	Sleep timer
T/R	Tape and reel
T/R	Transmit/receive
THD	Total harmonic distortion
TI	Texas Instruments
TX	Transmit
UART	Universal asynchronous receiver/transmitter
USART	Universal synchronous/asynchronous receiver/transmitter
VCO	Voltage-controlled oscillator
VGA	Variable-gain amplifier
WDT	Watchdog timer
XOSC	Crystal oscillator

## ***Additional Information***

---



---

Texas Instruments offers a wide selection of cost-effective, low-power RF solutions for proprietary and standard-based wireless applications for use in industrial and consumer applications. Our selection includes RF transceivers, RF transmitters, RF front ends and System-on-Chips as well as various software solutions for the sub-1 and 2.4-GHz frequency bands.

In addition, Texas Instruments provides a large selection of support collateral such as development tools, technical documentation, reference designs, application expertise, customer support, third-party and university programs.

The Low-Power RF E2E Online Community provides you with technical support forums, videos and blogs, and the chance to interact with fellow engineers from all over the world.

With a broad selection of product solutions, end application possibilities, and the range of technical support, Texas Instruments offers the broadest low-power RF portfolio. **We make RF easy!**

The following subsections point to where to find more information.

<b>Topic</b>	<b>Page</b>
<b>B.1 Texas Instruments Low-Power RF Web Site .....</b>	<b>351</b>
<b>B.2 Low-Power RF Online Community .....</b>	<b>351</b>
<b>B.3 Texas Instruments Low-Power RF Developer Network .....</b>	<b>351</b>
<b>B.4 Low-Power RF eNewsletter .....</b>	<b>351</b>

### **B.1 Texas Instruments Low-Power RF Web Site**

Texas Instruments' Low-Power RF Web site has all our latest products, application and design notes, FAQ section, news and events updates, and much more. Just go to [www.ti.com/lprf](http://www.ti.com/lprf).

### **B.2 Low-Power RF Online Community**

- Forums, videos, and blogs
- RF design help
- E2E interaction - Posting one's own and reading other users' questions

Join us today at [www.ti.com/lprf-forum](http://www.ti.com/lprf-forum)

### **B.3 Texas Instruments Low-Power RF Developer Network**

Texas Instruments has launched an extensive network of low-power RF development partners to help customers speed up their application development. The network consists of recommended companies, RF consultants, and independent design houses that provide a series of hardware module products and design services, including:

- RF circuit, low-power RF and ZigBee design services
- Low-power RF and ZigBee module solutions and development tools
- RF certification services and RF circuit manufacturing

Need help with modules, engineering services or development tools?

Search the [Low-Power RF Developer Network](#) to find a suitable partner! [www.ti.com/lprfnetwork](http://www.ti.com/lprfnetwork)

### **B.4 Low-Power RF eNewsletter**

The Low-Power RF eNewsletter keeps you up to date on new products, news releases, developers' news, and other news and events associated with low-power RF products from TI. The Low-Power RF eNewsletter articles include links to get more online information.

Sign up today on [www.ti.com/lprfnewsletter](http://www.ti.com/lprfnewsletter).

---

---

## References

---

---

References and other useful material:

1. IEEE Std. 802.15.4-2006: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Low Rate Wireless Personal Area Networks (LR-WPANs)  
<http://standards.ieee.org/getieee802/download/802.15.4-2006.pdf>
2. CC2530 Data Sheet ([SWRS081](#))
3. CC2531 Data Sheet ([SWRS086](#))
4. CC2533 Data Sheet ([SWRS087](#))
5. CC2540 Data Sheet ([SWRS084](#))
6. CC2541 Data Sheet ([SWRS110](#))
7. Bluetooth® Core Technical Specification document, version 4.0  
<https://www.bluetooth.org/technical/specifications/adopted.htm>
8. Universal Serial Bus Revision 2.0 specification [http://www.usb.org/developers/docs/usb\\_20\\_101111.zip](http://www.usb.org/developers/docs/usb_20_101111.zip)



## Revision History

### Changes from E Revision (April 2014) to F Revision

**Page**

- 
- Added framed note in [Section 3.4.1](#) ..... [57](#)
- 

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

## IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as "components") are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI's terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers' products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers' products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI's goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or "enhanced plastic" are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have **not** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

### Products

Audio	<a href="http://www.ti.com/audio">www.ti.com/audio</a>
Amplifiers	<a href="http://amplifier.ti.com">amplifier.ti.com</a>
Data Converters	<a href="http://dataconverter.ti.com">dataconverter.ti.com</a>
DLP® Products	<a href="http://www.dlp.com">www.dlp.com</a>
DSP	<a href="http://dsp.ti.com">dsp.ti.com</a>
Clocks and Timers	<a href="http://www.ti.com/clocks">www.ti.com/clocks</a>
Interface	<a href="http://interface.ti.com">interface.ti.com</a>
Logic	<a href="http://logic.ti.com">logic.ti.com</a>
Power Mgmt	<a href="http://power.ti.com">power.ti.com</a>
Microcontrollers	<a href="http://microcontroller.ti.com">microcontroller.ti.com</a>
RFID	<a href="http://www.ti-rfid.com">www.ti-rfid.com</a>
OMAP Applications Processors	<a href="http://www.ti.com/omap">www.ti.com/omap</a>
Wireless Connectivity	<a href="http://www.ti.com/wirelessconnectivity">www.ti.com/wirelessconnectivity</a>

### Applications

Automotive and Transportation	<a href="http://www.ti.com/automotive">www.ti.com/automotive</a>
Communications and Telecom	<a href="http://www.ti.com/communications">www.ti.com/communications</a>
Computers and Peripherals	<a href="http://www.ti.com/computers">www.ti.com/computers</a>
Consumer Electronics	<a href="http://www.ti.com/consumer-apps">www.ti.com/consumer-apps</a>
Energy and Lighting	<a href="http://www.ti.com/energy">www.ti.com/energy</a>
Industrial	<a href="http://www.ti.com/industrial">www.ti.com/industrial</a>
Medical	<a href="http://www.ti.com/medical">www.ti.com/medical</a>
Security	<a href="http://www.ti.com/security">www.ti.com/security</a>
Space, Avionics and Defense	<a href="http://www.ti.com/space-avionics-defense">www.ti.com/space-avionics-defense</a>
Video and Imaging	<a href="http://www.ti.com/video">www.ti.com/video</a>

### TI E2E Community

[e2e.ti.com](http://e2e.ti.com)