

Z20K14xMC Series Reference Manual

Supports: Z20K14xMC

Document Number: Z20K14xMC-RM
Rev. 1.3, November, 2024



Contents

Contents.....	2
Chapter 1 About This Manual	21
1.1 Introduction.....	21
1.2 Audience	21
1.3 Module descriptions	21
1.4 Register descriptions	21
1.5 Conventions	21
1.5.1 Numbering systems.....	21
1.5.2 Acronyms and abbreviations	22
1.5.3 Definition of terms.....	22
Chapter 2 Introduction	23
2.1 Overview	23
2.2 Z20K14xMC series introduction	23
2.3 Feature summary	23
2.4 Block diagram.....	26
2.5 Feature comparison	27
2.6 Applications.....	29
2.7 Module functional categories.....	30
2.7.1 ARM Cortex-M4F core modules	31
2.7.2 System modules	31
2.7.3 Memories and memory interfaces	32
2.7.4 Power management.....	32
2.7.5 Clocking	33
2.7.6 Analog modules	33
2.7.7 Timer modules	33
2.7.8 Communication interfaces	35
Chapter 3 Memory Map	36
3.1 Introduction.....	36
3.2 SRAM and Flash memory map	36
3.3 Peripheral bridge memory map	36
3.3.1 Read-after-write sequence and required serialization of memory options.....	39
3.4 Private peripheral bus (PPB) memory map	39
Chapter 4 Signal Multiplexing and Pin Assignment	41
4.1 Introduction.....	41
4.2 Functional description	41
4.3 Pad description.....	41
4.4 Signal multiplexing sheet.....	42
Chapter 5 Cortex-M4F Overview	43
5.1 ARM Cortex-M4F core introductions	43
5.1.1 Buses, interconnects, and interfaces	43
5.1.2 System tick timer	43
5.1.3 Debug facilities.....	43
5.1.4 Caches.....	43
5.1.5 Core privilege levels	43
5.2 Nested vectored interrupt controller (NVIC).....	43
5.2.1 Interrupt priority levels	43
5.2.2 Non-maskable interrupt	44
5.2.3 Interrupt vector.....	44
5.3 WIC introduction.....	49
5.3.1 Wake-up sources.....	49
Chapter 6 System Control Module (SCM).....	51

6.1	Introduction.....	51
6.1.1	Features.....	51
6.2	Signal description	51
6.3	Memory map and register definition	51
6.3.1	Memory map.....	51
6.3.2	Chip Control Register (SCM_CHIPCTL)	52
6.3.3	Misc Control Register 1 (SCM_MISCCTL1).....	52
6.3.4	Misc Control Register 2 (SCM_MISCCTL2).....	54
6.3.5	Misc Status Register 1 (SCM_MISCSTAT1).....	55
6.3.6	Misc Data Register 1 (SCM_MISCDATA1)	56
6.3.7	Misc Data register 2 (SCM_MISCDATA2).....	57
6.3.8	Chip Option Register (SCM_CHIPOPT).....	57
6.3.9	Flash Configuration Register (SCM_FLSCFG)	58
6.3.10	RAM Configuration Register (SCM_RAMCFG).....	58
6.3.11	System Device Identification Register (SCM_DEVID)	59
6.3.12	Unique Identification Register n (SCM_UNIQUE _n)	60
Chapter 7	Port Control and Pin Interrupts (PORT)	61
7.1	Introduction.....	61
7.1.1	Features.....	61
7.2	Signal description	61
7.3	Memory map and register definition	62
7.3.1	Memory map.....	62
7.3.2	Global Pin Data Register (PORTx_GLB _{DATA})	67
7.3.3	Global Pin Control Register (PORTx_GLB _{CTL}).....	68
7.3.4	PORT Pin Control Register n (PORTx_PCR _n)	68
7.3.5	PORT Interrupt Request Flag Register(PORTx_IRQ _{FLG})	70
7.4	Function description	70
7.4.1	Pin control.....	70
7.4.2	Global pin control.....	71
7.4.3	Global interrupt status.....	71
7.4.4	Pin interrupts.....	71
7.4.5	Digital filter	72
Chapter 8	General-Purpose Input/Output (GPIO).....	73
8.1	Introduction.....	73
8.1.1	Features.....	73
8.2	Signal description	73
8.3	Memory map and register definition	74
8.3.1	Memory map.....	74
8.3.2	Port Data Output Register (GPIOx_PDOR).....	75
8.3.3	Port Set Output Register (GPIOx_PSOR).....	76
8.3.4	Port Clear Output Register (GPIOx_PCOR).....	76
8.3.5	Port Toggle Output Register (GPIOx_PTOR)	77
8.3.6	Port Data Input Register (GPIOx_PDIR)	77
8.3.7	Port Data Direction Register (GPIOx_PDDR)	78
8.4	Function description	78
8.4.1	General-purpose input.....	78
8.4.2	General-purpose output.....	78
Chapter 9	Direct Memory Access Controller (DMA)	80
9.1	Introduction.....	80
9.1.1	Features.....	80
9.1.2	Modes of operation	80
9.1.3	Block diagram	80
9.2	Memory map and register definition	81
9.2.1	Memory map.....	81
9.2.2	DMA Configure Register (DMA_CONF)	82
9.2.3	DMA Request Enable Register (DMA_DMAE).....	83

9.2.4	DMA Error Interrupt Enable Register (DMA_EIE)	85
9.2.5	DMA Global Channel Control Register (DMA_GCC)	86
9.2.6	DMA Error Status Register (DMA_ES)	88
9.2.7	DMA Channel Error Status Register (DMA_CES)	89
9.2.8	DMA Channel Interrupt Status Register (DMA_CIS)	91
9.2.9	DMA Channel Hardware Request Status Register (DMA_CHRS)	92
9.2.10	DMA Channel Priority Register 0 (DMA_CPRI0)	93
9.2.11	DMA Channel Priority Register 1 (DMA_CPRI1)	95
9.2.12	DMA Channel Priority Register 2 (DMA_CPRI2)	97
9.2.13	DMA Channel Priority Register 3 (DMA_CPRI3)	99
9.2.14	DMA Channel n Source Address Register (DMA_SADDRn)	101
9.2.15	DMA Channel n Destination Address Register (DMA_DADDRn)	102
9.2.16	DMA Channel n Address Offset Register (DMA_AOFFn)	102
9.2.17	DMA Channel n Major Loop Address Offset Register (DMA_MLAOFFn)	102
9.2.18	DMA Channel n Number of Transfer Data (DMA_NUMn)	103
9.2.19	DMA Channel n Iteration Number Register (DMA_ITERn)	103
9.2.20	DMA Channel n Control And Status Register (DMA_CSn)	104
9.3	Function description	105
9.3.1	DMA transactions	105
9.3.2	Major loop and minor loop	106
9.3.3	DMA channel priority arbitration	108
9.3.4	DMA channel program	109
9.3.5	DMA channel error	109
9.3.6	Interrupts	110
9.4	Initialization information	110
9.4.1	DMA initialization	110
Chapter 10	Direct Memory Access Multiplexer (DMAMUX)	112
10.1	Introduction	112
10.1.1	Features	112
10.1.2	Block diagram	112
10.2	Memory map and register definition	112
10.2.1	Memory map	112
10.2.2	DMA Channel n Mux Configure Register (DMA_MUX_CHn_CFG)	113
10.3	Function description	115
10.3.1	Normal routing from peripheral	115
10.3.2	Always-on routing channel	115
10.4	Initialization/Application information	115
10.4.1	Peripheral DMA routing	115
10.4.2	Always-on DMA routing	115
Chapter 11	Trigger Mux Unit (TMU)	117
11.1	Introduction	117
11.1.1	Block diagram	117
11.2	Signal description	117
11.3	Memory map and register definition	117
11.3.1	Memory map	117
11.3.2	Trigger Mux Unit Configuration (TMU_CFGn)	118
11.4	Function description	118
11.4.1	TMU input channel description	118
11.4.2	TMU output channel description	121
11.5	Initialization application information	122
Chapter 12	Advanced Encryption Standard Accelerator (AES)	123
12.1	Introduction	123
12.1.1	Features	123
12.1.2	Modes of operation	123
12.2	Memory map and register definition	123
12.2.1	Memory map	123

12.2.2	Status Register (AES_STATUS)	124
12.2.3	Control Register (AES_CONTROL)	125
12.2.4	Config Register (AES_CONFIG)	125
12.2.5	Data Input 0 Register (AES_DATA_I0).....	126
12.2.6	Data Input 1 Register (AES_DATA_I1).....	127
12.2.7	Data Input 2 Register (AES_DATA_I2).....	127
12.2.8	Data Input 3 Register (AES_DATA_I3).....	127
12.2.9	Data Output 0 Register (AES_DATA_O0).....	128
12.2.10	Data Output 1 Register (AES_DATA_O1).....	128
12.2.11	Data Output 2 Register (AES_DATA_O2).....	128
12.2.12	Data Output 3 Register (AES_DATA_O3).....	129
12.2.13	Key 0 Register (AES_KEY0)	129
12.2.14	Key 1 Register (AES_KEY1)	129
12.2.15	Key 2 Register (AES_KEY2)	130
12.2.16	Key 3 Register (AES_KEY3)	130
12.2.17	Initialization Vector (AES_IV0).....	130
12.2.18	Initialization Vector (AES_IV1).....	131
12.2.19	Initialization Vector (AES_IV2).....	131
12.2.20	Initialization Vector (AES_IV3).....	132
12.3	Function Description.....	132
12.3.1	ECB.....	132
12.3.2	CBC.....	132
12.3.3	OFB.....	133
12.3.4	CTR.....	133
12.3.5	CFB.....	133
12.3.6	GCM.....	134
12.3.7	CCM.....	137
12.3.8	CMAC	138
12.3.9	XTS	139
12.3.10	Interrupts.....	140
Chapter 13	System Memory Protect Unit (SMPU).....	141
13.1	Introduction.....	141
13.1.1	SMPU slave port assignments	141
13.1.2	Features.....	141
13.1.3	Block diagram	142
13.2	Memory map and register definition	142
13.2.1	Memory map.....	142
13.2.2	Control and Status Register (SMPU_CS).....	143
13.2.3	Slave Port n Error Address Register (SMPU_Sn_EA)	143
13.2.4	Slave Port n Error Status Register (SMPU_Sn_ES)	144
13.2.5	Memory Region n Configuration Word 0 Register (SMPU_MRCn_W0)	144
13.2.6	Memory Region n Configuration Word 1 Register (SMPU_MRCn_W1)	145
13.2.7	Memory Region n Configuration Word 2 Register (SMPU_MRCn_W2)	145
13.2.8	Memory Region n Configuration Word 3 Register (SMPU_MRCn_W3)	147
13.3	Function description	147
13.3.1	Memory region protect unit.....	147
13.3.2	Memory protect address.....	147
13.3.3	Bus master.....	147
13.3.4	Slave port.....	148
13.3.5	Address hit.....	148
13.3.6	Privilege control	148
13.3.7	Access arbitration	149
13.3.8	Error report.....	149
13.4	Initialization application information.....	149
Chapter 14	Watchdog Timer (WDOG).....	150
14.1	Introduction.....	150
14.2	Features	150

14.2.1	Modes of operation	150
14.2.2	Block diagram	151
14.3	Memory map and register definition	151
14.3.1	Memory map	151
14.3.2	Watchdog Parameter (WDOG_PARAM)	151
14.3.3	Watchdog Control and Status Register (WDOG_CS)	152
14.3.4	Watchdog Timeout Register (WDOG_TMO)	153
14.3.5	Watchdog Window Register (WDOG_WIN)	154
14.3.6	Watchdog Counter Register (WDOG_CNT)	154
14.4	Function description	155
14.4.1	Watchdog refresh mechanism	155
14.4.2	Configure the watchdog	155
14.4.3	Interrupt	156
14.4.4	Relief reset	156
14.4.5	Quick testing of the watchdog counter	156
14.4.6	Clock source	157
14.5	Application information	157
14.5.1	WDOG application sequence	157
Chapter 15	External Watchdog Timer (EWDT)	158
15.1	Introduction	158
15.1.1	Features	158
15.1.2	Modes of operation	158
15.1.3	Block diagram	159
15.2	Signal description	159
15.2.1	The EWDT_OUT signal	159
15.2.2	The EWDT_IN signal	159
15.3	Memory map and register definition	159
15.3.1	Memory map	160
15.3.2	External Watchdog Timer Version Parameter Register (EWDT_PARAM)	160
15.3.3	External Watchdog Timer Control and Status Register (EWDT_CS)	160
15.3.4	External Watchdog Timer Compare High Register (EWDT_COMPH)	161
15.3.5	External Watchdog Timer Compare Low Register (EWDT_COMPL)	162
15.3.6	External Watchdog Timer Counter Register (EWDT_CNT)	162
15.4	Function description	163
15.4.1	The EWDT_OUT signal	163
15.4.2	The EWDT_IN signal	163
15.4.3	EWDT counter	163
15.4.4	EWDT compare registers	163
15.4.5	EWDT service operation	163
15.4.6	EWDT interrupts	164
15.5	Application information	164
15.5.1	EWDT application sequence	164
Chapter 16	Clock Monitor Unit (CMU)	165
16.1	Introduction	165
16.1.1	Features	165
16.1.2	Block diagram	165
16.2	Memory map and register definition	165
16.2.1	Memory map	165
16.2.2	Clock Monitor Unit Parameter (CMU_PARAM)	166
16.2.3	Clock Monitor Unit Control (CMU_CTRL)	166
16.2.4	Clock Monitor Unit Status (CMU_STAT)	167
16.2.5	Clock Monitor Unit Monitor Window (CMU_MONWIN)	168
16.2.6	Clock Monitor Unit Clock under Monitor High Compare Threshold for Interrupt (CMU_CUMHCT4INT)	168
16.2.7	Clock Monitor Unit Clock under Monitor Low Compare Threshold for Interrupt (CMU_CUMLCT4INT)	168
16.2.8	Clock Monitor Unit Clock under Monitor High Compare Threshold for Reset	

(CMU_CUMHCT4RST)	169
16.2.9 Clock Monitor Unit Clock under Monitor Low Compare Threshold for Reset (CMU_CUMLCT4RST)	169
16.2.10 Clock Monitor Unit Clock under Monitor Measurement Result (CMU_CUMMEARSLT)	170
16.3 Function description	170
16.3.1 Clock monitor mode	170
16.3.2 Frequency measure mode	171
16.3.3 Monitored clock and reference clock	172
Chapter 17 Error Injection and Report Unit (EIRU)	173
17.1 Introduction	173
17.1.1 Features	173
17.1.2 Block diagram	173
17.2 Memory map and register definition	174
17.2.1 Memory map	174
17.2.2 EIRU Control Register (EIRU_CR)	175
17.2.3 EIRU ECC Error Injection Address (EIRU_EEIADDR)	176
17.2.4 EIRU SRAM n ECC Error Status Register (EIRU_SRAMn_EESR)	176
17.2.5 EIRU SRAM n ECC Fault Address Register (EIRU_SRAMn_FADDR)	177
17.2.6 EIRU SRAM n ECC Fault Data Register (EIRU_SRAMn_FDATA)	178
17.2.7 EIRU SRAM n ECC Check Bit MASK (EIRU_SRAMn_ECCMASK)	178
17.2.8 EIRU SRAM n DATA Bit MASK (EIRU_SRAMn_DATAMASK)	178
17.3 Function description	179
17.3.1 ECC error injection	179
17.3.2 ECC error report	179
17.3.3 Write protection	180
Chapter 18 System Error Report Unit (SERU)	181
18.1 Introduction	181
18.1.1 Features	181
18.1.2 Modes of operation	182
18.1.3 Block diagram	182
18.2 Memory map and register definition	183
18.2.1 Memory map	183
18.2.2 SERU Parameter Register (SERU_PARAM)	183
18.2.3 SERU Configure and Setting Register (SERU_CSR)	184
18.2.4 SERU INT Flag Register (SERU_INTF)	185
18.2.5 SERU Configure Key Register (SERU_CFG_KEY)	185
18.2.6 SERU Channel Fake Fault Injection Register (SERU_CH_FFI)	185
18.2.7 SERU Software Error Generator Register (SERU_SWERR_GNT)	186
18.2.8 SERU Channel Flag Register (SERU_CH_FLAG)	186
18.2.9 SERU Channel Configure Register (SERU_CFG_CHn)	187
18.3 Functional description	188
18.3.1 Interrupts	188
18.3.2 Reset requests	188
18.3.3 EOUT outputs	189
18.3.4 SERU configure mode	189
18.3.5 SERU software error channels	189
18.3.6 SERU false fault injection	189
18.3.7 SERU lock-step function	190
18.3.8 SERU register parity check	190
18.4 Initialization	190
Chapter 19 True Random Number Generator (TRNG)	191
19.1 Introduction	191
19.1.1 Features	191
19.2 Memory map and register definition	191
19.2.1 Memory map	191

19.2.2	TRNG Status Register (TRNG_STATUS)	192
19.2.3	TRNG Configuration Register (TRNG_CONFIG)	193
19.2.4	TRNG CTR-DRBG Control Register (TRNG_CTR_DRBG_CONTROL)	194
19.2.5	TRNG CTR-DRBG Request Size Register (TRNG_CTR_DRBG_REQUEST_SIZE)	194
19.2.6	TRNG CTR-DRBG Configuration Register 1 (TRNG_CTR_DRBG_CONFIG_1)	195
19.2.7	TRNG CTR-DRBG Configuration Register 2 (TRNG_CTR_DRBG_CONFIG_2)	195
19.2.8	TRNG Alarm Register (TRNG_ALARM)	196
19.2.9	TRNG CTR_DRBG Alarm Register (TRNG_CTR_DRBG_ALARM)	196
19.2.10	TRNG Raw Random Control Register (TRNG_RAW_RANDOM_CONTROL)	197
19.2.11	TRNG Raw Random Data Register (TRNG_RAW_RANDOM_DATA)	198
19.2.12	TRNG CTR_DRBG Random Data Register (TRNG_CTR_DRBG_RANDOM_DATA)	198
19.2.13	TRNG Additional Configuration Register 0 (TRNG_ADDITIONAL_CONFIG_0)	198
19.2.14	TRNG Additional Configuration Register 1 (TRNG_ADDITIONAL_CONFIG_1)	199
19.2.15	TRNG Additional Configuration Register 2 (TRNG_ADDITIONAL_CONFIG_2)	200
19.2.16	TRNG Additional Measurement Control Register (TRNG_ADDITIONAL_MEAS_CONTROL)	200
19.2.17	TRNG Additional Measurement Configuration Register (TRNG_ADDITIONAL_MEAS_CONFIG)	201
19.2.18	TRNG Additional Measurement Status Register (TRNG_ADDITIONAL_MEAS_STATUS)	201
19.2.19	TRNG Additional Measurement Data Register 0 (TRNG_ADDITIONAL_MEAS_DATA_0)	202
19.2.20	TRNG Additional Measurement Data Register 1 (TRNG_ADDITIONAL_MEAS_DATA_1)	202
19.2.21	TRNG Interrupt Request Status Register (TRNG_IRQ_STATUS)	202
19.2.22	TRNG Interrupt Request Clear Register (TRNG_IRQ_CLEAR)	203
19.2.23	TRNG Interrupt Request Configuration Register (TRNG_IRQ_CONFIG)	203
19.2.24	TRNG Error Configuration Register (TRNG_ERROR_CONFIG)	204
19.2.25	TRNG Error Control Register (TRNG_ERROR_CONTROL)	204
19.2.26	TRNG Error Status Register (TRNG_ERROR_STATUS)	205
19.2.27	TRNG Monitoring Feature Configuration Register (TRNG_STATS_CONFIG)	205
19.2.28	TRNG Monitoring Feature Control Register (TRNG_STATS_CONTROL)	206
19.2.29	TRNG Monitoring Feature Status Register (TRNG_STATS_DATA)	206
19.3	Function description	207
19.3.1	NORMAL mode	207
19.3.2	Stream mode	208
19.3.3	IRQ management	208
19.3.4	Timeout management	209
Chapter 20	Cyclic Redundancy Check (CRC)	210
20.1	Introduction	210
20.1.1	Features	210
20.1.2	Block diagram	210
20.2	Memory map and register definition	210
20.2.1	CRC memory map	210
20.2.2	CRC Data Register (CRC_DATA)	211
20.2.3	CRC Polynomial Register (CRC_POLY)	211
20.2.4	CRC Control Register (CRC_CTRL)	212
20.3	Function description	212

20.3.1	CRC initialization/reinitialization	213
20.3.2	CRC calculations	213
20.3.3	16-bit CRC	213
20.3.4	32-bit CRC	213
20.3.5	Transpose feature.....	214
20.3.6	CRC result complement.....	214
20.4	Initialization application information	214
Chapter 21	Reset and Boot	215
21.1	Introduction.....	215
21.2	Reset.....	215
21.2.1	Power-on reset (POR)	216
21.2.2	Low voltage detect (LVD)	216
21.2.3	SERU cold reset	217
21.2.4	System reset sources	217
21.2.5	Reset pin.....	218
21.3	Boot.....	218
21.3.1	Boot sources.....	219
21.3.2	Boot sequence	219
Chapter 22	Clock Distribution	220
22.1	Introduction.....	220
22.2	High level clocking diagram.....	220
22.3	Clock definitions	220
22.4	Internal clocking requirements.....	221
22.5	Clock gating.....	222
22.6	Peripheral clock definition.....	222
Chapter 23	System Clock Controller (SCC).....	226
23.1	Introduction.....	226
23.2	Features	226
23.2.1	Block diagram	226
23.3	Memory map and register definition	227
23.3.1	Memory map	227
23.3.2	System Clock Control Parameter (SCC_PARAM)	228
23.3.3	System Clock Control Clock Status Register (SCC_CST).....	228
23.3.4	System Clock Control Clock Configuration Register (SCC_CFG)	230
23.3.5	System Clock Control Clock Out Configuration Register (SCC_CLKOUTCFG) 232	
23.3.6	System Clock Control OSC Control Status Register (SCC_OSCCS).....	232
23.3.7	System Clock Control OSC Configuration Register (SCC_OSCCFG)	233
23.3.8	System Clock Control FIRC Control Status Register (SCC_FIRCCS)	234
23.3.9	System Clock Control SPLLC Control Status Register (SCC_SPLLC).....	235
23.3.10	System Clock Control SPLLC Configuration Register 1 (SCC_SPLLCFG1) ...	236
23.3.11	System Clock Control SPLLC Configuration Register 2 (SCC_SPLLCFG2) ...	237
23.4	Function description	238
23.4.1	FIRC clock mode	238
23.4.2	OSC clock mode.....	238
23.4.3	SPLL clock mode.....	239
23.4.4	SPLL frequency	239
23.4.5	System clock source.....	239
Chapter 24	Peripheral Access & Reset & Clock Controller (PARCC)	241
24.1	Introduction.....	241
24.1.1	Features.....	241
24.1.2	Block diagram	241
24.2	Memory map and register definition	242
24.2.1	Memory map	242
24.2.2	PARCC for WDOG (PARCC_WDOG).....	244
24.2.3	PARCC for EWDT (PARCC_EWDT).....	245

24.2.4	PARCC for STIM (PARCC_STIM).....	246
24.2.5	PARCC for TIM0 (PARCC_TIM0).....	248
24.2.6	PARCC for TIM1 (PARCC_TIM1).....	249
24.2.7	PARCC for TIM2 (PARCC_TIM2).....	251
24.2.8	PARCC for TIM3 (PARCC_TIM3).....	253
24.2.9	PARCC for MCPWM0 (PARCC_MCPWM0).....	254
24.2.10	PARCC for MCPWM1(PARCC_MCPWM1).....	256
24.2.11	PARCC for TDG0 (PARCC_TDG0).....	257
24.2.12	PARCC for TDG1 (PARCC_TDG1).....	259
24.2.13	PARCC for CAN0 (PARCC_CAN0).....	260
24.2.14	PARCC for CAN1 (PARCC_CAN1).....	262
24.2.15	PARCC for CAN2 (PARCC_CAN2).....	264
24.2.16	PARCC for CAN3 (PARCC_CAN3).....	265
24.2.17	PARCC for CAN4 (PARCC_CAN4).....	267
24.2.18	PARCC for CAN5 (PARCC_CAN5).....	268
24.2.19	PARCC for CAN6 (PARCC_CAN6).....	270
24.2.20	PARCC for CAN7 (PARCC_CAN7).....	271
24.2.21	PARCC for UART0 (PARCC_UART0).....	273
24.2.22	PARCC for UART1 (PARCC_UART1).....	275
24.2.23	PARCC for UART2 (PARCC_UART2).....	276
24.2.24	PARCC for UART3 (PARCC_UART3).....	278
24.2.25	PARCC for UART4 (PARCC_UART4).....	279
24.2.26	PARCC for UART5 (PARCC_UART5).....	281
24.2.27	PARCC for SPI0 (PARCC_SPI0).....	282
24.2.28	PARCC for SPI1 (PARCC_SPI1).....	284
24.2.29	PARCC for SPI2 (PARCC_SPI2).....	286
24.2.30	PARCC for SPI3 (PARCC_SPI3).....	287
24.2.31	PARCC for I2C0 (PARCC_I2C0).....	289
24.2.32	PARCC for I2C1 (PARCC_I2C1).....	290
24.2.33	PARCC for I2S0 (PARCC_I2S0).....	292
24.2.34	PARCC for I2S1 (PARCC_I2S1).....	293
24.2.35	PARCC for ADC0 (PARCC_ADC0).....	295
24.2.36	PARCC for ADC1 (PARCC_ADC1).....	297
24.2.37	PARCC for CMP0 (PARCC_CMP0).....	298
24.2.38	PARCC for CRC (PARCC_CRC).....	299
24.2.39	PARCC for AES (PARCC_AES).....	300
24.2.40	PARCC for TRNG (PARCC_TRNG).....	301
24.2.41	PARCC for FLASH (PARCC_FLASH).....	302
24.2.42	PARCC for DMA (PARCC_DMA).....	304
24.2.43	PARCC for DMAMUX (PARCC_DMAMUX).....	305
24.2.44	PARCC for PORTA (PARCC_PORTA).....	306
24.2.45	PARCC for PORTB (PARCC_PORTB).....	308
24.2.46	PARCC for PORTC (PARCC_PORTC).....	309
24.2.47	PARCC for PORTD (PARCC_PORTD).....	311
24.2.48	PARCC for PORTE (PARCC_PORTE).....	312
24.2.49	PARCC for TMU (PARCC_TMU).....	314
24.2.50	PARCC for REGFILE (PARCC_REGFILE).....	315
24.2.51	PARCC for SMPU (PARCC_SMPU).....	316
24.2.52	PARCC for GPIO (PARCC_GPIO).....	317
24.3	Function description	318
24.3.1	Peripheral register write protection.....	318
24.3.2	Register access protection in supervisor mode	318
24.3.3	Reset peripheral	319
24.3.4	Functional clock source switch	319
24.3.5	Functional clock source divide.....	319
24.3.6	Clock mode	319
24.4	Initialization/Application information	319
24.4.1	Supervisor mode write access control.....	319

24.4.2	Reset control.....	319
24.4.3	Peripheral clock mode control	320
Chapter 25	Power Management Unit (PMU)	321
25.1	Introduction.....	321
25.1.1	Features.....	321
25.1.2	Block diagram	321
25.2	Power modes description	322
25.3	Memory map and register definition	323
25.3.1	Memory map.....	323
25.3.2	PMU Lock Protection Register (PMU_LOCK)	323
25.3.3	PMU LPO32K Configure Register (PMU_LPO_32K_CFG)	324
25.3.4	PMU Isolation Clear Register (PMU_ISO_CLR)	324
25.3.5	PMU Control and Status Register (PMU_CSR)	325
25.3.6	PMU LDO Enable Configure Register (PMU_LDO_EN_CFG).....	326
25.3.7	PMU LDO LVD Status Register (PMU_LDO_LVD_STATUS).....	327
25.3.8	PMU LDO LVD Configure Register (PMU_LDO_LVD_CFG).....	327
25.4	Function description	329
25.4.1	Clocking gating	329
25.4.2	Power mode transitions	329
25.4.3	Module operation in available power modes	330
25.4.4	POR (power-on reset).....	331
25.4.5	LVD (low voltage detect) circuit	331
Chapter 26	System Reset and Mode Control (SRMC)	333
26.1	Introduction.....	333
26.1.1	Features.....	333
26.2	Memory map and register definition	333
26.2.1	Memory map.....	333
26.2.2	Parameter Register (SRMC_PARAM).....	334
26.2.3	System Reset Status Register (SRMC_SRS)	335
26.2.4	Reset Control Register (SRMC_CTRL).....	337
26.2.5	Sticky System Reset Status Register (SRMC_SSRS).....	338
26.2.6	System Reset Interrupt Enable Register (SRMC_SRIE)	340
26.2.7	Power Mode Protection register (SRMC_PMPORT).....	341
26.2.8	Power Mode Control Register (SRMC_PMCTRL)	342
26.2.9	Power Mode Status Register (SRMC_PMSTAT).....	342
26.2.10	DSM Wakeup Control Register (SRMC_DSMWUPC1)	343
26.2.11	DSM Wakeup Control Register (SRMC_DSMWUPC2)	344
26.2.12	DSM Wakeup Status Register (SRMC_DSMWUPS).....	345
26.3	Function description	346
26.3.1	Reset generate	346
26.3.2	Mode control	346
26.4	Reset and interrupt.....	351
Chapter 27	Flash	352
27.1	Introduction.....	352
27.1.1	Glossary.....	352
27.1.2	Features.....	353
27.1.3	Block diagram	353
27.2	Flash memory.....	353
27.3	Memory map and register definition	354
27.3.1	Memory map.....	354
27.3.2	Flash Status Register (FLASH_FSTAT).....	355
27.3.3	Flash Configuration Register (FLASH_FCNG)	357
27.3.4	Flash Control Register (FLASH_FCTRL)	358
27.3.5	Flash Command Register (FLASH_FCMD)	359
27.3.6	Flash Address Register (FLASH_FADDR).....	359
27.3.7	Flash End Address Register (FLASH_FADDR_END).....	359

27.3.8	Flash Data0 Register (FLASH_FDATA0)	360
27.3.9	Flash Data1 Register (FLASH_FDATA1)	360
27.3.10	Flash Data2 Register (FLASH_FDATA2)	360
27.3.11	Flash Data3 Register (FLASH_FDATA3)	360
27.3.12	Flash Bus Error Ignored Start Address Register (FLASH_BED_ADDR_STRT) 361	
27.3.13	Flash Bus Error Ignored End Phrase Address Register (FLASH_BED_ADDR_END)	361
27.3.14	Flash ECC Injection Address Register (FLASH_ECC_IN_ADDR)	361
27.3.15	Flash ECC Injection Control Register (FLASH_ECC_IN_CTRL)	362
27.3.16	Flash Multi-bit Error State Register (FLASH_MBE_STATE)	362
27.3.17	Flash Single-bit Error State Register (FLASH_SBE_STATE)	363
27.4	Function description	363
27.4.1	Interrupts	363
27.4.2	Bus error due to multi-bit ECC error	364
27.4.3	Flash fixed clock	364
27.4.4	Security and backdoor	364
27.4.5	ECC injection	365
27.4.6	Flash command operations	365
27.4.7	Flash command descriptions	367
27.5	Initialization application information	374
Chapter 28	Register File (REGFILE)	376
28.1	Introduction	376
28.1.1	Features	376
28.2	Memory Map and Register Definition	376
28.2.1	Memory Map	376
28.2.2	REGFILE_REGn	376
Chapter 29	Motor Control PWM (MCPWM)	377
29.1	Introduction	377
29.1.1	Features	377
29.1.2	Block diagram	378
29.1.3	Instantiation and channel information	379
29.2	Memory map and register definition	380
29.2.1	Memory map	380
29.2.2	MCPWM Time Base Register (MCPWM_TIMEBASE)	381
29.2.3	MCPWM Counter n Register (MCPWM_CNTn)	381
29.2.4	MCPWM Counter n Modulo Value Register (MCPWM_MODn)	382
29.2.5	MCPWM Deadtime Value n Register (MCPWM_DTVAln)	382
29.2.6	MCPWM Global Control Register (MCPWM_GLBcr)	382
29.2.7	MCPWM Global Status Register (MCPWM_GLBSR)	385
29.2.8	MCPWM Channel n Mode Configuration Register (MCPWM_CFGn)	387
29.2.9	MCPWM Channel n Value Register (MCPWM_CVn)	388
29.2.10	MCPWM Output Control Register (MCPWM_OUTCR)	388
29.2.11	MCPWM Output Trigger Register (MCPWM_OUTTRIG)	391
29.2.12	MCPWM Channel Output Software Control Register (MCPWM_OUTSWCR) 392	
29.2.13	MCPWM Pairs Of Channel Control Register (MCPWM_PCR)	393
29.2.14	MCPWM Fault Control Register (MCPWM_FLTCR)	398
29.2.15	MCPWM Fault Status Register (MCPWM_FLTSR)	400
29.2.16	MCPWM Fault A Safe Status Value Register (MCPWM_FLTASS)	401
29.2.17	MCPWM Fault B Safe Status Value Register (MCPWM_FLTBSS)	403
29.2.18	MCPWM Synchronization Register (MCPWM_SYNC)	404
29.2.19	MCPWM Reload Register (MCPWM_RELOAD)	405
29.2.20	MCPWM Reload Frequency Register (MCPWM_LDFREQ)	407
29.2.21	MCPWM MODn Dither Register (MCPWM_MODn_DITHER)	407
29.2.22	MCPWM CVn Dither Register (MCPWM_CVn_DITHER)	408
29.3	Function description	408

29.3.1	Time base unit	408
29.3.2	Counter	409
29.3.3	Channels	410
29.3.4	PWM output mode	411
29.3.5	Deadtime insertion	419
29.3.6	Generation of match trigger and initial trigger	420
29.3.7	Fault protection	420
29.3.8	Registers synchronization	423
29.3.9	Dithering	425
29.4	Interrupt and DMA signals	428
29.4.1	DMA	428
29.4.2	Interrupt	428
Chapter 30	Timer (TIM)	430
30.1	Introduction	430
30.1.1	Features	430
30.1.2	Block diagram	431
30.1.3	Instantiation and channel information	431
30.2	Signal description	432
30.3	Memory map and register definition	432
30.3.1	Memory map	432
30.3.2	TIM Time Base Register (TIM_TIMEBASE)	433
30.3.3	TIM Counter Register (TIM_CNT)	434
30.3.4	TIM Counter Initial Value Register (TIM_CNTINIT)	434
30.3.5	TIM Counter Modulo Value Register (TIM_MOD)	435
30.3.6	TIM Global Control Register (TIM_GLBCCR)	435
30.3.7	TIM Global Status Register (TIM_GLBSTR)	437
30.3.8	TIM Channel n Mode Configuration Register (TIM_CMCn)	438
30.3.9	TIM Channel n Capture/Compare Value Register (TIM_CCVn)	439
30.3.10	TIM Output Control Register (TIM_OUTCR)	440
30.3.11	TIM Channel Output Software Control Register (TIM_OUTSWCR)	443
30.3.12	TIM Pairs of Channel Control Register (TIM_PCR)	445
30.3.13	TIM Input Filter Register (TIM_FILTER)	449
30.3.14	TIM Fault Control Register (TIM_FLTCR)	450
30.3.15	TIM Fault Status Register (TIM_FLTSR)	451
30.3.16	TIM Synchronization Register (TIM_SYNC)	452
30.3.17	TIM Reload Register (TIM_RELOAD)	454
30.3.18	TIM Half Cycle Reload Value Register (TIM_HCV)	455
30.4	Function description	456
30.4.1	Time base unit	456
30.4.2	Counter mode	457
30.4.3	Channels mode	458
30.4.4	Input capture mode	459
30.4.5	Dual-edge capture mode	462
30.4.6	Input pulse filter for capture mode	465
30.4.7	Output compare mode	465
30.4.8	PWM output mode	467
30.4.9	Generation of match trigger and initial trigger	473
30.4.10	Channel output software control	474
30.4.11	Fault input control	474
30.4.12	Registers synchronization	477
30.4.13	Interrupt and DMA signals	480
Chapter 31	System Timer (STIM)	481
31.1	Introduction	481
31.1.1	Features	481
31.1.2	Block diagram	481
31.1.3	Instantiation and channel information	482
31.2	Signal description	482

31.3	Memory map and register definition	482
31.3.1	Memory map	482
31.3.2	STIM Channel n Counter Register (STIM_CNTn)	482
31.3.3	STIM Channel n Compare Value Register (STIM_CVn)	482
31.3.4	STIM Channel n Status and Control Register (STIM_SCn)	483
31.4	Function description	485
31.4.1	Counter clock source	486
31.4.2	Counter mode	486
31.4.3	Counter pulse source	486
31.4.4	Counter match	486
31.5	Initialization application information	487
Chapter 32	Real Time Counter (RTC)	488
32.1	Introduction	488
32.1.1	Features	488
32.1.2	Block diagram	488
32.1.3	Instantiation information	489
32.2	Signal description	489
32.3	Memory map and register definition	489
32.3.1	Memory map	489
32.3.2	RTC Control and Status Register (RTC_CSR)	489
32.3.3	RTC Interrupt Enable Register (RTC_IER)	491
32.3.4	RTC Alarm Counter Register (RTC_ALARM_CNTR)	492
32.3.5	RTC One Second Counter Register (RTC_ONE_SECOND_CNTR)	493
32.3.6	RTC Match Counter Register (RTC_MATCH_CNTR)	493
32.3.7	RTC Compensation Register (RTC_COMPENSATIONR)	494
32.3.8	RTC Lock Register (RTC_LOCKR)	495
32.3.9	RTC Periodic Timer Counter (RTC_PERIODIC_TMR_CNT)	495
32.3.10	RTC Periodic Timer Match Counter (RTC_PERIODIC_TMR_MATCH_CNT)	496
32.4	Function description	496
32.4.1	Oscillator control	496
32.4.2	Timer counter	496
32.4.3	Alarm match	497
32.4.4	Periodic interrupt	497
32.4.5	Register lock	497
32.4.6	Compensation	497
32.5	Initialization application information	497
32.5.1	Alarm match interrupt setup via OSC32K	497
32.5.2	Alarm match interrupt setup via LPO32K	498
32.5.3	Seconds interrupt setup via OSC32K	498
Chapter 33	Trigger Delay Generator (TDG)	499
33.1	Introduction	499
33.1.1	Features	499
33.1.2	Block diagram	499
33.1.3	Instantiation information	499
33.2	Memory map and register definition	500
33.2.1	Memory map	500
33.2.2	TDG Parameter (TDG_PARAM)	501
33.2.3	TDG Control Register 1 (TDG_CTRL1)	501
33.2.4	TDG Control Register 2 (TDG_CTRL2)	502
33.2.5	TDG Status Register (TDG_STAT)	503
33.2.6	TDG Modulate Register (TDG_MOD)	504
33.2.7	TDG Counter Register (TDG_CNT)	505
33.2.8	TDG Channel n Control Register (TDG_CHnCTRL)	505
33.2.9	TDG Channel n Delay Output 0 Offset Register (TDG_CHnDO0OFS)	506
33.2.10	TDG Channel n Delay Output 1 Offset Register (TDG_CHnDO1OFS)	507
33.2.11	TDG Channel n Delay Output 2 Offset Register (TDG_CHnDO2OFS)	507
33.2.12	TDG Channel n Delay Output 3 Offset Register (TDG_CHnDO3OFS)	507

33.2.13	TDG Channel n Delay Output 4 Offset Register (TDG_CHnDO4OFS)	508
33.2.14	TDG Channel n Delay Output 5 Offset Register (TDG_CHnDO5OFS)	508
33.2.15	TDG Channel n Delay Output 6 Offset Register (TDG_CHnDO6OFS)	509
33.2.16	TDG Channel n Delay Output 7 Offset Register (TDG_CHnDO7OFS)	509
33.2.17	TDG Channel n Complete Delay Output Interrupt Delay Register (TDG_CHnCDOINTDLY).....	509
33.3	Function description	510
33.3.1	Input trigger.....	510
33.3.2	Counter	510
33.3.3	Delayed output.....	511
33.3.4	Interrupt.....	511
Chapter 34	Analog-to-digital converter (ADC)	512
34.1	Introduction.....	512
34.1.1	Features.....	512
34.1.2	Block diagram	512
34.1.3	Instantiation and channel information.....	513
34.2	Signal description	513
34.3	Memory map and register definition	514
34.3.1	Memory map.....	514
34.3.2	ADC Parameter Register (ADC_PARAM)	514
34.3.3	ADC Control Register (ADC_CTRL).....	515
34.3.4	ADC Status Register (ADC_STAT)	516
34.3.5	ADC Interrupt Enable Register (ADC_IE).....	517
34.3.6	ADC Configure Information (ADC_CFG).....	517
34.3.7	ADC Compare Value Register (ADC_CV).....	519
34.3.8	ADC Data Register (ADC_DATA_RD)	520
34.3.9	ADC FIFO Configuration Register (ADC_FCTRL)	520
34.3.10	ADC Software Trigger Register (ADC_SWTRIG)	521
34.3.11	ADC Command Buff Register0 (ADC_CMD_BUFF0)	521
34.3.12	ADC Command Buff Register1 (ADC_CMD_BUFF1)	522
34.4	Function description	523
34.4.1	ADC voltage reference select.....	523
34.4.2	Channel and mode select.....	523
34.4.3	Start trigger select.....	523
34.4.4	Loop trigger mode and map trigger mode	524
34.4.5	Compare function	524
34.4.6	Conversion function	525
34.4.7	Conversion time	525
Chapter 35	Comparator (CMP).....	527
35.1	Introduction.....	527
35.2	Features	527
35.2.1	Block diagram	527
35.2.2	Instantiation and channel information.....	527
35.2.3	Signal description	528
35.3	Memory map and register definition	528
35.3.1	Memory map.....	528
35.3.2	CMP Parameter Register (CMP_PARAM)	528
35.3.3	Comparator Control Register 0 (CMP_CCR0)	529
35.3.4	Comparator Control Register 1 (CMP_CCR1)	529
35.3.5	Comparator Control Register 2 (CMP_CCR2)	531
35.3.6	CMP DAC Control Register (CMP_DCR).....	532
35.3.7	CMP Interrupt Enable Register (CMP_IER)	532
35.3.8	Comparator Status Register (CMP_CSR).....	533
35.4	Function description	533
35.4.1	Polarity selection.....	533
35.4.2	Basic mode	534
35.4.3	Window control	534

35.4.4	Filter mode	534
35.4.5	Sample mode	534
35.4.6	DAC operation	534
35.4.7	IRQ/DMA request	534
Chapter 36	Universal Asynchronous Receiver/Transmitter (UART)	536
36.1	Introduction	536
36.1.1	Features	536
36.1.2	Block diagram	536
36.1.3	Instantiation information	538
36.2	Signal description	538
36.3	Memory map and register definition	538
36.3.1	Memory map	538
36.3.2	Receive Buffer Register (UART_RBR)	539
36.3.3	Transmit Holding Register (UART_THR)	540
36.3.4	Divisor Latch Low (UART_DLL)	540
36.3.5	Divisor Latch High (UART_DLH)	541
36.3.6	Interrupt Enable Register (UART_IER)	541
36.3.7	Interrupt Identity Register (UART_IIR)	542
36.3.8	FIFO Control Register (UART_FCR)	543
36.3.9	Line Control Register (UART_LCR)	544
36.3.10	Autoflow Control Register (UART_AFCR)	545
36.3.11	Line Status Register (UART_LSR)	546
36.3.12	Modem Status Register (UART_MSR)	548
36.3.13	UART Status Register (UART_USR)	548
36.3.14	Fraction Divisor Register (UART_FD)	549
36.3.15	Receive Address Register (UART_RAR)	549
36.3.16	Transmit Address Register (UART_TAR)	550
36.3.17	Line Extended Control Register (UART_LCR_EXT)	550
36.3.18	LIN Mode Control Register (UART_LIN_CTL)	551
36.3.19	LIN Response Length Register (UART_LIN_RSP_LENGTH)	552
36.3.20	LIN ID Register (UART_LIN_ID_VALUE)	552
36.3.21	LIN Checksum Register (UART_LIN_CHECKSUM)	553
36.3.22	LIN Delimiter Field Length (UART_LIN_DEL_LENGTH)	553
36.3.23	LIN PID Filter 0 (UART_LIN_PID_FILTER_0)	554
36.3.24	LIN PID Filter 1 (UART_LIN_PID_FILTER_1)	554
36.3.25	LIN PID Filter 2 (UART_LIN_PID_FILTER_2)	555
36.3.26	LIN PID Filter 3 (UART_LIN_PID_FILTER_3)	555
36.3.27	LIN PID Filter Vague (UART_LIN_PID_FILTER_VAGUE)	556
36.3.28	LIN PID Filter Control Register (UART_LIN_PID_FILTER_CTRL)	556
36.4	UART function description	557
36.4.1	Transmit programming	557
36.4.2	Receive programming	558
36.4.3	Tx/Rx data format	558
36.4.4	Baud rate clock generation	559
36.4.5	9-Bit data transfer	559
36.4.6	Tx&Rx FIFO operation	560
36.4.7	DMA and interrupt	560
36.5	LIN function description	561
36.5.1	Frame auto send and receive	561
36.5.2	Checksum calculation and check	563
36.5.3	PID parity calculation and check	563
36.5.4	PID filter	564
36.5.5	Baud rate auto sync	564
Chapter 37	Serial Peripheral Interface (SPI)	565
37.1	Introduction	565
37.1.1	Features	565
37.1.2	Block diagram	565

37.1.3	Instantiation Information	566
37.2	Signal description	566
37.3	Memory map and register definition	566
37.3.1	Memory map	566
37.3.2	Control Register 0 (SPI_CTRLR0).....	567
37.3.3	Control Register 1 (SPI_CTRLR1).....	568
37.3.4	SPI Enable Register (SPI_SSENr).....	568
37.3.5	Baud Rate Select Register (SPI_BAUDR)	569
37.3.6	FIFO Threshold Level Register (SPI_FTLR)	570
37.3.7	FIFO Level Register (SPI_FLR).....	570
37.3.8	Status Register (SPI_SR)	571
37.3.9	Interrupt Enable Register (SPI_IER)	572
37.3.10	Interrupt Status Register (SPI_ISR)	572
37.3.11	Transmit FIFO Overflow Interrupt Clear Register (SPI_TXOICR).....	574
37.3.12	Receive FIFO Overflow Interrupt Clear Register (SPI_RXOICR)	574
37.3.13	Receive FIFO Underflow Interrupt Clear Register (SPI_RXUICR)	575
37.3.14	Interrupt Clear Register (SPI_ICR).....	575
37.3.15	DMA Control Register (SPI_DMAGR)	575
37.3.16	DMA Transmit Data Level (SPI_DMATDLR).....	576
37.3.17	DMA Receive Data Level (SPI_DMARDLR)	577
37.3.18	Data Register (SPI_DR_LOW)	577
37.3.19	Data Register (SPI_DR_HIGH).....	578
37.4	Function description	578
37.4.1	Operation mode	578
37.4.2	Transfer mode.....	578
37.4.3	Clock configuration	579
37.4.4	Transfer formats	579
37.4.5	Data frame size and FIFO	580
37.4.6	DMA and interrupt.....	580
37.5	Initialization application information	580
Chapter 38	Inter-Integrated Circuit (I2C).....	582
38.1	Introduction.....	582
38.1.1	I2C terminology.....	582
38.1.2	Features.....	582
38.1.3	Block diagram	582
38.1.4	Instantiation information.....	583
38.2	Signal description	583
38.3	Memory map and register definition	583
38.3.1	Memory map.....	583
38.3.2	I2C Parameter Register (I2C_PARAMETER)	585
38.3.3	I2C Configure0 Register (I2C_CONFIG0)	585
38.3.4	I2C Configure1 Register (I2C_CONFIG1)	586
38.3.5	I2C SDA Setup Timing Register (I2C_SDA_SETUP_TIMING)	588
38.3.6	I2C SDA Hold Timing Register (I2C_SDA_HOLD_TIMING)	589
38.3.7	I2C DMA Control Register (I2C_DMA_CTRL).....	590
38.3.8	I2C Interrupt Enable Register (I2C_INT_ENABLE)	591
38.3.9	I2C Rx/Tx Data Buffer and Command Register (I2C_COMMAND_DATA) ..	592
38.3.10	I2C Destination Address Register (I2C_DEST_ADDR).....	593
38.3.11	I2C Slave Address Register (I2C_SLAVE_ADDR).....	593
38.3.12	Standard Speed I2C SCL High Count Register (I2C_STD_SCL_HCNT)	594
38.3.13	Standard Speed I2C SCL Low Count Register (I2C_STD_SCL_LCNT).....	594
38.3.14	Fast Mode I2C SCL High Count register (I2C_FST_SCL_HCNT)	595
38.3.15	Fast Mode I2C SCL Low Count Register (I2C_FST_SCL_LCNT)	596
38.3.16	High Speed I2C SCL High Count Register (I2C_HS_SCL_HCNT).....	596
38.3.17	High Speed I2C SCL Low Count Register (I2C_HS_SCL_LCNT)	597
38.3.18	I2C Receive FIFO Water Mark Register (I2C_RXFIFO_WATER_MARK)	597
38.3.19	I2C Transmit FIFO Water Mark Register (I2C_TXFIFO_WATER_MARK)	598

38.3.20	I2C Transmit FIFO Valid Entry Counter Register (I2C_TX_FIFO_CNT)	598
38.3.21	I2C Receive FIFO Valid Entry Counter Register (I2C_RX_FIFO_CNT)	598
38.3.22	I2C Fast and Standard Spike Suppression Counter Register (I2C_FSTD_SPKCNT)	599
38.3.23	I2C High Speed Spike Suppression Counter Register (I2C_HS_SPKCNT) ..	599
38.3.24	I2C Interrupt Status Register (I2C_STATUS0)	600
38.3.25	I2C Status Register (I2C_STATUS1)	601
38.3.26	I2C Error Status Register (I2C_ERROR_STATUS)	603
38.3.27	I2C SCL Stuck at Low Timeout Register (I2C_SCL_LOW_TIMEOUT)	605
38.3.28	I2C SDA Stuck at Low Timeout Register (I2C_SDA_LOW_TIMEOUT)	605
38.3.29	I2C Read Clear Error Status Register (I2C_RD_CLR_ERR_STATUS)	605
38.4	Function description	606
38.4.1	I2C protocol behavior	606
38.4.2	Multi-master arbitration and clock synchronization	611
38.4.3	Bus recovering feature	612
38.4.4	Spike filter counter	613
38.4.5	SDA hold timing	614
38.4.6	I2C SCL configuration	615
38.5	I2C DMA feature	616
38.6	Initialization application information	617
Chapter 39	Inter-IC Sound (I2S)	618
39.1	Introduction	618
39.1.1	I2S terminology	618
39.1.2	I2S overview	618
39.1.3	Features	619
39.1.4	Block diagram	619
39.1.5	Instantiation information	620
39.2	Memory map and register definition	620
39.2.1	I2S memory map	620
39.2.2	I2S Control Register (I2S_CONTROL)	621
39.2.3	I2S Master Configuration Register (I2S_MASTER_CONFIG)	622
39.2.4	I2S Channeln Left Data Register (I2S_CHn_LEFT_DATA)	623
39.2.5	I2S Channeln Right Data Register (I2S_CHn_RIGHT_DATA)	623
39.2.6	I2S Channeln Configuration Register (I2S_CHn_CFG)	624
39.2.7	I2S Channeln Interrupt Configuration (I2S_CHn_INT_CFG)	625
39.2.8	I2S Channeln Trigger Level Configuration Register (I2S_CHn_TRIGGER_LEVEL_CFG)	626
39.2.9	I2S Cycle Tx Data Register (I2S_CYCLE_TX_DATA)	627
39.2.10	I2S Cycle Tx Reset Register (I2S_CYCLE_TX_RESET)	627
39.2.11	I2S Cycle Rx Data Register (I2S_CYCLE_RX_DATA)	628
39.2.12	I2S Cycle Rx Reset Register (I2S_CYCLE_RX_RESET)	628
39.3	Function description	629
39.3.1	I2S module enable	629
39.3.2	I2S master clock generator	629
39.3.3	I2S as transmitter	629
39.3.4	I2S as receiver	631
39.3.5	I2S configuration example	633
Chapter 40	Controller Area Network (CAN)	635
40.1	Introduction	635
40.1.1	Features	635
40.1.2	Modes of operation	636
40.1.3	Block diagram	637
40.1.4	Instantiation information	637
40.2	Signal description	638
40.3	Memory map and register definition	638
40.3.1	Memory map	638
40.3.2	CAN Module Configure Register (CAN_MCR)	641

40.3.3	Control 1 Register (CAN_CTRL1)	646
40.3.4	Free Running Timer Register (CAN_TIMER)	650
40.3.5	Rx Mailbox Global Mask Register (CAN_RXMGMSK)	651
40.3.6	Rx 14 Mask Register (CAN_RX14MASK)	651
40.3.7	Rx 15 Mask Register (CAN_RX15MASK)	652
40.3.8	Error Counter Register (CAN_ECR)	652
40.3.9	CAN Error Status Register (CAN_ESR1)	653
40.3.10	Interrupt Mask 2 Register (CAN_IMASK2)	656
40.3.11	Interrupt Mask 1 Register (CAN_IMASK1)	656
40.3.12	Interrupt Flag 2 Register (CAN_IFLAG2)	657
40.3.13	Interrupt Flag 1 Register (CAN_IFLAG1)	657
40.3.14	Control 2 Register (CAN_CTRL2)	659
40.3.15	Error and Status 2 Register (CAN_ESR2)	661
40.3.16	CRC Register (CAN_CRCCR)	662
40.3.17	Legacy FIFO Global Mask Register (CAN_RXFGMASK)	662
40.3.18	RX FIFO ID Indicator (CAN_RXFIR)	662
40.3.19	CAN Bit Timing Register (CAN_CBT)	663
40.3.20	Interrupt Mask 4 Register (CAN_IMASK4)	664
40.3.21	Interrupt Mask 3 Register (CAN_IMASK3)	664
40.3.22	Interrupt Flag 4 Register (CAN_IFLAG4)	664
40.3.23	Interrupt Flag 3 Register (CAN_IFLAG3)	665
40.3.24	Message Buffer Control and Status Register (CAN_MBn_CS)	665
40.3.25	Message Buffer ID Register (CAN_MBn_ID)	667
40.3.26	Message Buffer Data03 Register (CAN_MBn_DATA03)	667
40.3.27	Message Buffer Data47 Register (CAN_MBn_DATA47)	668
40.3.28	RX Individual Mask Register (CAN_RXIMRn)	668
40.3.29	Memory Error Control Register (CAN_MECR)	668
40.3.30	Error Injection Address Register (CAN_ERRIAR)	670
40.3.31	Error Injection Data Pattern Register (CAN_ERRIDPR)	671
40.3.32	Error Injection Parity Pattern Register (CAN_ERRIPPR)	671
40.3.33	Error Report Address Register (CAN_RERRAR)	672
40.3.34	Error Report Data Register (CAN_RERRDR)	672
40.3.35	Error Report Syndrome Register (CAN_RERRSYNR)	673
40.3.36	Error Status Register (CAN_ERRSR)	674
40.3.37	Pretend Network Control 1 Register (CAN_CTRL1_PN)	675
40.3.38	Pretend Network Control 2 Register (CAN_CTRL2_PN)	676
40.3.39	Pretend Network Wakeup Match Register (CAN_WU_MTC)	677
40.3.40	Pretend Network ID Filter 1 Register (CAN_FLT_ID1)	678
40.3.41	Pretend Network DLC Filter Register (CAN_FLT_DLC)	678
40.3.42	Pretend Network Payload Low Filter 1 Register (CAN_PL1_LO)	679
40.3.43	Pretend Network Payload High Filter 1 Register (CAN_PL1_HI)	680
40.3.44	Pretended Networking ID Filter 2 Register / ID Mask Register (CAN_FLT_ID2_IDMASK)	680
40.3.45	Pretended Networking Payload Low Filter 2 Register / Payload Low Mask Register (CAN_PL2_PLMASK_LO)	681
40.3.46	Pretended Networking Payload High Filter 2 Low Order Bits / Payload High Mask Register (CAN_PL2_PLMASK_HI)	682
40.3.47	Wake Up Message Buffer Register for C/S (CAN_WMBn_CS)	682
40.3.48	Wake Up Message Buffer Register for ID (CAN_WMBn_ID)	683
40.3.49	Wake Up Message Buffer Register for Data 0-3 (CAN_WMBn_D03)	684
40.3.50	Wake Up Message Buffer Register for Data 4-7 (CAN_WMBn_D47)	684
40.3.51	CAN FD Control Register (CAN_FDCTRL)	684
40.3.52	CAN FD Bit Timing Control Register (CAN_FDCBT)	687
40.3.53	CAN FD CRC Register (CAN_FDCRC)	688
40.3.54	Message buffer structure	689
40.3.55	CAN MB memory partition	692
40.3.56	CAN MB memory partition for CAN FD	700
40.3.57	Legacy Rx FIFO structure	701

40.4	Function description	704
40.4.1	Transmit process	704
40.4.2	Arbitration process	705
40.4.3	Arbitration process (continued)	705
40.4.4	Receive process	706
40.4.5	Match process	708
40.4.6	Move process	710
40.4.7	Data coherence	711
40.4.8	Legacy RX FIFO	713
40.4.9	CAN protocol related features	715
40.4.10	Transceiver delay compensation	718
40.4.11	Remote frame	720
40.4.12	Overload frames	721
40.4.13	Message buffer time stamp	721
40.4.14	Protocol timing	721
40.4.15	Arbitration and matching timing	723
40.4.16	TX arbitration start delay	723
40.4.17	Detection and correction of memory errors	725
40.4.18	CLOCK domain and restrictions	728
40.4.19	Interrupt and DMA signals	731
Chapter 41	Debug	733
41.1	Introduction	733
41.2	Block diagram	733
41.3	ROM table	735
41.4	Debug port pin descriptions	735
41.5	ADM-AP status and control registers	736
41.5.1	ADM-AP control register	737
41.5.2	ADM-AP status register	738
41.6	Debug reset	738
41.7	Debug in low-power modes	739
41.8	Debug and security	739
Appendix A	Release History	740
Appendix B	Revision History	741

Chapter 1

About This Manual

1.1 Introduction

The Z20K14xMC Series Reference Manual describes the functionality of the superset device of the Z20K14xMC series. To locate the available features and register implementation of a specific Z20K14xMC derivative (derivative device), please refer to the respective Chip-specific module information.

1.2 Audience

The intended audience for this document is system software and hardware developers and applications programmers who want to develop products with this device. It is assumed that readers understand operating systems, microprocessor system design, and basic principles of software and hardware.

1.3 Module descriptions

Each module chapter has two main parts:

- The Chip-specific [module name] section provides details about chip, such as the number of module instances on the chip and the connections between the module and other modules. Read this section **first** because the content in this section is essential for readers to understand the information in other sections of this chapter.
- The subsequent sections provide general information about the module, including the signals, registers, and functional descriptions.

1.4 Register descriptions

Module chapters present register information about:

- Memory maps, which contain
 - An offset from the module's base address
 - The name and acronym/abbreviation of each register
 - The width of each register (in bits)
 - Each register's reset value
- Field-description tables
- Associated text

1.5 Conventions

1.5.1 Numbering systems

The following suffixes identify different numbering systems:

Suffixes	Numbering systems
b	Binary number system. Binary number system has two symbols: 0 and 1. We shall denote a binary number with a suffix b. For example, the binary equivalent of decimal number 5 is 101b.
d	Decimal number system. We shall denote decimal numbers with an optional suffix d when ambiguity arises. In general, decimal numbers are displayed without a suffix.

Suffixes	Numbering systems
h	Hexadecimal number system. In most cases, we shall denote a hexadecimal number with prefix 0x. For example, the hexadecimal equivalent of decimal number 60 is 0x3Ch.

1.5.2 Acronyms and abbreviations

The following table lists acronyms and abbreviations used in this document.

Acronyms/Abbreviations	Whole Word/Description
AHB	Advanced High-performance Bus, version of AMBA
AMBA	Advanced Microcontroller Bus Architecture
APB	Advanced Peripheral Bus, version of AMBA
LSB	Least Significant Bit
MSB	Most Significant Bit

1.5.3 Definition of terms

The following terms are used with special meanings:

Term	Meaning
asserted	A state of a signal. An asserted signal indicates that a signal is active. An active-high signal is asserted, when the high level (1). An active-low signal is asserted, when the low level (0).
deasserted	A state of a signal. A deasserted signal indicates that a signal is inactive. An active-high signal is deasserted, when the low level (0). An active-low signal is deasserted, when the high level (1). In some cases, deasserted signals are described as <i>negated</i> .
reserved	Memory space, register, field, or programming setting. Device operation is not guaranteed when reserved locations are written to any value. Do not modify the default value of a reserved programming setting, such as the reset value of a reserved register field. The values written to should be consistent with the original values, and inconsistent writes will result in undefined behavior. Consider undefined locations in memory to be reserved.
RO	Read-only
WO	Write-only
RW	Read-Write
W1C	Write One to Clear bit. A bit field of register that requires to write 1 to clear.
W0C	Write Zero to Clear bit A bit field of register that requires to write 0 to clear

Chapter 2

Introduction

2.1 Overview

The Z20K14xMC series extend the scalability of the automatic ARM® Cortex®-M4F MCU portfolio with a more robust adaptability of environment. With wide range power supply, the Z20K14xMC series devices suit well with a broad range of applications. The Z20K14xMC series provide low pin-count options, targeting at the cost-effective applications. Designed with a wide range of memory, peripherals, and package options, the Z20K14xMC product series can easily migrate to an MCU family or integrate with other MCU families.

2.2 Z20K14xMC series introduction

This device is a 32-bit Z20K micro-controller based on ARM® Cortex®-M4F processor with up to 160 MHz CPU performance, up to 2MB P-Flash, 128KB D-Flash and up to 256KB SRAM.

The Z20K14xMC series have the following main features:

- 32-bit ARM Cortex-M4F core, up to 160 MHz
- Up to 2 MB code flash and up to 128 KB data flash (support software solution to emulate EEPROM)
- Up to 256 KB SRAM with ECC control
- 4KB I/D cache to enhance Flash code running performance
- Integrate clocking architecture with on-chip Fast IRC 64 MHz, LPO 32 kHz, SPLL 160 MHz and external crystal OSC8-40M and 32K
- Newly designed analog modules to provide precision mixed-signal capabilities, including 12-bit 1 Msps SAR ADC, high-speed comparator with integrated 8-bit DAC, etc.
- Power management and Regulators with support for multiple power modes including RUN, WAIT, STOP, and STANDBY
- IO to support wide range power supply and high ESD performance
- Wide operating voltage ranges with fully functional flash program with erase/read operations
- 64-pin LQFP, 100-pin LQFP, 144-pin LQFP, with up to 125 GPIO pins
- Ambient operating temperature ranges from -40 °C to 125 °C

The Z20K14xMC MCU portfolio is supported by a highly comprehensive set of development tools and software.

2.3 Feature summary

The Z20K14xMC product series has integrated a great number of features. These features include:

Table 2-1 Device feature summary

Feature	Z20K14xMC product series
Hardware characteristics	
Package	64-pin LQFP, 10*10 mm, 0.5 mm pitch 100-pin LQFP, 14*14 mm, 0.5 mm pitch 144-pin LQFP, 20*20 mm, 0.5 mm pitch
Voltage range	2.97 V to 5.5 V
Temperature range (TA)	-40 °C to 125 °C
Temperature range (TJ)	-40 °C to 135 °C
System	

Feature	Z20K14xMC product series
Central processing unit (CPU)	ARM Cortex-M4F with FPU
Maximum CPU frequency	Up to 160 MHz
Digital signal processor (DSP)	Yes
Floating point unit (FPU)	Yes
System memory protection unit (SMPU)	Yes
SysTick	Yes
I/D Cache	4 KB I/D cache
Nested vectored interrupt controller (NVIC)	Up to 256 vectored interrupts
	16 programmable interrupt priority levels
Wake-up interrupt controller (WIC)	Yes (Independent wakeup interrupt control)
Direct memory access (DMA)	16 channels
Direct memory access multiplexer (DMAMUX)	Yes
Non-maskable interrupt (NMI)	Yes
Internal watchdog (WDOG)	Yes
External watchdog timer (EWDG)	Yes, with external monitor pin.
Cyclic redundancy check (CRC)	16-bit or 32-bit CRC with programmable generator polynomial
Clock monitor unit (CMU)	Yes
Hardware encryption	Yes (AES128 and TRNG)
Debug	2-pin serial wire debug (SWD) IEEE1149.1 Joint Test Access Group (JTAG)
Boundary scan	Yes
Unique identification (ID) number	128-bit wide
Flash back door function	Yes (96bit backdoor key)
Memory	
Program flash memory	Up to 2 MB
Data flash memory	128 KB
Flash read buffer	Yes (128-bit)
Random-access memory (RAM)	Up to 256 KB
Clocks	
System clock controller (SCC)	Integrated OSC40M, FIRC64M, SPLL
External crystal oscillator or resonator	OSC32K: 30 - 40 kHz with low power OSC40M: high range 8-40 MHz
External square wave input clock	DC to 40 MHz
Internal clock references (IRC)	FIRC: fast internal IRC (64 MHz)
	LPO: low power oscillator (32 kHz)
System Phase-locked loop (SPLL)	Up to 1 GHz VCO, up to 160 M SPLL output
Human-Machine Interface (HMI)	
General-purpose input/output (GPIO)	Up to 125 GPIOs
	Pin interrupt/DMA request capability

Feature	Z20K14xMC product series
	Configurable digital glitch filter on all the PORTs Hysteresis, open-drain, pull up device and pull down device on all input pins
	Passive input filter on NMI_b and RESET_B input pins only
Analog	
Power management unit (PMU)	Low voltage warning
	Internal regulators offering various power modes
	32 kHz LPO clock
12-bit analog-to-digital converter (ADC)	2x
	Up to 1 Msps with 12-bit mode in continuous conversion
	Up to 20 single-ended external channels for each unit
	16 words FIFO
	Support automatic compare and hardware average
	Support command buffer function
High-speed comparator (CMP) with internal 8-bit DAC	1x
	Up to 13 external analog input channels with entire supply range
	Support programmable hysteresis
	Support high-speed and low-speed compare mode
Timers	
Trigger delay generator (TDG)	2x TDG
	6 pulse-out channels
	1 input trigger
16-bit timer (TIM)	Up to 4x TIM
	8 channels
	Input capture and output compare
	Support PWM output function
	Support 2 fault inputs
16-bit motor control pwm (MCPWM)	Up to 2x MCPWM
	8 channels
	Support PWM output function
	Support 4 fault inputs
Real time clock (RTC)	Yes
	Support external 32.768 kHz CRYSTAL and internal 32kHz LPO
32-bit system timer (STIM)	4-channel counter
	Support 4 trig inputs
Communication Interfaces	
Control area network (CAN)	Up to 8x CAN FD
	CAN0-CAN5 support up to 64 Message Buffers (MBs)

Feature	Z20K14xMC product series
	CAN6, CAN7 support up to 128 Message Buffers (MBs)
	Support CAN 2.0 and CAN FD protocol
Serial peripheral interface (SPI)	Up to 4x SPI
	Functional in RUN/WAIT modes
	DMA support, 4 words FIFO for both TX and RX
Inter-integrated circuit (I2C)	Up to 2x I2C
	Functional in RUN/WAIT modes
	DMA support, 4 bytes FIFO for both TX and RX
UART and LIN (UART)	Up to 6x UART
	Support 9-bit data transfer
	Functional in RUN/WAIT/STOP modes
	DMA support, 16 words FIFO for both TX and RX
	Support LIN
Inter-IC Sound (I2S)	Up to 2x I2S
	Transmitter and receiver based on I2S serial protocol
	Configurable number of stereo channels (up to 4)
	Half duplex communication

2.4 Block diagram

The following figure is the block diagram of the Z20K14xMC product series.

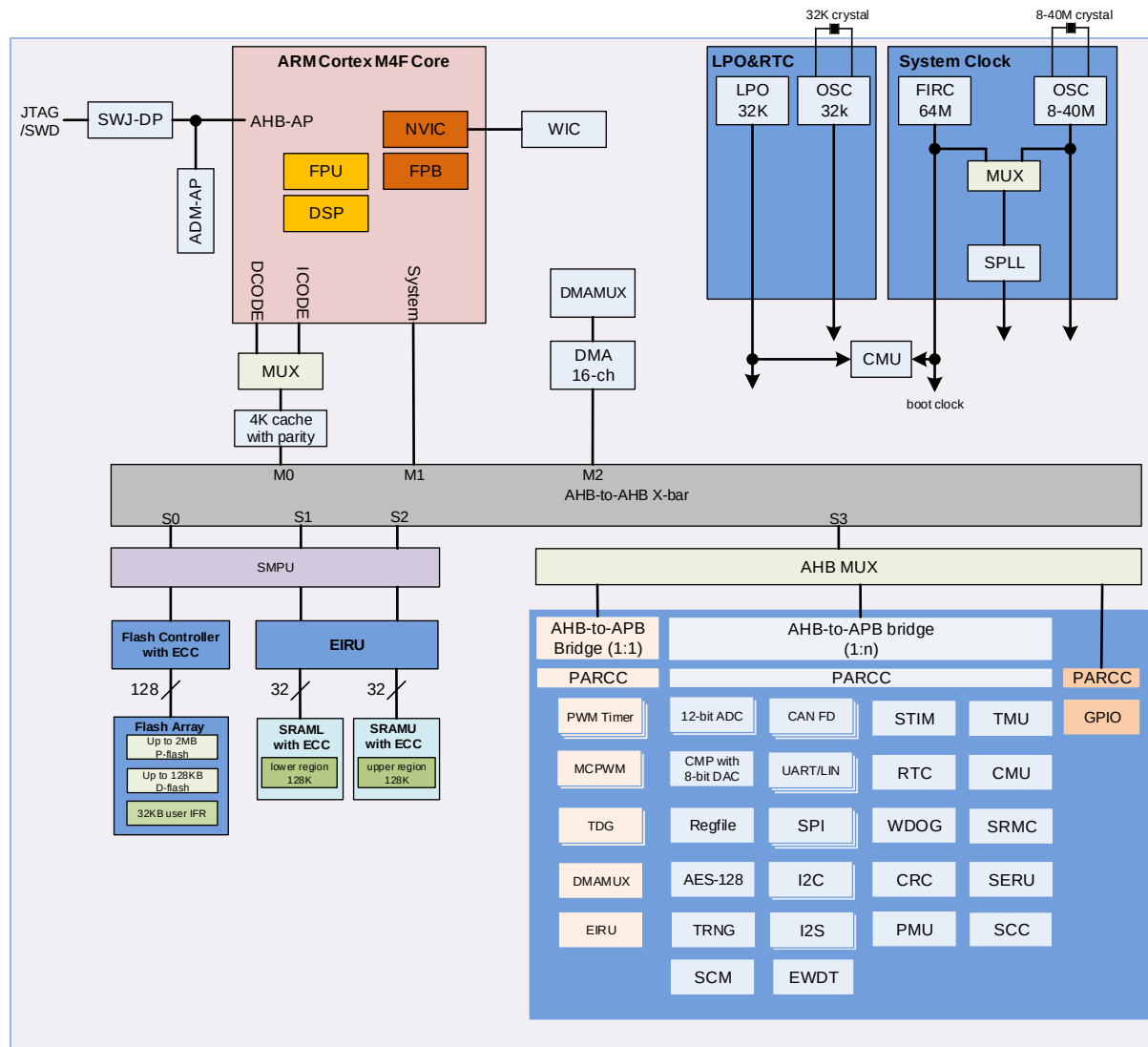


Figure 2-1 Z20K14xMC product block diagram

2.5 Feature comparison

The following figure summarizes memory, peripherals and packaging options for Z20K14xMC devices.

Table 2-2 Z20K14xMC product series comparison

		Z20K14xMC		
	Parameter	K144MC	K146MC	K148MC
	Core	ARM® Cortex®-M4F core		
	Frequency	120MHz	160MHz	160MHz
System	IEEE-754 FPU	●		
	CRC module	1x		
	ISO 26262	capable up to ASIL-B		
	Peripheral speed	up to 80 MHz		

		Z20K14xMC		
Parameter		K144MC	K146MC	K148MC
	Crossbar	●		
	DMA	●		
	External Watchdog Detect Timer (EWDt)	●		
	System memory protection unit (SMPU)	●		
	Clock Monitor Unit (CMU)	●		
	Watchdog	1x		
	System Control module (SCM)	●		
	System reset and mode controller (SRMC)	●		
	AHB to APB Peripheral bridge (AHB-APB)	●		
	System Error Report Unit (SERU)	●		
	ECC Injection and Report Unit (EIRU)	●		
	Power management unit (PMU)	●		
	Low power modes	●		
	AES128	1x		
	TRNG	1x		
	Number of I/Os	up to 83	up to 125	up to 125
	Single supply voltage	2.97 - 5.5 V		
	Ambient Operation Temperature (Ta)	-40°C to +125°C		
Memory	Program flash memory	512 KB	1 MB	2 MB
	Data flash memory	128 KB		
	Error Correcting Code (ECC)	●		
	System RAM	64 KB	128 KB	256 KB
	Regfile	32 x 32Bit		
	Cache	4 KB		
	EEPROM	Emulated by Data flash		
Timer	System Timer	1x (4ch 32bit)	1x (4ch 32bit)	1x (4ch 32bit)

		Z20K14xMC		
	Parameter	K144MC	K146MC	K148MC
	Timer	4x (8ch 16bit)	4x (8ch 16bit)	4x (8ch 16bit)
	Motor Control PWM (MCPWM)	2x (8ch 16bit)	2x (8ch 16bit)	2x (8ch 16bit)
	Real Time Counter (RTC)	1x	1x	1x
	Trigger delay generator (TDG)	2x	2x	2x
Analog	Trigger mux (TMU)	●		
	12-bit SAR ADC (1 Msp/s each)	2x (16ch)	2x (20ch)	2x (20ch)
	Comparator with 8-bit DAC	1x (13ch)	1x (13ch)	1x (13ch)
Communication	Inter IC Sound (I2S)	1x	1x	2x
	UART and LIN (UART)	6x	6x	6x
	Serial peripheral interface (SPI)	4x	4x	4x
	Inter-integrated circuit (I2C)	2x	2x	2x
	Control area network (CAN)	4x(4x with FD)	6x(6x with FD)	8x(8x with FD)
IDE	Debug & trace	SWD, JTAG	SWD, JTAG	SWD, JTAG
	Ecosystem (IDE, compiler, debugger)	IAR, GHS, Arm®, Lauterbach		
Other	Packages	64-pin LQFP 100-pin LQFP	64-pin LQFP 100-pin LQFP 144-pin LQFP	64-pin LQFP 100-pin LQFP 144-pin LQFP

LEGEND:

○ Not implemented

● Available on the device

2.6 Applications

The Z20K14xMC product series suit well with the general-purpose automotive applications, which include but not limited to the following:

- Exterior and interior lighting
- HVAC

- Door/window/wiper/seat controller
- BLDC/PMSM motor control
- Park assistant
- E-shifter
- TPMS
- Real time control in infotainment system
- Battery management system
- Human machine interface, such as touch sensing controls
- Secured vehicle data transfer
- Safety controller
- Over the air update

Besides the automotive applications, the Z20K14xMC product series can also be used in the automation, communication, transportation, and medical applications that require high-level quality, reliability, and safety.

2.7 Module functional categories

These device modules are grouped into several functional categories. The following table provides more details of the modules assigned to each category.

Table 2-3 Module functional categories

Module category	Description
ARM Cortex-M4F core	32-bit MCU core from ARM Cortex-M class based on Armv7 architecture Add DSP instructions and single-precision floating point unit
System	System Control module (SCM) Power management unit (PMU) System reset and mode controller (SRMC) Multiple power modes available based on RUN, WAIT, STOP and STANDBY modes Crossbar switch (X-BAR) System memory protection unit (SMPU) AHB to APB Peripheral bridge (AHB-APB) Direct memory access (DMA) controller Watchdog (WDOG) and External watchdog timer (EWDT) ECC Injection and Report Unit (EIRU) System Error Report Unit (SERU) Clock Monitor Unit (CMU)
Memories	Internal memories include: Program flash memory (Flash0) Data flash memory (Flash1) SRAM
Clocks	Multiple clock generation options available from internally- and externally-generated clocks System oscillator to provide clock source for the MCU 32K oscillator to provide clock source for RTC
Security and safety	Cyclic Redundancy Check (CRC) module for error detection Error-correcting code (ECC) on Flash and SRAM memories 128-bit unique identification (ID) number System memory protection unit (SMPU) module Hardware security engine (AES128 and TRNG)
Analog	High speed analog-to-digital converter (ADC) Comparator (CMP)

Module category	Description
Timers	Trigger delay generator (TDG) Timer (TIM) Motor control PWM timer (MCPWM) System timer (STIM) Independent real time clock (RTC)
Communications	CAN (CAN FD) Serial peripheral interface (SPI) Inter-integrated circuit (I2C) Universal Asynchronous Receiver/Transmitter (UART with LIN) Inter-IC Sound (I2S)
Human-Machine Interfaces (HMI)	General purpose input/output controller (GPIO)

2.7.1 ARM Cortex-M4F core modules

The following table details the core modules that are available on this device.

Table 2-4 Core modules

Module	Description
ARM Cortex-M4F	Cost sensitive, deterministic, interrupt driven environments Armv7 Architecture and Thumb®-2 ISA and is upward compatible with the Cortex M3 and Cortex M0 architectures An Armv7 Thumb-2 DSP (ported from the Armv7-A/R profile architectures) provides 32-bit instructions with SIMD (single instruction multiple data) DSP style multiply-accumulates and saturating arithmetic
NVIC	A relocatable vector table supporting many external interrupts, a single non-maskable interrupt (NMI), and priority levels Replaces shadow registers with equivalent system and simplified programmability Includes address of the function to execute for a particular handler The address is fetched via the instruction port allowing parallel register stacking and look-up The first sixteen entries are allocated to Arm internal sources, with the others mapping to MCU-defined interrupts
WIC	Detects wake-up events in STOP modes and signal to clock control logic to resume system clocking After clock has restarted, NVIC observes the pending interrupt and performs the normal interrupt or event processing
Debug interfaces	Based on the ARM CoreSight™ architecture Serial wire IEEE 1149.1 JTAG debug Port (SWJ-DP), with 2 pin serial wire debug (SWD) for external debugger

2.7.2 System modules

The following table details the system modules that are available on this device.

Table 2-5 System modules

Module	Description
System control module (SCM)	Provides miscellaneous device configuration and status registers.
PORT	Supports port control, digital filtering, and pin interrupt functions.

Module	Description
GPIO	Communicates with the processor core via an AHB interface for maximum pin performance. Supports 8-bit, 16-bit or 32-bit accesses.
Peripheral bridge	Converts the crossbar switch interface to an interface that allows access to a majority of peripherals on the device.
Direct memory access multiplexer (DMAMUX)	Selects from many DMA requests down to a smaller number for the DMA controller.
Direct memory access controller (DMA)	Provides programmable channels with transfer control descriptors for data movement via dual-address transfers for 8-bit, 16-bit, and 32-bit data values.
Trigger mux unit (TMU)	Allows software to configure the trigger sources for various peripherals.
External watchdog timer (EWDG)	A redundant mechanism to the software watchdog module that monitors both internal and external system operation for fail conditions.
Internal watchdog (WDG)	Monitors internal system operation and forces a reset in case of failure. Can run from an independent 32 kHz low power oscillator with a programmable refresh window to detect deviations in program flow or system frequency.
CRC	Uses 16/32-bit shift register. Error detection for all single, double, odd, and most multi-bit errors, programmable initial seed value, and optional feature to transpose input data and CRC result via transpose register.

2.7.3 Memories and memory interfaces

The following table details memories and memory interfaces that are available on this device.

Table 2-6 Memories and memory interfaces

Module	Description
Flash memory	Program flash memory — non-volatile flash memory that can execute program code. Data flash memory — non-volatile flash memory that can store user data.
Flash memory controller	Manages the interface between the device and the on-chip flash memory.
SRAM	Internal system RAM.
REGFILE	Stores the pre-written data in STANDBY mode.

2.7.4 Power management

The following table details modules that are available on this device for power management and control of system power modes.

Table 2-7 Power management modules

Module	Description
PMU	The PMU contains the internal voltage regulator, Bandgap, power-on reset (POR) and the low voltage detect (LVD) system.
SRMC	The System Reset and Mode Controller (SRMC) is responsible for reset control

Module	Description
	and sequencing the system into and out of all low-power modes.

2.7.5 Clocking

The following table details clock modules that are available on this device.

Table 2-8 Clock modules

Module	Description
System clock controller (SCC)	The SCC provides several clock sources for the MCU that include: - Fast internal reference clock (FIRC) — An internally-generated 64 MHz clock - System oscillator (OSC) — The system oscillator, in conjunction with an external crystal or resonator, generates a reference clock for the MCU.
Low power oscillator (LPO)	An internally-generated low power clock with typical frequency of 32 kHz, which can be used as clock source for modules operational in low power modes.
32K oscillator (OSC32K)	The 32K oscillator, in conjunction with an external crystal or resonator, generates a reference clock for RTC.
Peripheral access & reset & clock control (PARCC)	PARCC controls the clock selection for most modules.

2.7.6 Analog modules

The following table details the analog modules that are available on this device:

Table 2-9 Analog Modules

Module	Description
12-bit Analog-to-digital converters (ADC)	12-bit SAR ADC
Analog comparators (CMP)	Compares two analog input voltages across the full range of the supply voltage.
8-bit digital-to-analog converters (DAC) within CMP	256-tap resistor ladder network which provides a selectable voltage reference for applications where a voltage reference is needed.

2.7.7 Timer modules

The following table details the timer modules that are available on this device.

Table 2-10 Timer modules

Module	Description
Trigger delay generator (TDG)	16-bit resolution 3-bit prescaler - Positive transition of trigger event signal initiates the counter - Supports multiple triggered delay output signals, each with an independently-controlled delay from the trigger event - Continuous-pulse output or single-shot mode supported, each output is independently enabled, with possible trigger events - Supports DMA
Timer (TIM)	Selectable TIM source clock, programmable prescaler.

Module	Description
	• 16-bit counter supporting free-running or initial/final value, and counting is up or up-down. • Input capture, output compare, and edge-aligned and center-aligned PWM modes. • Operation of TIM channels as pairs with equal outputs, pairs with complementary outputs, or independent channels with independent outputs. • Deadtime insertion is available for each complementary pair. • Generation of hardware triggers. • Software control of PWM outputs. • Up to 2 fault-inputs for global fault control. • Configurable channel polarity. • Programmable interrupt on input capture, reference compare, overflowed counter, or detected fault condition and reload opportunity. • DMA support.
System timer (STIM)	4 independent channels to service: • Each channel implements a 32-bit counter. • Four clock sources can be selected for each channel, the clock sources are specified by SoC integration guide. • Four input pulses can be selected for each channel, three pulses come from pad and one pulse comes from TMU. • Each channel counter can work as free counting mode or pulse counting mode. • In free counting mode, counter increments on the rising edge of divided clock source. • In pulse counter mode, counter increments on the rising or falling edge of input pulse. A filter can be enabled to filter the glitch of input pulse. • Each channel can generate a match trigger when the counter value matches. • Each channel can generate an interrupt and DMA request. • Some fields are write protected, and they can only be written when channel is disabled.
Real-time clock (RTC)	• A 32-bit seconds counter to generate alarm and second interrupt. • 16-bit prescaler with compensation that can correct errors between 0.12 ppm and 3906 ppm. • An optional counter clock source from LPO32K or OSC32K. • Configurable crystal square wave output to dedicated pin.
Motor control PWM (MCPWM)	• Support multiple counter clock sources, programmable prescaler. • Integrated 4 16-bit counters. Counting is up or up-down. • Support 7 PWM output modes. • Operation of TIM channels as pairs with equal outputs, pairs with complementary outputs, or independent channels with independent outputs. • Deadtime insertion is available for each complementary pair. • Software control of PWM outputs. • Up to 4 fault inputs for global fault control. • Configurable channel polarity. • Programmable interrupt on reference compare,

Module	Description
	overflowed counter, or detected fault condition and reload opportunity. DMA support.

2.7.8 Communication interfaces

The following table details the communication interfaces that are available on this device:

Table 2-11 Communication modules

Module	Description
Serial peripheral interface (SPI)	Synchronous serial bus for communication to an external device.
Inter-integrated circuit (I2C)	- Allows communication between a number of devices. - Supports an efficient interface to an I2C bus as a master and/or a slave. I2C implements logic support for standard mode, fast mode, fast mode plus, high-speed mode of operation.
Universal asynchronous receiver/transmitter (UART)	- Asynchronous serial bus communication interface. - Supports LIN master and slave operation. - Optionally remains functional in low power modes.
CAN	A communication controller implementing the CAN protocol according to the ISO 11898-1 standard and CAN 2.0 B protocol specifications.
Inter-IC sound (I2S)	- I2S transmitter/receiver based on the Philips I2S serial protocol. - Supports 4 stereo data channels for both transmitter and receivers. - I2S module only supports half duplex communication. - Supports master or slave option. - Asynchronous APB clock and I2S function clock. - Configurable function clock divider to achieve the expected sample rate. - Configurable audio data resolutions of 12, 16, 20, 24, and 32 bits. - Transmitter and Receiver FIFO depth is 16 words, and word size is determined via data resolutions configuration. - Configurable FIFO threshold level. - Supports cyclic transmission during all enabled transmission channels.

Chapter 3

Memory Map

3.1 Introduction

Memories and memory-mapped peripherals of the chip are in one 32-bit contiguous memory space. This chapter details the memory and peripheral locations of the Z20K14xMC system.

3.2 SRAM and Flash memory map

Memory	Z20K14xMC								
	K144MC			K146MC			K148MC		
	Size	Start Address	End Address	Size	Start Address	End Address	Size	Start Address	End Address
	(KB)	(hex)	(hex)	(KB)	(hex)	(hex)	(KB)	(hex)	(hex)
Program flash memory	512	00000000	0007FFFF	1024	00000000	000FFFFFFF	2048	00000000	001FFFFFFF
Data flash memory	128	01000000	0101FFFF	128	01000000	0101FFFF	128	01000000	0101FFFF
Flash IFR	32	02000000	02007FFF	32	02000000	02007FFF	32	02000000	02007FFF
SRAM_L	0	-	-	0	-	-	128	1FFE0000	1FFFFFFF
SRAM_U	64	20000000	2000FFFF	128	20000000	2001FFFF	128	20000000	2001FFFF

3.3 Peripheral bridge memory map

The peripheral memory map is accessible via a crossbar slave port. There are two regions associated with peripheral space.

Table 3-1 Regions associated with peripheral space

Address space	Region description
0x4000_0000–0x4007_FFFF	A 512 KB region is partitioned as 128 spaces, each 4 KB in size and reserved for peripheral devices. The region generates unique module enables for all 128 spaces.
0x400F_F000–0x400F_FFFF	A 4 KB region is for accessing the GPIO module. This block is connected to the AMBA bus via the port splitter and provides direct master access. The GPIO is implemented only in the upper space of this region. The 4KB region begins at the address 0x400F_F000.

NOTE

Modules that are disabled via their clock gate control bits in the PARCC registers disable the associated 4K slots. Access to any address within an unimplemented or disabled peripheral bridge slot results in a transfer error termination.

Table 3-2 Peripheral space assignment

Peripheral Instance	Z20K14xMC								
	K144MC			K146MC			K148MC		
	Size	Start Address	End Address	Size	Start Address	End Address	Size	Start Address	End Address
	(KB)	(hex)	(hex)	(KB)	(hex)	(hex)	(KB)	(hex)	(hex)
DMA0	4	40008000	40008FFF	4	40008000	40008FFF	4	40008000	40008FFF
DMAMUX0	4	40009000	40009FFF	4	40009000	40009FFF	4	40009000	40009FFF
SMPU	4	4000A000	4000AFFF	4	4000A000	4000AFFF	4	4000A000	4000AFFF
EIRU	4	4000B000	4000BFFF	4	4000B000	4000BFFF	4	4000B000	4000BFFF
TIM0	4	4000C000	4000CFFF	4	4000C000	4000CFFF	4	4000C000	4000CFFF
TIM1	4	4000D000	4000DFFF	4	4000D000	4000DFFF	4	4000D000	4000DFFF
MCPWM0	4	4000E000	4000EFFF	4	4000E000	4000EFFF	4	4000E000	4000EFFF
TDG0	4	4000F000	4000FFFF	4	4000F000	4000FFFF	4	4000F000	4000FFFF
TIM2	4	40018000	40018FFF	4	40018000	40018FFF	4	40018000	40018FFF
TIM3	4	40019000	40019FFF	4	40019000	40019FFF	4	40019000	40019FFF
MCPWM1	4	4001A000	4001AFFF	4	4001A000	4001AFFF	4	4001A000	4001AFFF
TDG1	4	4001B000	4001BFFF	4	4001B000	4001BFFF	4	4001B000	4001BFFF
Flash	4	40020000	40020FFF	4	40020000	40020FFF	4	40020000	40020FFF
CAN 0	4	40024000	40024FFF	4	40024000	40024FFF	4	40024000	40024FFF
CAN1	4	40025000	40025FFF	4	40025000	40025FFF	4	40025000	40025FFF
ADC1	4	40027000	40027FFF	4	40027000	40027FFF	4	40027000	40027FFF
SPI 0	4	4002C000	4002CFFF	4	4002C000	4002CFFF	4	4002C000	4002CFFF
SPI 1	4	4002D000	4002DFFF	4	4002D000	4002DFFF	4	4002D000	4002DFFF
SPI 2	4	40030000	40030FFF	4	40030000	40030FFF	4	40030000	40030FFF
SPI 3	4	40031000	40031FFF	4	40031000	40031FFF	4	40031000	40031FFF
CRC	4	40032000	40032FFF	4	40032000	40032FFF	4	40032000	40032FFF
AES	4	40033000	40033FFF	4	40033000	40033FFF	4	40033000	40033FFF
TRNG	4	40034000	40034FFF	4	40034000	40034FFF	4	40034000	40034FFF
CAN2	4	40038000	40038FFF	4	40038000	40038FFF	4	40038000	40038FFF
CAN3	4	40039000	40039FFF	4	40039000	40039FFF	4	40039000	40039FFF

Peripheral Instance	Z20K14xMC								
	K144MC			K146MC			K148MC		
	Size	Start Address	End Address	Size	Start Address	End Address	Size	Start Address	End Address
	(KB)	(hex)	(hex)	(KB)	(hex)	(hex)	(KB)	(hex)	(hex)
ADC0	4	4003B000	4003BFFF	4	4003B000	4003BFFF	4	4003B000	4003BFFF
CMU0	4	4003E000	4003EFFF	4	4003E000	4003EFFF	4	4003E000	4003EFFF
STIM	4	40040000	40040FFF	4	40040000	40040FFF	4	40040000	40040FFF
CMU1	4	40042000	40042FFF	4	40042000	40042FFF	4	40042000	40042FFF
CMU2	4	40043000	40043FFF	4	40043000	40043FFF	4	40043000	40043FFF
CAN4	Reserved			4	40044000	40044FFF	4	40044000	40044FFF
CAN5	Reserved			4	40045000	40045FFF	4	40045000	40045FFF
SCM	4	40048000	40048FFF	4	40048000	40048FFF	4	40048000	40048FFF
Port A	4	40049000	40049FFF	4	40049000	40049FFF	4	40049000	40049FFF
Port B	4	4004A000	4004AFFF	4	4004A000	4004AFFF	4	4004A000	4004AFFF
Port C	4	4004B000	4004BFFF	4	4004B000	4004BFFF	4	4004B000	4004BFFF
Port D	4	40050000	40050FFF	4	40050000	40050FFF	4	40050000	40050FFF
Port E	4	40051000	40051FFF	4	40051000	40051FFF	4	40051000	40051FFF
WDOG	4	40052000	40052FFF	4	40052000	40052FFF	4	40052000	40052FFF
I2S0	4	40054000	40054FFF	4	40054000	40054FFF	4	40054000	40054FFF
I2S1	Reserved			Reserved			4	40055000	40055FFF
CAN6	Reserved			Reserved			4	40056000	40056FFF
CAN7	Reserved			Reserved			4	40057000	40057FFF
I2C1	4	40058000	40058FFF	4	40058000	40058FFF	4	40058000	40058FFF
EWDT	4	40061000	40061FFF	4	40061000	40061FFF	4	40061000	40061FFF
TMU	4	40063000	40063FFF	4	40063000	40063FFF	4	40063000	40063FFF
SCC	4	40064000	40064FFF	4	40064000	40064FFF	4	40064000	40064FFF
PARCC	4	40065000	40065FFF	4	40065000	40065FFF	4	40065000	40065FFF
I2C0	4	40066000	40066FFF	4	40066000	40066FFF	4	40066000	40066FFF
UART0	4	4006A000	4006AFFF	4	4006A000	4006AFFF	4	4006A000	4006AFFF
UART1	4	4006B000	4006BFFF	4	4006B000	4006BFFF	4	4006B000	4006BFFF

Z20K14xMC									
Peripheral Instance	K144MC			K146MC			K148MC		
	Size	Start Address	End Address	Size	Start Address	End Address	Size	Start Address	End Address
	(KB)	(hex)	(hex)	(KB)	(hex)	(hex)	(KB)	(hex)	(hex)
UART2	4	4006C000	4006CFFF	4	4006C000	4006CFFF	4	4006C000	4006CFFF
UART3	4	40070000	40070FFF	4	40070000	40070FFF	4	40070000	40070FFF
UART4	4	40071000	40071FFF	4	40071000	40071FFF	4	40071000	40071FFF
UART5	4	40072000	40072FFF	4	40072000	40072FFF	4	40072000	40072FFF
CMP0	4	40073000	40073FFF	4	40073000	40073FFF	4	40073000	40073FFF
RTC	4	40079000	40079FFF	4	40079000	40079FFF	4	40079000	40079FFF
REGFILE	4	4007C000	4007CFFF	4	4007C000	4007CFFF	4	4007C000	4007CFFF
PMU	4	4007D000	4007DFFF	4	4007D000	4007DFFF	4	4007D000	4007DFFF
SRMC	4	4007E000	4007EFFF	4	4007E000	4007EFFF	4	4007E000	4007EFFF
SERU	4	4007F000	4007FFFF	4	4007F000	4007FFFF	4	4007F000	4007FFFF
GPIO	4	400FF000	400FFFFF	4	400FF000	400FFFFF	4	400FF000	400FFFFF

3.3.1 Read-after-write sequence and required serialization of memory options

In many occasions, a write to a peripheral must be completed fully before a subsequent action takes place. Some examples of these occasions are as follows:

- Exiting an interrupt service routine (ISR)
- Changing a mode
- Configuring a function

In these occasions, the application software must perform a read-after-write sequence to guarantee the required serialization of the memory operations, as shown below:

Step 1: Writing the peripheral register

Step 2: Reading the written peripheral register to verify the write

Step 3: Continuing with subsequent operations

3.4 Private peripheral bus (PPB) memory map

The PPB is part of the defined ARM bus architecture, which provides access to select processor-local modules. These resources can be only accessed from the core. Other system masters do not have access to them.

Table 3-3 PPB memory map for CM4F

System 32-bit address range	Resource
0xE000_0000–0xE000_0FFF	Reserved
0xE000_1000–0xE000_1FFF	Reserved
0xE000_2000–0xE000_2FFF	Flash Patch and Breakpoint (FPB)

System 32-bit address range	Resource
0xE000_3000–0xE000_DFFF	Reserved
0xE000_E000–0xE000_EFFF	System Control Space (SCS) (for NVIC and FPU)
0xE000_F000–0xE003_FFFF	Reserved
0xE004_0000–0xE004_0FFF	Reserved
0xE004_1000–0xE004_1FFF	Reserved
0xE004_2000–0xE004_2FFF	Reserved
0xE004_3000–0xE004_3FFF	Reserved
0xE004_4000–0xE007_FFFF	Reserved
0xE008_0000–0xE008_0FFF	Reserved
0xE008_1000–0xE008_1FFF	Reserved
0xE008_2000–0xE008_2FFF	Reserved
0xE008_3000–0xE00F_EFFF	Reserved
0xE00F_F000–0xE00F_FFFF	Arm Core ROM Table

Chapter 4

Signal Multiplexing and Pin Assignment

4.1 Introduction

The signal multiplexing unit, which consists of control signals from GPIO, PORT, and pad interface, provides multiple functions for sharing the single pad. Every one of the sub-units under the signal multiplexing unit handles the signal multiplexing of one pad.

The Port Control block controls the signal presented on the external pin as well as the specific pad setting, such as the pull-up setting of the module specific pad. See PORTx_PCRn for the detailed description of control signals.

4.2 Functional description

The following figure shows the signal multiplexing architectural implementation.

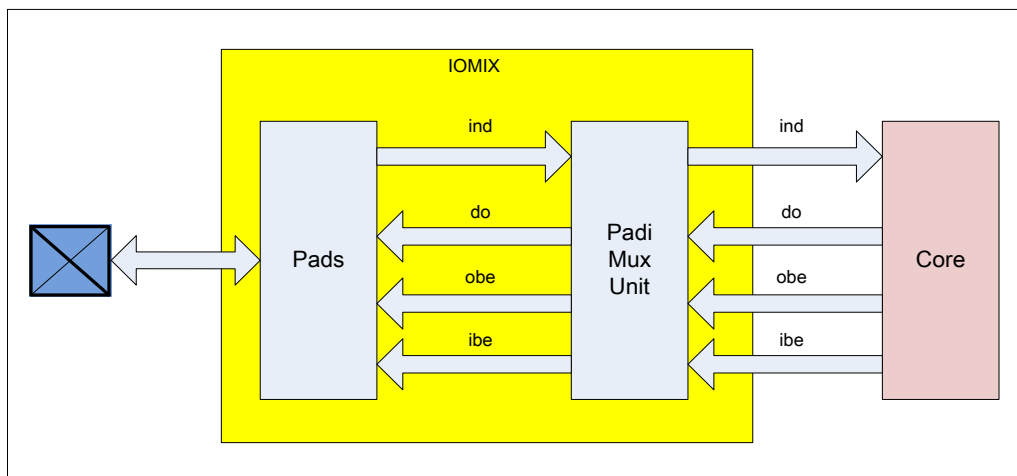


Figure 4-1 Signal multiplexing

4.3 Pad description

The following figure shows the basic representation of a GPIO pad.

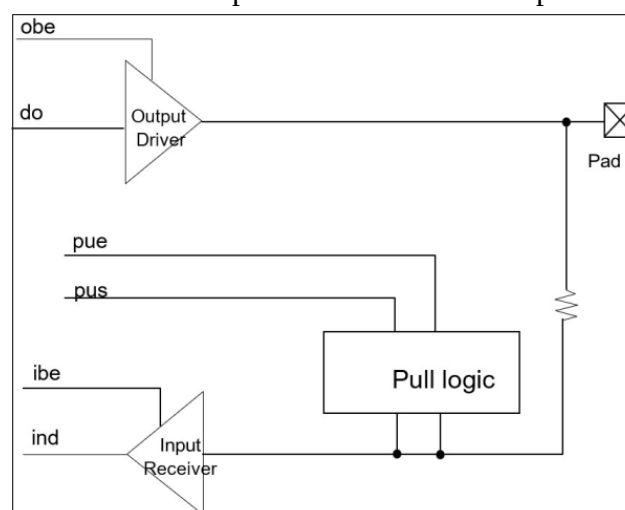


Figure 4-2 GPIO pad representation

Table 4-1 Pad signal description

Signal name	Direction	Description
pad	I/O	I/O to external world
do	I	Data comes out from the core into the pad
obe	I	Enable output driver
pue	I	0b: Disable internal pullup or pulldown the resistor 1b: Enable internal pullup or pulldown the resistor
pus	I	0b: Enable internal pulldown resistor if pue is set 1b: Enable internal pullup resistor if pue is set
ibe	I	Enable input receiver
ind	O	Data comes out of the pad into the core

Table 4-2 Truth Table

obe	do	PAD	Description
0	X	Z	Output buffer is disabled. Pad hi-Z (If not configured as input).
1	0/1	0/1	Output buffer is enabled. Pad = do.

obe	pue	pus	PAD	Description
1	X	X	-	Weak pull is disabled.
0	0	X	-	Weak pull is disabled. Pad retains the previous state.
0	1	0	0	Weak pull down is enabled.
0	1	1	1	Weak pull up is enabled.

ibe	pad	ind	Description
0	X	0	Input buffer is disabled. ind gets low
1	0/1	0/1	Input buffer is enabled. ind = pad

The device supports open drain on all the pins.

4.4 Signal multiplexing sheet

Detailed information can be found in PinMux attached in this Reference Manual.

Chapter 5

Cortex-M4F Overview

5.1 ARM Cortex-M4F core introductions

The Arm® Cortex®-M4F is a member of the Cortex M Series of processors focused on very cost sensitive and interrupt driven environments. The Cortex M4F processor is based on the Armv7 Architecture and Thumb®-2 ISA and is compatible with the Cortex M3 and Cortex M0 architectures. Cortex M4F improvements include an Armv7 Thumb-2 DSP (ported from the Armv7-A/R profile architectures) providing 32-bit instructions with SIMD (single instruction multiple data) DSP style multiply-accumulates and saturating arithmetic. It includes floating-point arithmetic functionality.

5.1.1 Buses, interconnects, and interfaces

The ARM Cortex-M4F core has four bus interfaces:

- ICODE bus: code access from address 0x00000000-0x1FFFFFFF.
- DCODE bus: data access from address 0x00000000-0x1FFFFFFF.
- System bus: system bus connected to a separate master port on the crossbar.
- Private peripheral bus (PPB): PPB bus is used to access Arm module such as NVIC, FPB and ROM table.

5.1.2 System tick timer

The CLKSOURCE field in SysTick Control and Status register can select the core clock (when CLKSOURCE = 1) as clock source only.

The TENMS field in the SysTick Calibration Value Register is always 0, because the timing reference is a variable frequency.

5.1.3 Debug facilities

This device supports standard ARM 2-pin SWD debug port and standard 4-pin JTAG interface.

5.1.4 Caches

The device includes one 4KB code cache to improve the access performance of memory access. Address 0x00000000-0x0FFFFFFF can be cached. ICODE bus and DCODE bus can access the Caches.

5.1.5 Core privilege levels

The core on this device is implemented with both privileged (Supervisor) and unprivileged (User) levels.

5.2 Nested vectored interrupt controller (NVIC)

5.2.1 Interrupt priority levels

This device supports sixteen priority levels for interrupts. Therefore, in the NVIC, each source in the IPR registers contains 4 bits.

5.2.2 Non-maskable interrupt

The non-maskable interrupt request to the NVIC is controlled by the external NMI signal. The pin on which the NMI signal is multiplexed must be configured for the NMI function to generate the non-maskable interrupt request.

The user does not need to clear the interrupt flag when NMI happens because there is no non-maskable interrupt status flag. Before exiting the NMI handler, the NMI signal must be deasserted, otherwise it will trigger the NMI again.

5.2.3 Interrupt vector

Table 5-1 lists the interrupt vector on this device.

Table 5-1 Interrupt vector assignment

Address	Vector	IRQ	Source module	Source description	Type
0	0	-	ARM core	Initial Stack Pointer	
4	1	-	ARM core	reset	
8	2	-	ARM core	Non-maskable Interrupt (NMI)	Level
C	3	-	ARM core	Hard fault	Pulse
10	4	-	ARM core	MemManage Fault	Pulse
14	5	-	ARM core	Bus Fault	Pulse
18	6	-	ARM core	Usage Fault	Pulse
1C	7	-			
20	8	-			
24	9	-			
28	10	-			
2C	11	-	ARM core	Supervisor call (SVCall)	Pulse
30	12	-	ARM core	Debug Monitor	Pulse
34	13	-			
38	14	-	ARM core	Pendable request for system service (PendableSrvReq)	Pulse
3C	15	-	ARM core	System tick timer (SysTick)	Pulse
40	16	0	DMA	DMA channel 0 transfer complete	Level
44	17	1	DMA	DMA channel 1 transfer complete	Level
48	18	2	DMA	DMA channel 2 transfer complete	Level
4C	19	3	DMA	DMA channel 3 transfer complete	Level
50	20	4	DMA	DMA channel 4 transfer complete	Level
54	21	5	DMA	DMA channel 5 transfer complete	Level
58	22	6	DMA	DMA channel 6 transfer complete	Level
5C	23	7	DMA	DMA channel 7 transfer complete	Level
60	24	8	DMA	DMA channel 8 transfer complete	Level
64	25	9	DMA	DMA channel 9 transfer complete	Level
68	26	10	DMA	DMA channel 10 transfer complete	Level
6C	27	11	DMA	DMA channel 11 transfer complete	Level
70	28	12	DMA	DMA channel 12 transfer complete	Level
74	29	13	DMA	DMA channel 13 transfer complete	Level

Address	Vector	IRQ	Source module	Source description	Type
78	30	14	DMA	DMA channel 14 transfer complete	Level
7C	31	15	DMA	DMA channel 15 transfer complete	Level
80	32	16	DMA	DMA channels 0-15 error interrupt	Level
84	33	17	Flash	Command complete	Level
88	34	18	Flash	ECC error(double bit error and single bit error)	Level
8C	35	19	PMU	LVD, LVW	Level
90	36	20	WDOG	Watch dog	Level
94	37	21	EWDT	External watchdog monitor	Level
98	38	22	SRMC	System reset and mode control	Level
9C	39	23	I2C0	I2C0 Interrupt	Level
A0	40	24	I2C1	I2C1 Interrupt	Level
A4	41	25	SPI0	SPI0 txe Interrupt	Level
A8	42	26	SPI0	SPI0 txo Interrupt	Level
AC	43	27	SPI0	SPI0 rxf Interrupt	Level
B0	44	28	SPI0	SPI0 rxo Interrupt	Level
B4	45	29	SPI0	SPI0 rxu Interrupt	Level
B8	46	30	Reserved		
BC	47	31	SPI1	SPI1 txe Interrupt	Level
C0	48	32	SPI1	SPI1 txo Interrupt	Level
C4	49	33	SPI1	SPI1 rxf Interrupt	Level
C8	50	34	SPI1	SPI1 rxo Interrupt	Level
CC	51	35	SPI1	SPI1 rxu Interrupt	Level
D0	52	36	Reserved		
D4	53	37	SPI2	SPI2 txe Interrupt	Level
D8	54	38	SPI2	SPI2 txo Interrupt	Level
DC	55	39	SPI2	SPI2 rxf Interrupt	Level
E0	56	40	SPI2	SPI2 rxo Interrupt	Level
E4	57	41	SPI2	SPI2 rxu Interrupt	Level
E8	58	42	Reserved		
EC	59	43	SPI3	SPI3 txe Interrupt	Level
F0	60	44	SPI3	SPI3 txo Interrupt	Level
F4	61	45	SPI3	SPI3 rxf Interrupt	Level
F8	62	46	SPI3	SPI3 rxo Interrupt	Level
FC	63	47	SPI3	SPI3 rxu Interrupt	Level
100	64	48	Reserved		
104	65	49	UART0	UART0 Interrupt	Level
108	66	50	UART1	UART1 Interrupt	Level
10C	67	51	UART2	UART2 Interrupt	Level
110	68	52	UART3	UART3 Interrupt	Level
114	69	53	UART4	UART4 Interrupt	Level
118	70	54	UART5	UART5 Interrupt	Level

Address	Vector	IRQ	Source module	Source description	Type
11C	71	55	CAN0	CAN 0 bus off	Level
120	72	56	CAN0	CAN 0 tx warning	Level
124	73	57	CAN0	CAN 0 rx warning	Level
128	74	58	CAN0	CAN 0 error Interrupt	Level
12C	75	59	CAN0	CAN 0 FD error Interrupt	Level
130	76	60	CAN0	CAN 0 pretended network wakeup Interrupt	Level
134	77	61	CAN0	CAN 0 self-wakeup Interrupt	Level
138	78	62	CAN0	CAN 0 ECC interrupt	Level
13C	79	63	CAN0	CAN 0 Message buffer (0-15)	Level
140	80	64	CAN0	CAN 0 Message buffer (16-31)	Level
144	81	65	CAN0	CAN 0 Message buffer (32-47)	Level
148	82	66	CAN0	CAN 0 Message buffer (48-63)	Level
14C	83	67	CAN1	CAN 1 bus off	Level
150	84	68	CAN1	CAN 1 tx warning	Level
154	85	69	CAN1	CAN 1 rx warning	Level
158	86	70	CAN1	CAN 1 error Interrupt	Level
15C	87	71	CAN1	CAN 1 FD error Interrupt	Level
160	88	72	CAN1	CAN 1 pretended network wakeup Interrupt	Level
164	89	73	CAN1	CAN 1 self-wakeup Interrupt	Level
168	90	74	CAN1	CAN 1 ECC interrupt	Level
16C	91	75	CAN1	CAN 1 Message buffer (0-15)	Level
170	92	76	CAN1	CAN 1 Message buffer (16-31)	Level
174	93	77	CAN1	CAN 1 Message buffer (32-47)	Level
178	94	78	CAN1	CAN 1 Message buffer (48-63)	Level
17C	95	79	CAN2	CAN 2 bus off	Level
180	96	80	CAN2	CAN 2 tx warning	Level
184	97	81	CAN2	CAN 2 rx warning	Level
188	98	82	CAN2	CAN 2 error Interrupt	Level
18C	99	83	CAN2	CAN 2 FD error Interrupt	Level
190	100	84	CAN2	CAN 2 pretended network wakeup Interrupt	Level
194	101	85	CAN2	CAN 2 self-wakeup Interrupt	Level
198	102	86	CAN2	CAN 2 ECC interrupt	Level
19C	103	87	CAN2	CAN 2 Message buffer (0-15)	Level
1A0	104	88	CAN2	CAN 2 Message buffer (16-31)	Level
1A4	105	89	CAN2	CAN 2 Message buffer (32-47)	Level
1A8	106	90	CAN2	CAN 2 Message buffer (48-63)	Level
1AC	107	91	CAN3	CAN 3 bus off	Level
1B0	108	92	CAN3	CAN 3 tx warning	Level
1B4	109	93	CAN3	CAN 3 rx warning	Level
1B8	110	94	CAN3	CAN 3 error Interrupt	Level
1BC	111	95	CAN3	CAN 3 FD error Interrupt	Level

Address	Vector	IRQ	Source module	Source description	Type
1C0	112	96	CAN3	CAN 3 pretended network wakeup Interrupt	Level
1C4	113	97	CAN3	CAN 3 self-wakeup Interrupt	Level
1C8	114	98	CAN3	CAN 3 ECC interrupt	Level
1CC	115	99	CAN3	CAN 3 Message buffer (0-15)	Level
1D0	116	100	CAN3	CAN 3 Message buffer (16-31)	Level
1D4	117	101	CAN3	CAN 3 Message buffer (32-47)	Level
1D8	118	102	CAN3	CAN 3 Message buffer (48-63)	Level
1DC	119	103	CAN4	CAN 4 bus off	Level
1E0	120	104	CAN4	CAN 4 tx warning	Level
1E4	121	105	CAN4	CAN 4 rx warning	Level
1E8	122	106	CAN4	CAN 4 error Interrupt	Level
1EC	123	107	CAN4	CAN 4 FD error Interrupt	Level
1F0	124	108	CAN4	CAN 4 pretended network wakeup Interrupt	Level
1F4	125	109	CAN4	CAN 4 self-wakeup Interrupt	Level
1F8	126	110	CAN4	CAN 4 ECC interrupt	Level
1FC	127	111	CAN4	CAN 4 Message buffer (0-15)	Level
200	128	112	CAN4	CAN 4 Message buffer (16-31)	Level
204	129	113	CAN4	CAN 4 Message buffer (32-47)	Level
208	130	114	CAN4	CAN 4 Message buffer (48-63)	Level
20C	131	115	CAN5	CAN 5 bus off	Level
210	132	116	CAN5	CAN 5 tx warning	Level
214	133	117	CAN5	CAN 5 rx warning	Level
218	134	118	CAN5	CAN 5 error Interrupt	Level
21C	135	119	CAN5	CAN 5 FD error Interrupt	Level
220	136	120	CAN5	CAN 5 pretended network wakeup Interrupt	Level
224	137	121	CAN5	CAN 5 self-wakeup Interrupt	Level
228	138	122	CAN5	CAN 5 ECC interrupt	Level
22C	139	123	CAN5	CAN 5 Message buffer (0-15)	Level
230	140	124	CAN5	CAN 5 Message buffer (16-31)	Level
234	141	125	CAN5	CAN 5 Message buffer (32-47)	Level
238	142	126	CAN5	CAN 5 Message buffer (48-63)	Level
23C	143	127	CAN6	CAN 6 bus off	Level
240	144	128	CAN6	CAN 6 tx warning	Level
244	145	129	CAN6	CAN 6 rx warning	Level
248	146	130	CAN6	CAN 6 error Interrupt	Level
24C	147	131	CAN6	CAN 6 FD error Interrupt	Level
250	148	132	CAN6	CAN 6 pretended network wakeup Interrupt	Level
254	149	133	CAN6	CAN 6 self-wakeup Interrupt	Level
258	150	134	CAN6	CAN 6 ECC interrupt	Level
25C	151	135	CAN6	CAN 6 Message buffer (0-31)	Level
260	152	136	CAN6	CAN 6 Message buffer (32-63)	Level

Address	Vector	IRQ	Source module	Source description	Type
264	153	137	CAN6	CAN 6 Message buffer (64-95)	Level
268	154	138	CAN6	CAN 6 Message buffer (96-127)	Level
26C	155	139	CAN7	CAN 7 bus off	Level
270	156	140	CAN7	CAN 7 tx warning	Level
274	157	141	CAN7	CAN 7 rx warning	Level
278	158	142	CAN7	CAN 7 error Interrupt	Level
27C	159	143	CAN7	CAN 7 FD error Interrupt	Level
280	160	144	CAN7	CAN 7 pretended network wakeup Interrupt	Level
284	161	145	CAN7	CAN 7 self-wakeup Interrupt	Level
288	162	146	CAN7	CAN 7 ECC interrupt	Level
28C	163	147	CAN7	CAN 7 Message buffer (0-31)	Level
290	164	148	CAN7	CAN 7 Message buffer (32-63)	Level
294	165	149	CAN7	CAN 7 Message buffer (64-95)	Level
298	166	150	CAN7	CAN 7 Message buffer (96-127)	Level
29C	167	151	Reserved		
2A0	168	152	Reserved		
2A4	169	153	Reserved		
2A8	170	154	TIM0	TIM0 channel interrupt	Level
2AC	171	155	TIM0	TIM0 fault interrupt	Level
2B0	172	156	TIM0	TIM0 overflow interrupt	Level
2B4	173	157	TIM0	TIM0 reload interrupt	Level
2B8	174	158	TIM1	TIM1 channel interrupt	Level
2BC	175	159	TIM1	TIM1 fault interrupt	Level
2C0	176	160	TIM1	TIM1 overflow interrupt	Level
2C4	177	161	TIM1	TIM1 reload interrupt	Level
2C8	178	162	TIM2	TIM2 channel interrupt	Level
2CC	179	163	TIM2	TIM2 fault interrupt	Level
2D0	180	164	TIM2	TIM2 overflow interrupt	Level
2D4	181	165	TIM2	TIM2 reload interrupt	Level
2D8	182	166	TIM3	TIM3 channel interrupt	Level
2DC	183	167	TIM3	TIM3 fault interrupt	Level
2E0	184	168	TIM3	TIM3 overflow interrupt	Level
2E4	185	169	TIM3	TIM3 reload interrupt	Level
2E8	186	170	TDG0	TDG0 tco interrupt	Level
2EC	187	171	TDG0	TDG0 err interrupt	Level
2F0	188	172	TDG1	TDG1 tco interrupt	Level
2F4	189	173	TDG1	TDG1 err interrupt	Level
2F8	190	174	I2S0	I2S0 interrupt	Level
2FC	191	175	I2S1	I2S1 interrupt	Level
300	192	176	PORTA	PORTA Interrupt	Level
304	193	177	PORTB	PORTB Interrupt	Level

Address	Vector	IRQ	Source module	Source description	Type
308	194	178	PORTC	PORTC Interrupt	Level
30C	195	179	PORTD	PORTD Interrupt	Level
310	196	180	PORTE	PORTE Interrupt	Level
314	197	181	STIM	STIM interrupt	Level
318	198	182	RTC	RTC alarm	Level
31C	199	183	RTC	RTC second	Level
320	200	184	AES	AES interrupt	Pulse
324	201	185	TRNG	TRNG interrupt	Level
328	202	186	CMU0	CMU0 interrupt	Level
32C	203	187	CMU1	CMU1 interrupt	Level
330	204	188	CMU2	CMU2 interrupt	Level
334	205	189	SERU	SERU parity error interrupt	Level
338	206	190	SERU	SERU channel error interrupt1	Level
33C	207	191	SCC	System clock controller	Level
340	208	192	MCPWM0	MCPWM0 CH interrupt	Level
344	209	193	MCPWM0	MCPWM0 FAULT interrupt	Level
348	210	194	MCPWM0	MCPWM0 OVFL interrupt	Level
34C	211	195	MCPWM0	MCPWM0 RLFL interrupt	Level
350	212	196	MCPWM1	MCPWM1 CH interrupt	Level
354	213	197	MCPWM1	MCPWM1 FAULT interrupt	Level
358	214	198	MCPWM1	MCPWM1 OVFL interrupt	Level
35C	215	199	MCPWM1	MCPWM1 RLFL interrupt	Level
360	216	200	ADC0	ADC0 interrupt request.	Level
364	217	201	ADC1	ADC1 interrupt request.	Level
368	218	202	CMP	CMP interrupt request	Level
36C	219	203	FPU	FPU error interrupt	Level
370	220	204	CACHE	cache error interrupt	Level

5.3 WIC introduction

The primary function of the WIC block is to detect asynchronous wake-up events in STOP modes and signal to clock control logic to resume system clocking. After clock restart, the NVIC observes the pending interrupt and performs the normal interrupt or event processing.

5.3.1 Wake-up sources

The device uses the following internal and external inputs to the WIC module.

Table 5-2 WIC stop wake-up sources

Wake-up source	Description
Available system resets	RESET pin, WDOG
Pin interrupts	Port Control Module - Any enabled pin interrupt is capable of waking the system
ADC	Async Interrupt

Wake-up source	Description
CMP	Async Interrupt
STIM	Async Interrupt
UART, CAN, RTC	Async Interrupt
NMI	Non-maskable interrupt
PMU	LVW/LVD async interrupt
SRMC	Reset source async interrupt
SERU	Channel error/parity error async interrupt

Chapter 6

System Control Module (SCM)

6.1 Introduction

The System Control Module (SCM) provides system control and chip configuration registers.

6.1.1 Features

The SCM supports the following key features:

- Clock output configuration
- FPU Error Detect
- Cache Parity Error Detect
- Flash and system RAM size configuration
- Timer clock and channel selection and configuration
- Generate software trigger to TMU
- System device identification (ID)

6.2 Signal description

The SCM has an output signal connected to the external chip pin for clock output.

6.3 Memory map and register definition

This section provides the memory map and register definitions for the SCM module. The SCM has fields for selecting the clock output source and dividers, timer clock, device identification and unique identification number.

6.3.1 Memory map

SCM base address: 0x40048000

Table 6-1 SCM memory map

Offset	Register	Width (In bits)	Reset Value
0x00	Chip Control Register (SCM_CHIPCTL)	32	0x00001000
0x04	Misc Control Register 1 (SCM_MISCCTL1)	32	0x00004000
0x28	Misc Control Register 2 (SCM_MISCCTL2)	32	0x00000000
0x40	Misc Status register 1 (SCM_MISCSTAT1)	32	0x00000000
0x44	Misc Data Register 1 (SCM_MISCDATA1)	32	0x00000000
0x48	Misc Data Register 2 (SCM_MISCDATA2)	32	0x00000000
0x100	Chip Option Register (SCM_CHIPOPT)	32	0x00000000
0x104	Flash Configuration Register (SCM_FLSCFG)	32	Depends on device
0x108	RAM Configuration Register (SCM_RAMCFG)	32	Depends on device

Offset	Register	Width (In bits)	Reset Value
0x11C	System Device Identification Register (SCM_DEVID)	32	Depends on device
0x120+0x4*n (n=0~3)	Unique Identification Register n (SCM_UNIQUE _n)	32	See register description

6.3.2 Chip Control Register (SCM_CHIPCTL)

6.3.2.1 Offset

Register	Offset
SCM_CHIPCTL	0x00

6.3.2.2 Function

This register is used for chip level control.

6.3.2.3 Fields

Field	Access	Description	Reset Value
3:0 RSVD_3_0	RO	Reserved	0x0
5:4 CLKOUTSEL	RW	Clock out selection 00b: SCC clock out 01b: OSC32K 10b: LPO32K 11b: Reserved	0x0
7:6 RSVD_7_6	RO	Reserved	0x0
10:8 CLKOUTDIV	RW	Clock out divider 000b: Divide by 1 001b: Divide by 2 010b: Divide by 4 011b: Divide by 8 Others: Unsupported	0x0
11 CLKOUTEN	RW	Clock out enable 0b: Clock out disable 1b: Clock out enable NOTE The below sequence should be followed for CLKOUT configuration: 1.Configure SCM_CHIPCTL[CLKOUTSEL] 2.Configure SCM_CHIPCTL[CLKOUTDIV] 3.Enable SCM_CHIPCTL[CLKOUTEN] (While switching, CLKOUTEN should be first cleared and then the above sequence should be followed.)	0x0
31:12 RSVD_31_12	RO	Reserved	0x1

6.3.3 Misc Control Register 1 (SCM_MISCCTL1)

6.3.3.1 Offset

Register	Offset
SCM_MISCCTL1	0x04

6.3.3.2 Function

This register is used for misc control.

6.3.3.3 Fields

Field	Access	Description	Reset Value
0 SWTRIG0	RW	Software Trigger to TMU 0b: Releases the trigger to TMU 1b: Generates software trigger to TMU	0x0
1 SWTRIG1	RW	Software Trigger to TMU 0b: Releases the trigger to TMU 1b: Generates software trigger to TMU	0x0
2 SWTRIG2	RW	Software Trigger to TMU 0b: Releases the trigger to TMU 1b: Generates software trigger to TMU	0x0
3 SWTRIG3	RW	Software Trigger to TMU 0b: Releases the trigger to TMU 1b: Generates software trigger to TMU	0x0
4 SRAML_READ_ BUFF_EN	RW	SRAML read buffer enable 0b: Disable 1b: Enable	0x0
5 SRAMU_READ_ BUFF_EN	RW	SRAMU read buffer enable 0b: Disable 1b: Enable	0x0
6 XBAR_ROUND_ ROBIN	RW	Round robin enable 0b: Disable 1b: Enable	0x0
10:7 RSVD_10_7	RO	Reserved.	0x0
11 CCACHE_CLR	RW	Cache clear 0b: Cache is not cleared. 1b: Cache is cleared. Note: the cache can be cleared by writing 1 and then 0. It is recommended to mask all interrupts during the cache clearing.	0x0
12 CCACHE_PARIT Y_MISS_EN	RW	Cache parity miss enable 0b: Disable cache parity miss 1b: Enable cache parity miss Note: System handle it as "miss", not a "hit", when a parity error occurs.	0x0
13 CCACHE_PARIT Y_FAULT_EN	RW	This bit only enables the reporting of parity fault to CACHE_BE_FLAG, it does not influence the CCACHE_PARITY_MISS_EN. 0b: Disable cache parity fault 1b: Enable cache parity fault Note: If it is enabled, it must be enabled before writing 0b to CACHE_DIS.	0x0
14 CACHE_DIS	RW	Cache disable 0b: Cache is enabled.	0x1

Field	Access	Description	Reset Value
		1b: Cache is disabled, all cacheable accesses pass through cache. Note: It is recommended to enable CCACHE_PARITY_MISS_EN before writing 0b to CACHE_DIS.	
15 RSVD_15	RO	Reserved	0x0
16 CCACHE_BE_INT_EN	RW	Cache bit error interrupt enable 0b: Cache bit error interrupt disable. 1b: Cache bit error interrupt enable.	0x0
23:17 RSVD_23_17	RO	Reserved	0x0
24 FPU_INEXACT_INT_EN	RW	FPU inexact interrupt enable 0b: Disable. 1b: Enable.	0x0
25 FPU_OVERFLOW_INT_EN	RW	FPU overflow interrupt enable 0b: Disable. 1b: Enable.	0x0
26 FPU_UNDERFLOW_INT_EN	RW	FPU underflow interrupt enable 0b: Disable. 1b: Enable.	0x0
27 FPU_INVALIDOP_INT_EN	RW	FPU invalid option interrupt enable 0b: Disable. 1b: Enable.	0x0
28 FPU_DIVZERO_INT_EN	RW	FPU divide by zero interrupt enable 0b: Disable. 1b: Enable.	0x0
29 FPU_DENORMALIN_INT_EN	RW	FPU denormal input interrupt enable 0b: Disable. 1b: Enable.	0x0
31:30 RSVD_31_30	RO	Reserved	0x0

6.3.4 Misc Control Register 2 (SCM_MISCCTL2)

6.3.4.1 Offset

Register	Offset
SCM_MISCCTL2	0x28

6.3.4.2 Function

This register is used for misc control.

6.3.4.3 Fields

Field	Access	Description	Reset Value
7:0 RSVD_7_0	RO	Reserved	0x0
9:8 MCPWM0_TCLK_SEL	RW	MCPWM0 external input clock select 00b: TCLK0 01b: TCLK1 10b: TCLK2	0x0

Field	Access	Description	Reset Value
		11b: RSVD	
11:10 MCPWM1_TCLK_SEL	RW	MCPWM1 external input clock select 00b: TCLK0 01b: TCLK1 10b: TCLK2 11b: RSVD	0x0
13:12 TIM0_TCLK_SEL	RW	TIM0 external input clock select 00b: TCLK0 01b: TCLK1 10b: TCLK2 11b: RSVD	0x0
15:14 TIM1_TCLK_SEL	RW	TIM1 external input clock select 00b: TCLK0 01b: TCLK1 10b: TCLK2 11b: RSVD	0x0
17:16 TIM2_TCLK_SEL	RW	TIM2 external input clock select 00b: TCLK0 01b: TCLK1 10b: TCLK2 11b: RSVD	0x0
19:18 TIM3_TCLK_SEL	RW	TIM3 external input clock select 00b: TCLK0 01b: TCLK1 10b: TCLK2 11b: RSVD	0x0
23:20 RSVD_23_20	RO	Reserved	0x0
24 ADC0_CH8_SEL	RW	ADC0 channel 8 mux select 0b: ADC0 channel 8 mux to PTB13. 1b: ADC0 channel 8 mux to PTC0.	0x0
25 ADC0_CH9_SEL	RW	ADC0 channel 9 mux select 0b: ADC0 channel 9 mux to PTB14. 1b: ADC0 channel 9 mux to PTC1.	0x0
26 ADC1_CH14_SEL	RW	ADC1 channel 14 mux select 0b: ADC1 channel 14 mux to PTB0. 1b: ADC1 channel 14 mux to PTB15.	0x0
27 ADC1_CH15_SEL	RW	ADC1 channel 15 mux select 0b: ADC1 channel 15 mux to PTB1. 1b: ADC1 channel 15 mux to PTB16.	0x0
31:28 RSVD_31_28	RO	Reserved	0x0

6.3.5 Misc Status Register 1 (SCM_MISCSTAT1)

6.3.5.1 Offset

Register	Offset
SCM_MISCSTAT1	0x40

6.3.5.2 Function

This register is used for misc status indication.

6.3.5.3 Fields

Field	Access	Description	Reset Value
14:0 RSVD_14_0	RO	Reserved	0x0
15 CCACHE_WBE_FLAG	W1C	Cache write buffer error flag 0b: Cache write buffer error flag not happened. 1b: Cache write buffer error flag happened.	0x0
16 CACHE_BE_FLAG	W1C	CACHE bit error flag 0b: Cache bit error flag not happened. 1b: Cache bit error flag happened.	0x0
17 CACHE_BE_SUPV	RO	Indicates CPU mode when CACHE bit error happened. This bit is cleared by writing 1 to CACHE_BE_FLAG. 0b: User mode. 1b: Supervisor mode.	0x0
19:18 CACHE_BE_HSIZE	RO	Indicates bus access size when CACHE bit error happened. This bit is cleared by writing 1 to CACHE_BE_FLAG. 00b: No access 01b: 8-bit access 10b: 16-bit access 11b: 32-bit access	0x0
23:20 CACHE_BE_HMAST	RO	Indicates bus master when CACHE bit error happened. This bit is cleared by writing 1 to CACHE_BE_FLAG. 00b: CPU 01b: Debug 10b: DMA 11b: Reserved	0x0
24 FPU_INEXACT_FLAG	RO	FPU inexact flag 0b: FPU inexact not happened. 1b: FPU inexact happened.	0x0
25 FPU_OVERFLOW_FLAG	RO	FPU overflow flag 0b: FPU overflow not happened. 1b: FPU overflow happened.	0x0
26 FPU_UNDERFLOW_FLAG	RO	FPU underflow flag 0b: FPU underflow not happened. 1b: FPU underflow happened.	0x0
27 FPU_INVALIDOP_FLAG	RO	FPU invalid operation flag 0b: FPU invalid operation not happened. 1b: FPU invalid operation happened.	0x0
28 FPU_DIVZERO_FLAG	RO	FPU divide by zero flag 0b: FPU divide by zero not happened. 1b: FPU divide by zero happened.	0x0
29 FPU_DENORMALIN_FLAG	RO	FPU denormal input flag 0b: FPU denormal input not happened. 1b: FPU denormal input happened.	0x0
31:30 RSVD_31_30	RO	Reserved	0x0

6.3.6 Misc Data Register 1 (SCM_MISCDATA1)

6.3.6.1 Offset

Register	Offset
SCM_MISCDATA1	0x44

6.3.6.2 Function

This register is used for misc data storage.

6.3.6.3 Fields

Field	Access	Description	Reset Value
31:0 CACHE_BE_ADDR	RO	Cache bit error address.	0x0

6.3.7 Misc Data register 2 (SCM_MISCDATA2)

6.3.7.1 Offset

Register	Offset
SCM_MISCDATA2	0x48

6.3.7.2 Function

This register is used for misc data storage.

6.3.7.3 Fields

Field	Access	Description	Reset Value
31:0 CACHE_BE_DATA	RO	Cache bit error data.	0x0

6.3.8 Chip Option Register (SCM_CHIPOPT)

6.3.8.1 Offset

Register	Offset
SCM_CHIPOPT	0x100

6.3.8.2 Function

This register is used for chip level option.

6.3.8.3 Fields

Field	Access	Description	Reset Value
0 CCACHE_DIS_IFR	RO	This bit indicates if chip has the cache field. 0b: The chip contains the cache field. 1b: The chip does not contain the cache field.	0x0

Field	Access	Description	Reset Value
1 FPU_DIS	RO	This bit indicates if chip has the FPU module. 0b: The chip contains the FPU module. 1b: The chip does not contain the FPU module.	0x0
31:2 RSVD_31_2	RO	Reserved	0x0

6.3.9 Flash Configuration Register (SCM_FLSCFG)

6.3.9.1 Offset

Register	Offset
SCM_FLSCFG	0x104

6.3.9.2 Function

This register is used to indicate flash configuration on this device.

6.3.9.3 Fields

Field	Access	Description	Reset Value
23:0 RSVD_23_0	RO	Reserved	0x0
27:24 PFSIZE	RO	Program flash size 00b: 256K 01b: 512K 10b: 1M 11b: 2M	Depends on device
31:28 DFSIZE	RO	Data flash size 00b: 16K 01b: 32K 10b: 64K 11b: 128K	Depends on device

6.3.10 RAM Configuration Register (SCM_RAMCFG)

6.3.10.1 Offset

Register	Offset
SCM_RAMCFG	0x108

6.3.10.2 Function

This register is used to indicate sram configuration on this device.

6.3.10.3 Fields

Field	Access	Description	Reset Value
3:0 CACHESIZE	RO	Cache size 0100b: 4K 0011b: 2K	Depends on device

Field	Access	Description	Reset Value
		Others: Reserved	
27:4 RSVD_27_4	RO	Reserved	0x0
31:28 RAMSIZE	RO	SRAM size 00b: SRAML 0K, SRAMU 32K 01b: SRAML 0K, SRAMU 64K 10b: SRAML 0K, SRAMU 128K 11b: SRAML 128K, SRAMU 128K	Depends on device

6.3.11 System Device Identification Register (SCM_DEVID)

6.3.11.1 Offset

Register	Offset
SCM_DEVID	0x11C

6.3.11.2 Function

This register is for this device ID.

6.3.11.3 Fields

Field	Access	Description	Reset Value
3:0 MEMSIZE_ID	RO	Memory size ID 0010b: 256KB 0100b: 512KB 0110b: 1MB 1000b: 2MB Others: Reserved	Depends on device
6:4 SERIES_ID	RO	Series ID 000b: 1 001b: 4	Depends on device
11:7 SUBFAMILY_ID	RO	Sub family ID 0001b: 1 0011b: 3	Depends on device
15:12 FAMILY_ID	RO	Family ID 0000b: Z20K 0001b: Z20M	Depends on device
19:16 REV_ID	RO	Revision ID	Depends on device
26:20 FEATURE_ID	RO	Feature ID	Depends on device
31:27 PACKAGE_ID	RO	Package ID 00001b: 16 TSSOP 00010b: 20TSSOP 00011b: 24 QFN 00100b: 32 LQFP 00101b: 32 QFN 00110b: 48 LQFP 00111b: 48 QFN 01000b: 64 LQFP 01001b: 80 LQFP	Depends on device

Field	Access	Description	Reset Value
		01010b: 100 LQFP 01011b: 100 BGA 01100b: 144 LQFP 01101b: 176 LQFP	

6.3.12 Unique Identification Register n (SCM_UNIQUE_n)

6.3.12.1 Offset

Register	Offset
SCM_UNIQUE _n	0x120+0x4*n(n=0~3)

6.3.12.2 Function

UID_127_96, UID_95_64, UID_63_32, and UID_31_0 together represent a 128-bit unique identification number for this device.

6.3.12.3 Fields

Field	Access	Description	Reset Value
31:0 UID	RO	UID_31_0 UID_63_32 UID_95_64 UID_127_96	Depends on device

Chapter 7

Port Control and Pin Interrupts (PORT)

7.1 Introduction

The Port Control and Interrupt (PORT) module provides support for functions of port control, digital filtering, and pin interrupt. Most of the functions can be configured independently for each pin in the 32-bit port and affect the pin regardless of what its pin muxing state is. There is one instance of the PORT module for each port. However, not all pins within each port are implemented on a specific device.

7.1.1 Features

The PORT module has the following features:

Pin interrupt

- Interrupt flag and enable registers for each pin
- Support for edge sensitive (rising, falling, both) behaviors or level sensitive (low, high) behaviors which are configured per pin
- Support NMI interrupt
- Support for interrupt or DMA request configured per pin
- Asynchronous wake-up in low-power modes
- Pin interrupt is functional in all digital pin-multiplexing modes

Digital input filter

- A digital input filter for each pin that can be used by any digital peripheral multiplexed onto the pin. However, this digital filter function will not work in low power mode
- Individual enable or bypass control field per pin
- A selectable clock source for digital input filter with a five-bit resolution on filter size
- Functional in all digital pin multiplexing modes

Port control

- Individual pull control fields, which support pullup, pulldown, and pull-disable
- Enable and disable the individual open-drain mode
- Passive filter can be successfully enabled only on the pins of NMI_b and RESET_b. When this function is enabled, the input level holding time is filtered if it is less than 20ns, and it is not filtered if it is more than 150ns. This also works in low power mode
- Individual mux control field supporting analog or pin disabled, GPIO, and up to six chip-specific digital functions
- Pad configuration fields are functional in all digital pin-multiplexing modes
- When passive filter and digital filter are enabled at the same time, signals on the pin pass through the passive filter and then the digital filter
- Support slow slew rate and fast slew rate

7.2 Signal description

The following table details the signal description of the PORT interface.

Table 7-1 PORT interface signal descriptions

PORTx[31:0]	I/O	Pin interrupt	
		State meaning	Asserted—pin is set to logic 1.
			Negated—pin is set to logic 0.

		Timing	Assertion—may occur at any time and can assert asynchronously to the system clock.
			Negation—may occur at any time and can assert asynchronously to the system clock.

7.3 Memory map and register definition

This section details the memory map and register definitions of the PORT module.

7.3.1 Memory map

PORTA base address: 0x40049000
 PORTB base address: 0x4004A000
 PORTC base address: 0x4004B000
 PORTD base address: 0x40050000
 PORTE base address: 0x40051000

Table 7-2 PORT memory map

Offset	Register	Width (In bits)	Reset Value
0x8	Global Pin Data Register (PORTA_GLB DAT)	32	0x00000000
0xC	Global Pin Control Register (PORTA_GLB CTL)	32	0x00000000
0x10	PORTA Pin Control Register 0 (PORTA_PCR0)	32	0x00000000
0x14	PORTA Pin Control Register 1 (PORTA_PCR1)	32	0x00000000
0x18	PORTA Pin Control Register 2 (PORTA_PCR2)	32	0x00000000
0x1C	PORTA Pin Control Register 3 (PORTA_PCR3)	32	0x00000000
0x20	PORTA Pin Control Register 4 (PORTA_PCR4)	32	0x00000703
0x24	PORTA Pin Control Register 5 (PORTA_PCR5)	32	0x00000713
0x28	PORTA Pin Control Register 6 (PORTA_PCR6)	32	0x00000000
0x2C	PORTA Pin Control Register 7 (PORTA_PCR7)	32	0x00000000
0x30	PORTA Pin Control Register 8 (PORTA_PCR8)	32	0x00000000
0x34	PORTA Pin Control Register 9 (PORTA_PCR9)	32	0x00000000
0x38	PORTA Pin Control Register 10 (PORTA_PCR10)	32	0x00000000
0x3C	PORTA Pin Control Register 11 (PORTA_PCR11)	32	0x00000000
0x40	PORTA Pin Control Register 12 (PORTA_PCR12)	32	0x00000000
0x44	PORTA Pin Control Register 13 (PORTA_PCR13)	32	0x00000000

Offset	Register	Width (In bits)	Reset Value
	(PORTA_PCR13)		
0x4C	PORTA Pin Control Register 15 (PORTA_PCR15)	32	0x00000000
0x50	PORTA Pin Control Register 16 (PORTA_PCR16)	32	0x00000000
0x54	PORTA Pin Control Register 17 (PORTA_PCR17)	32	0x00000000
0x74	PORTA Pin Control Register 25 (PORTA_PCR25)	32	0x00000000
0x78	PORTA Pin Control Register 26 (PORTA_PCR26)	32	0x00000000
0x7C	PORTA Pin Control Register 27 (PORTA_PCR27)	32	0x00000000
0x80	PORTA Pin Control Register 28 (PORTA_PCR28)	32	0x00000000
0x84	PORTA Pin Control Register 29 (PORTA_PCR29)	32	0x00000000
0x88	PORTA Pin Control Register 30 (PORTA_PCR30)	32	0x00000000
0x8C	PORTA Pin Control Register 31 (PORTA_PCR31)	32	0x00000000
0x90	PORTA Interrupt Request Flag Register (PORTA_IRQFLG)	32	0x00000000
0x08	Global Pin Data Register (PORTB_GLB DAT)	32	0x00000000
0x0C	Global Pin Control Register (PORTB_GLB CTL)	32	0x00000000
0x10	PORTB Pin Control Register 0 (PORTB_PCR0)	32	0x00000000
0x14	PORTB Pin Control Register 1 (PORTB_PCR1)	32	0x00000000
0x18	PORTB Pin Control Register 2 (PORTB_PCR2)	32	0x00000000
0x1C	PORTB Pin Control Register 3 (PORTB_PCR3)	32	0x00000000
0x20	PORTB Pin Control Register 4 (PORTB_PCR4)	32	0x00000000
0x24	PORTB Pin Control Register 5 (PORTB_PCR5)	32	0x00000000
0x28	PORTB Pin Control Register 6 (PORTB_PCR6)	32	0x00000000
0x2C	PORTB Pin Control Register 7 (PORTB_PCR7)	32	0x00000000
0x30	PORTB Pin Control Register 8 (PORTB_PCR8)	32	0x00000000
0x34	PORTB Pin Control Register 9 (PORTB_PCR9)	32	0x00000000
0x38	PORTB Pin Control Register 10 (PORTB_PCR10)	32	0x00000000
0x3C	PORTB Pin Control Register 11 (PORTB_PCR11)	32	0x00000000
0x40	PORTB Pin Control Register 12 (PORTB_PCR12)	32	0x00000000
0x44	PORTB Pin Control Register 13	32	0x00000000

Offset	Register	Width (In bits)	Reset Value
	(PORTB_PCR13)		
0x48	PORTB Pin Control Register 14 (PORTB_PCR14)	32	0x00000000
0x4C	PORTB Pin Control Register 15 (PORTB_PCR15)	32	0x00000000
0x50	PORTB Pin Control Register 16 (PORTB_PCR16)	32	0x00000000
0x54	PORTB Pin Control Register 17 (PORTB_PCR17)	32	0x00000000
0x58	PORTB Pin Control Register 18 (PORTB_PCR18)	32	0x00000000
0x60	PORTB Pin Control Register 20 (PORTB_PCR20)	32	0x00000000
0x64	PORTB Pin Control Register 21 (PORTB_PCR21)	32	0x00000000
0x68	PORTB Pin Control Register 22 (PORTB_PCR22)	32	0x00000000
0x6C	PORTB Pin Control Register 23 (PORTB_PCR23)	32	0x00000000
0x74	PORTB Pin Control Register 25 (PORTB_PCR25)	32	0x00000000
0x7C	PORTB Pin Control Register 27 (PORTB_PCR27)	32	0x00000000
0x80	PORTB Pin Control Register 28 (PORTB_PCR28)	32	0x00000000
0x84	PORTB Pin Control Register 29 (PORTB_PCR29)	32	0x00000000
0x90	PORTB Interrupt Request Flag Register (PORTB_IRQFLG)	32	0x00000000
0x08	Global Pin Data Register (PORTC_GLB DAT)	32	0x00000000
0x0C	Global Pin Control Register (PORTC_GLBCTL)	32	0x00000000
0x10	PORTC Pin Control Register 0 (PORTC_PCR0)	32	0x00000000
0x14	PORTC Pin Control Register 1 (PORTC_PCR1)	32	0x00000000
0x18	PORTC Pin Control Register 2 (PORTC_PCR2)	32	0x00000000
0x1C	PORTC Pin Control Register 3 (PORTC_PCR3)	32	0x00000000
0x20	PORTC Pin Control Register 4 (PORTC_PCR4)	32	0x00000702
0x24	PORTC Pin Control Register 5 (PORTC_PCR5)	32	0x00000703
0x28	PORTC Pin Control Register 6 (PORTC_PCR6)	32	0x00000000
0x2C	PORTC Pin Control Register 7 (PORTC_PCR7)	32	0x00000000
0x30	PORTC Pin Control Register 8 (PORTC_PCR8)	32	0x00000000
0x34	PORTC Pin Control Register 9 (PORTC_PCR9)	32	0x00000000
0x38	PORTC Pin Control Register 10	32	0x00000000

Offset	Register	Width (In bits)	Reset Value
	(PORTC_PCR10)		
0x3C	PORTC Pin Control Register 11 (PORTC_PCR11)	32	0x00000000
0x40	PORTC Pin Control Register 12 (PORTC_PCR12)	32	0x00000000
0x44	PORTC Pin Control Register 13 (PORTC_PCR13)	32	0x00000000
0x48	PORTC Pin Control Register 14 (PORTC_PCR14)	32	0x00000000
0x4C	PORTC Pin Control Register 15 (PORTC_PCR15)	32	0x00000000
0x50	PORTC Pin Control Register 16 (PORTC_PCR16)	32	0x00000000
0x54	PORTC Pin Control Register 17 (PORTC_PCR17)	32	0x00000000
0x5C	PORTC Pin Control Register 19 (PORTC_PCR19)	32	0x00000000
0x6C	PORTC Pin Control Register 23 (PORTC_PCR23)	32	0x00000000
0x7C	PORTC Pin Control Register 27 (PORTC_PCR27)	32	0x00000000
0x80	PORTC Pin Control Register 28 (PORTC_PCR28)	32	0x00000000
0x84	PORTC Pin Control Register 29 (PORTC_PCR29)	32	0x00000000
0x88	PORTC Pin Control Register 30 (PORTC_PCR30)	32	0x00000000
0x8C	PORTC Pin Control Register 31 (PORTC_PCR31)	32	0x00000000
0x90	PORTC Interrupt Request Flag Register (PORTC_IRQFLG)	32	0x00000000
0x08	Global Pin Data Register (PORTD_GLB DAT)	32	0x00000000
0x0C	Global Pin Control Register (PORTD_GLBCTL)	32	0x00000000
0x10	PORTD Pin Control Register 0 (PORTD_PCR0)	32	0x00000000
0x14	PORTD Pin Control Register 1 (PORTD_PCR1)	32	0x00000000
0x18	PORTD Pin Control Register 2 (PORTD_PCR2)	32	0x00000000
0x1C	PORTD Pin Control Register 3 (PORTD_PCR3)	32	0x00000000
0x20	PORTD Pin Control Register 4 (PORTD_PCR4)	32	0x00000000
0x24	PORTD Pin Control Register 5 (PORTD_PCR5)	32	0x00000000
0x28	PORTD Pin Control Register 6 (PORTD_PCR6)	32	0x00000000
0x2C	PORTD Pin Control Register 7 (PORTD_PCR7)	32	0x00000000
0x30	PORTD Pin Control Register 8 (PORTD_PCR8)	32	0x00000000
0x34	PORTD Pin Control Register 9	32	0x00000000

Offset	Register	Width (In bits)	Reset Value
	(PORTD_PCR9)		
0x38	PORTD Pin Control Register 10 (PORTD_PCR10)	32	0x00000000
0x3C	PORTD Pin Control Register 11 (PORTD_PCR11)	32	0x00000000
0x40	PORTD Pin Control Register 12 (PORTD_PCR12)	32	0x00000000
0x44	PORTD Pin Control Register 13 (PORTD_PCR13)	32	0x00000000
0x48	PORTD Pin Control Register 14 (PORTD_PCR14)	32	0x00000000
0x4C	PORTD Pin Control Register 15 (PORTD_PCR15)	32	0x00000000
0x50	PORTD Pin Control Register 16 (PORTD_PCR16)	32	0x00000000
0x54	PORTD Pin Control Register 17 (PORTD_PCR17)	32	0x00000000
0x58	PORTD Pin Control Register 18 (PORTD_PCR18)	32	0x00000000
0x5C	PORTD Pin Control Register 19 (PORTD_PCR19)	32	0x00000000
0x68	PORTD Pin Control Register 22 (PORTD_PCR22)	32	0x00000000
0x6C	PORTD Pin Control Register 23 (PORTD_PCR23)	32	0x00000000
0x70	PORTD Pin Control Register 24 (PORTD_PCR24)	32	0x00000000
0x7C	PORTD Pin Control Register 27 (PORTD_PCR27)	32	0x00000000
0x80	PORTD Pin Control Register 28 (PORTD_PCR28)	32	0x00000000
0x84	PORTD Pin Control Register 29 (PORTD_PCR29)	32	0x00000000
0x88	PORTD Pin Control Register 30 (PORTD_PCR30)	32	0x00000000
0x90	PORTD Interrupt Request Flag Register (PORTD_IRQFLG)	32	0x00000000
0x08	Global Pin Data Register (PORTE_GLB DAT)	32	0x00000000
0x0C	Global Pin Control Register (PORTE_GLB CTL)	32	0x00000000
0x10	PORTE Pin Control Register 0 (PORTE_PCR0)	32	0x00000700
0x14	PORTE Pin Control Register 1 (PORTE_PCR1)	32	0x00000000
0x18	PORTE Pin Control Register 2 (PORTE_PCR2)	32	0x00000000
0x1C	PORTE Pin Control Register 3 (PORTE_PCR3)	32	0x00000000
0x28	PORTE Pin Control Register 6 (PORTE_PCR6)	32	0x00000000
0x2C	PORTE Pin Control Register 7 (PORTE_PCR7)	32	0x00000000
0x30	PORTE Pin Control Register 8	32	0x00000000

Offset	Register	Width (In bits)	Reset Value
	(PORTE_PCR8)		
0x34	PORTE Pin Control Register 9 (PORTE_PCR9)	32	0x00000000
0x3C	PORTE Pin Control Register 11 (PORTE_PCR11)	32	0x00000000
0x40	PORTE Pin Control Register 12 (PORTE_PCR12)	32	0x00000000
0x44	PORTE Pin Control Register 13 (PORTE_PCR13)	32	0x00000000
0x48	PORTE Pin Control Register 14 (PORTE_PCR14)	32	0x00000000
0x4C	PORTE Pin Control Register 15 (PORTE_PCR15)	32	0x00000000
0x50	PORTE Pin Control Register 16 (PORTE_PCR16)	32	0x00000000
0x5C	PORTE Pin Control Register 19 (PORTE_PCR19)	32	0x00000000
0x60	PORTE Pin Control Register 20 (PORTE_PCR20)	32	0x00000000
0x64	PORTE Pin Control Register 21 (PORTE_PCR21)	32	0x00000000
0x68	PORTE Pin Control Register 22 (PORTE_PCR22)	32	0x00000000
0x6C	PORTE Pin Control Register 23 (PORTE_PCR23)	32	0x00000000
0x70	PORTE Pin Control Register 24 (PORTE_PCR24)	32	0x00000000
0x74	PORTE Pin Control Register 25 (PORTE_PCR25)	32	0x00000000
0x90	PORTE Interrupt Request Flag Register (PORTE_IRQFLG)	32	0x00000000

7.3.2 Global Pin Data Register (PORTx_GLB DAT)

7.3.2.1 Offset

Register	Offset
PORTx_GLB DAT	0x8 + 0x1000 * x, When the user selects port A, x is equal to 0 When the user selects port B, x is equal to 1 When the user selects port C, x is equal to 2 When the user selects port D, x is equal to 7 When the user selects port E, x is equal to 8

7.3.2.2 Function

This is the Global Pin Data Register.

7.3.2.3 Fields

Field	Access	Description	Reset Value

Field	Access	Description	Reset Value
31:0 GPWD	RW	This register value is used to update all PCRN registers by writing 1 to GPWE bits	0x0

7.3.3 Global Pin Control Register (PORTx_GLBCTL)

7.3.3.1 Offset

Register	Offset
PORTx_GLBCTL	$0xC + 0x1000 * x$, When the user selects port A, x is equal to 0 When the user selects port B, x is equal to 1 When the user selects port C, x is equal to 2 When the user selects port D, x is equal to 7 When the user selects port E, x is equal to 8

7.3.3.2 Function

This is the Global Pin Control Register.

7.3.3.3 Fields

Field	Access	Description	Reset Value
31:0 GPWE	WO	0b: Corresponding Pin Control Register is not updated with the value in GPWD. 1b: Corresponding Pin Control Register is updated with the value in GPWD.	0x0

7.3.4 PORT Pin Control Register n (PORTx_PCRn)

7.3.4.1 Offset

Register	Offset
PORTx_PCRn	$0x10 + 4d * n + 0x1000 * x$, When the user selects port A, x is equal to 0 When the user selects port B, x is equal to 1 When the user selects port C, x is equal to 2 When the user selects port D, x is equal to 7 When the user selects port E, x is equal to 8

7.3.4.2 Function

This is the single pin control register. Each of the registers is one-to-one with the unique corresponding pin and is used to control the pin MUX, pull-up/down, drive strength, filter function, interrupt, DMA, etc.

7.3.4.3 Fields

Field	Access	Description	Reset Value
0 PS	RW	0b: Internal pulldown resistor is enabled if the corresponding PE field is set.	* Note: the

Field	Access	Description	Reset Value
		1b: Internal pullup resistor is enabled if the corresponding PE field is set. Note: The pull-up or pull-down function is only valid when the pin is pinmuxed as digital function (PORTx_PCRn[MUX]=1~7).	value varies by port.
1 PE	RW	0b: Internal pullup or pulldown resistor is not enabled on the corresponding pin. 1b: Internal pullup or pulldown resistor is enabled if the pin is configured as a digital input or open-drain mode. Note: The pull-up or pull-down function is only valid when the pin is pinmuxed as digital function (PORTx_PCRn[MUX]=1~7).	* Note: the value varies by port.
2 RSVD_2	RO	Reserved	0x0
3 SRE	RW	0b: Fast slew rate is configured if the pin is configured as a digital output. 1b: Slow slew rate is configured if the pin is configured as a digital output.	0x0
4 PFE	RW	0b: Passive input filter is disabled on the corresponding pin. 1b: Passive input filter is enabled if the pin is configured as a digital input.	* Note: the value varies by port.
5 ODE	RW	0b: Open drain output is disabled on the corresponding pin. 1b: Open drain output is enabled if the pin is configured as a digital output.	0x0
7:6 RSVD_7_6	RO	Reserved	0x0
10:8 MUX	RW	000b: Alternative 0 (analog). 001b: Alternative 1 (GPIO). 010b: Alternative 2 (chip-specific). 011b: Alternative 3 (chip-specific). 100b: Alternative 4 (chip-specific). 101b: Alternative 5 (chip-specific). 110b: Alternative 6 (chip-specific). 111b: Alternative 7 (chip-specific).	* Note: the value varies by port.
14:11 RSVD_14_11	RO	Reserved	0x0
15 LK	RW	0b: Pin Control Register fields [31:0] are not locked. 1b: Pin Control Register fields [31:0] are locked and cannot be updated by PCR writing or global writing until this bit is cleared by writing 0x5B to this register [31:24].	0x0
19:16 IRQC	RW	0000b: Interrupt Status Flag (ISF) is disabled. 0001b: ISF flag and DMA request on rising edge. 0010b: ISF flag and DMA request on falling edge. 0011b: ISF flag and DMA request on either edge. 1000b: ISF flag and Interrupt when logic 0. 1001b: ISF flag and Interrupt on rising edge. 1010b: ISF flag and Interrupt on falling edge. 1011b: ISF flag and Interrupt on either edge. 1100b: ISF flag and Interrupt when logic 1. Others: Reserved.	0x0
22:20 RSVD_22_20	RO	Reserved	0x0
23	W1C	The pin interrupt configuration is valid in all digital pin	0x0

Field	Access	Description	Reset Value
ISF		muxing modes. Writing 1 to this bit will clear this bit if the interrupt event completes. 0b: Configured interrupt is not detected. 1b: Configured interrupt is detected.	
28:24 FLT	RW	0b: Filter disabled regardless of FLTEN. Others: Filter enabled with current FLT value as filter counter limit when FLTEN = 1.	0x0
30:29 RSVD_30_29	RO	Reserved	0x0
31 FLTEN	RW	0b: Port filter is disabled. 1b: Port filter is enabled.	0x0

7.3.5 PORT Interrupt Request Flag Register(PORTx_IRQFLG)

7.3.5.1 Offset

Register	Offset
PORTx_IRQFLG	0x90 + 0x1000 * x, When the user selects port A, x is equal to 0 When the user selects port B, x is equal to 1 When the user selects port C, x is equal to 2 When the user selects port D, x is equal to 7 When the user selects port E, x is equal to 8

7.3.5.2 Function

The Interrupt Status Flag for each pin is also visible in the corresponding Pin Control Register. Each flag can be cleared at its corresponding position in the register.

7.3.5.3 Fields

Field	Access	Description	Reset Value
31:0 ISF	W1C	Each bit in the field indicates the detection of the configured interrupt of the same number as the field. Writing 1 to any bit will clear the bit if the interrupt event on the bit has completed. 0b: Configured interrupt is not detected. 1b: Configured interrupt is detected.	0x0

7.4 Function description

7.4.1 Pin control

Every port pin has PORTx_PCRn, a corresponding Pin Control register, associated with it. The upper half of the Pin Control register configures the capability of the digital filter and the pin to either interrupt the CPU or request a DMA transfer on a rising/falling edge or on both edges as well as in a logic level occurring on the port pin. The Pin Control register also sets up a flag to indicate that an interrupt has occurred.

The lower half of the Pin Control register configures the following functions for each pin within the 32-bit port.

- Pullup or pulldown enable

- Slew rate
- Open-drain
- Passive input filter enable
- Pin-multiplexing mode

The functions apply across all digital pin-multiplexing modes. Individual peripherals do not override the configuration in the Pin Control register.

For example, if an I2C function is enabled on a pin, it will not override the pullup configuration for that pin.

When the pin-multiplexing mode is configured for analog or it is disabled, all the digital functions on that pin are disabled. This includes the pullup and pulldown enables, output buffer enable, input buffer enable, and passive filter enable. When using the NMI filter function, the pin needs to be pinmuxed as GPIO, then a pull-up configuration is provided, and then the pin is pinmuxed as NMI_b pin.

- The LK bit (bit 15 of Pin Control Register PCRN) allows the configuration for each pin to be locked until the next system reset. When locked, writes to the pin control register are ignored, although a bus error is not generated on an attempted write to a locked register.
- The configuration of each Pin Control register is retained when the PORT module is disabled.
- Whenever a pin is configured in any digital pin-multiplexing mode, the input buffer for that pin is enabled, which allows the pin state to be read via the corresponding GPIO Port Data Input Register (GPIO_PDIR) or allows a pin interrupt or DMA request to be generated.
- If a pin is ever floating when its input buffer is enabled, then it can cause an increase in power consumption. The user must avoid this situation.
- A pin can be floating due to an input pin which is not connected or an output pin which has tri-stated (output buffer is disabled).
- Enabling the internal pull resistor (or implementing an external pull resistor) will guarantee that a pin does not float when its input buffer is enabled.

NOTE

The internal pull resistor is automatically disabled whenever the output buffer is enabled, which allows the Pull Enable bit to remain set. When the pin is in ALT0 mode, the digital input-output mode is disabled in order to achieve the lowest power consumption.

It is worth noting that when multiple pins are configured for the same function, only one pin can be successfully configured, depending on the internal priority.

7.4.2 Global pin control

In order to update the 32 pins, the two global pin control registers, PORTx_GLB DAT and PORTx_GLB CTL, allow multiple pins to be written with the same value. Registers that are locked cannot be written using the global pin control registers.

The global pin control registers are designed to enable the software to configure multiple pins within one port for the same peripheral function. However, the global pin control register cannot be used to configure both the interrupt function and the pinmux function.

The global pin control registers are write-only, which always read as 0.

7.4.3 Global interrupt status

The global interrupt status register is designed to indicate if any pin interrupt event has been generated or not. To clear its interrupt flag, write 1 to the corresponding bit regardless of the lock status of the PCR registers.

7.4.4 Pin interrupts

The pin interrupt capability of the PORT module is available in all digital pin-multiplexing

modes provided the PORT module is enabled.

Each pin can be individually configured for any of the following pin interrupt modes:

- Interrupt disabled by default
- Active high level sensitive interrupt
- Active low level sensitive interrupt
- Rising edge sensitive interrupt
- Falling edge sensitive interrupt
- Rising and falling edge sensitive interrupt
- Rising edge sensitive DMA request
- Falling edge sensitive DMA request
- Rising and falling edge sensitive DMA request

The interrupt status flag is set when either of the following conditions occurs, provided the digital input filter is enabled:

- The configured edge or level is detected on the pin of the digital input filter;
- The configured edge or level is detected at the output of the digital input filter.

When not in STOP mode, the input is synchronized to the bus clock to detect the configured level or edge transition firstly.

- The PORT module generates a single interrupt that asserts when the interrupt status flag is set for any enabled interrupt for that port.
- The interrupt negates after the interrupt status flags for all enabled interrupts have been cleared by writing a logic 1 to the ISF flag in either the PORTx_IRQFLG or PORTx_PCRn registers.
- The PORT module generates a single DMA request that asserts when the interrupt status flag is set for any enabled DMA request in that port.
- The DMA request negates after the DMA transfer is completed, because that clears the interrupt status flags for all the enabled DMA requests.
- In the STOP mode, for any enabled interrupt, the interrupt status flag is asynchronously set if the required level or edge is detected. This also generates an asynchronous wake-up signal to exit the low-power mode.
- If the pin is pinmuxed as NMI pin, it generates the non-maskable interrupt request rather than external interrupt request when the external NMI signal is low.

7.4.5 Digital filter

When the PORT module is enabled, its digital filter capabilities are available in all digital pin-multiplexing modes.

- The PARCC_PORTx can configure the clock that is used for all digital filters within one port.
- You must change this selection only when all digital filters for that port are disabled.
- While the digital filters are bypassed, the output of each digital filter is always equal to the input pin. However, the internal state of the digital filters remains static and does not get updated due to any change on the input pin.

The output of each digital filter is logic zero after system reset and whenever a digital filter is disabled.

- After a digital filter is enabled, the input is synchronized to the filter clock.
- If the synchronized input and the output of the digital filter remain different for several filter clock cycles that are equal to PORTx_PCRn[FLT], then the output of the digital filter is updated to equal the synchronized filter input.
- The maximum latency through a digital filter is equal to three filter clock cycles plus PORTx_PCRn[FLT].

Chapter 8

General-Purpose Input/Output (GPIO)

8.1 Introduction

The general-purpose input and output (GPIO) module communicates with the CPU via an AHB interface for maximum pin performance. The GPIO registers support 8-bit, 16-bit, or 32-bit accesses.

The GPIO data direction and output data registers control the direction and output data of each pin when the pin is configured for the GPIO function, only if the corresponding Port Control and Interrupt module for that pin is enabled.

Through the set, clear, and toggle write-only registers of each port output data register, the general-purpose outputs can have efficient bit manipulation.

8.1.1 Features

The GPIO module has the following features:

- Port Data Input register is visible in all digital pin-multiplexing modes
- Port Data Output register has corresponding set/clear/toggle registers
- Port Data Direction register
- The GPIO module is clocked by the CPU clock for maximum pin performance

8.2 Signal description

Table 8-1 GPIO signal descriptions

GPIO signal descriptions	Description	I/O
PTA31-0	General-purpose input/output	I/O
PTB31-0	General-purpose input/output	I/O
PTC31-0	General-purpose input/output	I/O
PTD31-0	General-purpose input/output	I/O
PTE31-0	General-purpose input/output	I/O

Not all pins within each port are implemented on each device. See the chapter on signal multiplexing for the number of GPIO ports available in the device.

Table 8-2 GPIO interface-detailed signal descriptions

Signal	I/O	Description	
PTA31-0 PTB31-0 PTC31-0 PTD31-0 PTE31-0	I/O	General-purpose input/output	
		State meaning	Asserted: The pin is logic 1.
			Deasserted: The pin is logic 0.
		Timing	Assertion: When configured as an output, this signal occurs on the rising edge of the system clock. When configured as an input, this signal may occur at any time and input may be asserted asynchronously to the system clock. Deassertion: When configured as an

Signal	I/O	Description
		output, this signal occurs on the rising edge of the system clock. When configured as input, the signal may occur at any time and input may be asserted asynchronously to the system clock.

8.3 Memory map and register definition

This section provides the memory map and register definitions of the GPIO module.

8.3.1 Memory map

8.3.1.1 GPIO ports memory map

Table 8-3 GPIO ports memory map

Start address	Port
Base + 0h	GPIO A
Base + 40h	GPIO B
Base + 80h	GPIO C
Base + C0h	GPIO D
Base + 100h	GPIO E

This version of GPIO implements five instances of GPIOA to GPIOE. The GPIO register descriptions in this chapter are generic. Regarding how addresses are allocated to every instance of GPIO in the device memory map, see the above table.

In Z20K14xMC MCU, GPIO is multi-ported and can be accessed directly by the core through system bus at $[0x400F\_F000 + 0x40*n]$ (n: 0 to 4). When in GPIO mode (PORTx_PCRn[MUX] = 001b), GPIO function is selected for this pin.

8.3.1.2 GPIO Memory map

GPIO base address: 0x400FF000

Table 8-4 GPIO Memory map

Offset	Register	Width (In bits)	Reset Value
0x0	Port Data Output Register (GPIOA_PDOR)	32	0x00000000
0x4	Port Set Output Register (GPIOA_PSOR)	32	0x00000000
0x8	Port Clear Output Register (GPIOA_PCOR)	32	0x00000000
0xC	Port Toggle Output Register (GPIOA_PTOR)	32	0x00000000
0x10	Port Data Input Register (GPIOA_PDIR)	32	0x00000000
0x14	Port Data Direction Register (GPIOA_PDDR)	32	0x00000000
0x40	Port Data Output Register (GPIOB_PDOR)	32	0x00000000
0x44	Port Set Output Register (GPIOB_PSOR)	32	0x00000000

Offset	Register	Width (In bits)	Reset Value
0x48	Port Clear Output Register (GPIOB_PCOR)	32	0x00000000
0x4C	Port Toggle Output Register (GPIOB_PTOR)	32	0x00000000
0x50	Port Data Input Register (GPIOB_PDIR)	32	0x00000000
0x54	Port Data Direction Register (GPIOB_PDDR)	32	0x00000000
0x80	Port Data Output Register (GPIOC_PDOR)	32	0x00000000
0x84	Port Set Output Register (GPIOC_PSOR)	32	0x00000000
0x88	Port Clear Output Register (GPIOC_PCOR)	32	0x00000000
0x8C	Port Toggle Output Register (GPIOC_PTOR)	32	0x00000000
0x90	Port Data Input Register (GPIOC_PDIR)	32	0x00000000
0x94	Port Data Direction Register (GPIOC_PDDR)	32	0x00000000
0xC0	Port Data Output Register (GPIOD_PDOR)	32	0x00000000
0xC4	Port Set Output Register (GPIOD_PSOR)	32	0x00000000
0xC8	Port Clear Output Register (GPIOD_PCOR)	32	0x00000000
0xCC	Port Toggle Output Register (GPIOD_PTOR)	32	0x00000000
0xD0	Port Data Input Register (GPIOD_PDIR)	32	0x00000000
0xD4	Port Data Direction Register (GPIOD_PDDR)	32	0x00000000
0x100	Port Data Output Register (GPIOE_PDOR)	32	0x00000000
0x104	Port Set Output Register (GPIOE_PSOR)	32	0x00000000
0x108	Port Clear Output Register (GPIOE_PCOR)	32	0x00000000
0x10C	Port Toggle Output Register (GPIOE_PTOR)	32	0x00000000
0x110	Port Data Input Register (GPIOE_PDIR)	32	0x00000000
0x114	Port Data Direction Register (GPIOE_PDDR)	32	0x00000000

8.3.2 Port Data Output Register (GPIOx_PDOR)

8.3.2.1 Offset

Register	Offset
GPIOx_PDOR	0x0 + 0x40 * x, When the user selects port A, x is equal to 0 When the user selects port B, x is equal to 1

Register	Offset
	When the user selects port C, x is equal to 2 When the user selects port D, x is equal to 3 When the user selects port E, x is equal to 4

8.3.2.2 Function

This register configures the logic levels that are driven on each general-purpose output pin.

8.3.2.3 Fields

Field	Access	Description	Reset Value
31:0 PDO	RW	0b: Logic level 0 is driven from GPIO when the pin is configured to the output pin. 1b: Logic level 1 is driven from GPIO when the pin is configured to the output pin.	0x0

8.3.3 Port Set Output Register (GPIOx_PSOR)

8.3.3.1 Offset

Register	Offset
GPIOx_PSOR	0x4 + 0x40 * x, When the user selects port A, x is equal to 0 When the user selects port B, x is equal to 1 When the user selects port C, x is equal to 2 When the user selects port D, x is equal to 3 When the user selects port E, x is equal to 4

8.3.3.2 Function

This register configures whether to set the fields of the PDOR.

8.3.3.3 Fields

Field	Access	Description	Reset Value
31:0 PSO	WO	Writing to this register will update the contents of the corresponding bit in the PDOR as follows: 0b: Corresponding bit in PDORn does not change. 1b: Corresponding bit in PDORn is set to logic 1.	0x0

8.3.4 Port Clear Output Register (GPIOx_PCOR)

8.3.4.1 Offset

Register	Offset
GPIOx_PCOR	0x8 + 0x40 * x, When the user selects port A, x is equal to 0 When the user selects port B, x is equal to 1 When the user selects port C, x is equal to 2 When the user selects port D, x is equal to 3

Register	Offset
	When the user selects port E, x is equal to 4

8.3.4.2 Function

This register configures whether to set the fields of the PDOR.

8.3.4.3 Fields

Field	Access	Description	Reset Value
31:0 PCO	WO	Writing to this register will update the contents of the corresponding bit in the PDOR as follows: 0b: Corresponding bit in PDORn does not change. 1b: Corresponding bit in PDORn is clear to logic 0.	0x0

8.3.5 Port Toggle Output Register (GPIOx_PTOR)

8.3.5.1 Offset

Register	Offset
GPIOx_PTOR	0xC + 0x40 * x, When the user selects port A, x is equal to 0 When the user selects port B, x is equal to 1 When the user selects port C, x is equal to 2 When the user selects port D, x is equal to 3 When the user selects port E, x is equal to 4

8.3.5.2 Function

This register configures whether to set the fields of the PDOR.

8.3.5.3 Fields

Field	Access	Description	Reset Value
31:0 PTO	WO	Writing to this register will update the contents of the corresponding bit in the PDOR as follows: 0b: Corresponding bit in PDORn does not change. 1b: Corresponding bit in PDORn is set to the inverse of its existing logic state.	0x0

8.3.6 Port Data Input Register (GPIOx_PDIR)

8.3.6.1 Offset

Register	Offset
GPIOx_PDIR	0x10 + 0x40 * x, When the user selects port A, x is equal to 0 When the user selects port B, x is equal to 1 When the user selects port C, x is equal to 2 When the user selects port D, x is equal to 3 When the user selects port E, x is equal to 4

8.3.6.2 Function

This register reads the logic level of the pin.

8.3.6.3 Fields

Field	Access	Description	Reset Value
31:0 PDI	RO	Pins that are not configured for a digital function read 0. 0b: Pin logic level is logic 0, or is not configured for use by digital function. 1b: Pin logic level is logic 1.	0x0

8.3.7 Port Data Direction Register (GPIOx_PDDR)

8.3.7.1 Offset

Register	Offset
GPIOx_PDDR	0x14 + 0x40 * x, When the user selects port A, x is equal to 0 When the user selects port B, x is equal to 1 When the user selects port C, x is equal to 2 When the user selects port D, x is equal to 3 When the user selects port E, x is equal to 4

8.3.7.2 Function

The PDDR configures the individual port pins for input or output.

8.3.7.3 Fields

Field	Access	Description	Reset Value
31:0 PDD	RW	Configures individual port pins for input or output. 0b: Pin is configured as general-purpose input, for the GPIO function. 1b: Pin is configured as general-purpose output, for the GPIO function.	0x0

8.4 Function description

8.4.1 General-purpose input

The logic state of each pin is available via the Port Data Input registers when the pin is configured for a digital function, and the corresponding Port Control and Interrupt module is enabled.

Because the input pin synchronizers are shared with the Port Control and Interrupt module, whenever the corresponding Port Control and Interrupt module is disabled, synchronizers are also disabled.

When a port is not required for general-purpose input functionality, it reduces the power consumption.

8.4.2 General-purpose output

If the pin is configured for the GPIO function, the logic state of each pin can be controlled via the port data output registers and port data direction register. The following table details the conditions for a pin to be configured as input/output.

Table 8-5 I/O configuration

Condition	Consequence
A pin is configured for the GPIO function and the corresponding port data direction register bit is clear	The pin is configured as an input
A pin is configured for the GPIO function and the corresponding port data direction register bit is set	The pin is configured as an output and the logic state of the pin is equal to the corresponding port data output register

The pin data set, pin data clear, and pin data toggle registers exist to allow one or more outputs within one port to be set, cleared, or toggled from a single register write. These ensure an efficient bit manipulation on the general-purpose outputs.

The corresponding Port Control and Interrupt module is not required to be enabled to update the state of the port data direction registers and port data output registers including the set, clear, and toggle registers.

Chapter 9

Direct Memory Access Controller (DMA)

9.1 Introduction

The Direct Memory Access Controller (DMA) is used to provide high-speed data transfer between peripherals and memory. Data can be quickly moved by DMA without any CPU actions. The DMA contains 16 independent channels to service DMA request.

9.1.1 Features

The DMA supports the following key features:

- 16 independent channels to service DMA request
 - Each channel can service software DMA request or hardware DMA request
 - Each channel completes data transfer via dual-address: source address for data read and destination address for data write
 - Source address offset and destination address offset of each channel in a minor loop are programmable
 - Source address offset and destination address offset of each channel between two major loops are programmable
 - Read transfer size and write transfer size of each channel are programmable
 - The number of transfer data of each channel in a minor loop is programmable
 - The iteration number of each channel is programmable
 - Each channel can generate DMA transfer done interrupt or transfer error interrupt
 - Round-robin channel priority arbitration and fixed channel priority arbitration are supported, and each channel priority is programmable in fixed priority mode

9.1.2 Modes of operation

The DMA operates in the following modes:

- **DEBUG mode:**
 - DMA operation is configurable in DEBUG mode by DMA_CONF register:
 - If DMA_CONF[DBGE] is cleared, the DMA continues to data transfer
 - If DMA_CONF[DBGE] is set, the DMA will halt after completing the executing channel data transfer, and will resume executing the remained channel after exiting from debug mode
- **WAIT mode:**
 - DMA is enabled in WAIT mode
- **STOP mode:**
 - DMA is disabled in STOP mode

9.1.3 Block diagram

The following figure is a block diagram of the Direct Memory Access Controller (DMA) module.

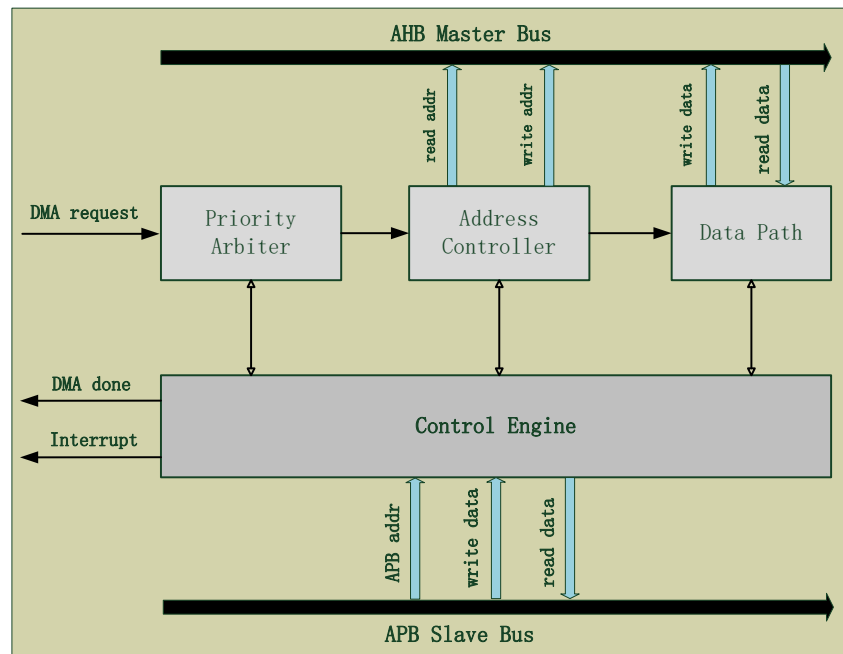


Figure 9-1 DMA block diagram

9.2 Memory map and register definition

This section provides the memory map and register definitions of DMA module. The DMA's registers can be divided into two categories:

- The first provides global control functions, including DMA request enable, interrupt mask, channel priority configuration and so on
- The second corresponds to program the data transfer read and write control for every DMA channel

9.2.1 Memory map

DMA base address: 0x40008000h

Table 9-1 DMA memory map

Offset	Register	Width (In bits)	Reset Value
0x00	DMA Configure Register (DMA_CONF)	32	0x00000000
0x04	DMA Request Enable Register (DMA_DMAE)	32	0x00000000
0x0C	DMA Error Interrupt Enable Register (DMA_EIE)	32	0x00000000
0x14	DMA Global Channel Control Register (DMA_GCC)	32	0x00000000
0x18	DMA Error Status Register (DMA_ES)	32	0x00000000
0x1C	DMA Channel Error Status Register (DMA_CES)	32	0x00000000
0x20	DMA Channel Interrupt Status Register (DMA_CIS)	32	0x00000000
0x24	DMA Channel Hardware Request Status Register (DMA_CHRS)	32	0x00000000

Offset	Register	Width (In bits)	Reset Value
0x28	DMA Channel Priority Register 0 (DMA_CPRI0)	32	0x00010203
0x2C	DMA Channel Priority Register 1 (DMA_CPRI1)	32	0x04050607
0x30	DMA Channel Priority Register 2 (DMA_CPRI2)	32	0x08090A0B
0x34	DMA Channel Priority Register 3 (DMA_CPRI3)	32	0x0C0D0E0F
0x200 + 0x20*n (n=0~15)	DMA Channel n Source Address Register (DMA_SADDRn)	32	0x00000000
0x204 + 0x20*n (n=0~15)	DMA Channel n Destination Address Register (DMA_DADDRn)	32	0x00000000
0x208 + 0x20*n (n=0~15)	DMA Channel n Address Offset Register (DMA_AOFFn)	32	0x00000000
0x20C + 0x20*n (n=0~15)	DMA Channel n Major Loop Address Offset Register (DMA_MLAOFFn)	32	0x00000000
0x210 + 0x20*n (n=0~15)	DMA Channel n Number of Transfer Data (DMA_NUMn)	32	0x00000000
0x214 + 0x20*n (n=0~15)	DMA Channel n Iteration Number Register (DMA_ITERn)	32	0x00010001
0x218 + 0x20*n (n=0~15)	DMA Channel n Control and Status Register (DMA_CSn)	32	0x00000000

9.2.2 DMA Configure Register (DMA_CONF)

9.2.2.1 Offset

Register	Offset
DMA_CONF	0x00

9.2.2.2 Function

This register defines the basic operating configuration of DMA, including DMA priority arbitration, DMA behavior in debug mode and when error occurs.

9.2.2.3 Fields

Field	Access	Description	Reset Value
0 HALT	RW	0b: DMA does not halt. 1b: DMA stalls the start of any new channels. The executing channel data transfer is allowed to complete. Channel execution will resume after this bit is cleared. NOTE This bit is either software set by writing 1, or hardware set when an error occurs and DMA_CONF[HOEE] =1. It is either software cleared by writing 0, or hardware cleared when error status was cleared.	0x0
1 HOEE	RW	0b: DMA will not halt when an error occurs. 1b: When an error occurs, halt bit will assert and DMA operation will halt until the halt bit was cleared.	0x0

Field	Access	Description	Reset Value
2 RRCAE	RW	0b: Channel uses fixed priority arbitration. 1b: Channel uses round robin arbitration.	0x0
3 DBGE	RW	0b: When in debug mode, DMA continues to transfer data. 1b: When in debug mode, DMA will halt after completing the executing channel data transfer, and will resume executing the remained channel after exiting from the debug mode.	0x0
5:4 RSVD_5_4	RO	Reserved.	0x0
6 BUSY	RO	0b: DMA is idle. 1b: DMA is busy and a channel event is executing.	0x0
31:7 RSVD_31_7	RO	Reserved	0x0

9.2.3 DMA Request Enable Register (DMA_DMAE)

9.2.3.1 Offset

Register	Offset
DMA_DMAE	0x04

9.2.3.2 Function

This register provides the enable control for the DMA request of each channel. The hardware DMA request is only serviced when the related channel DMAE bit is enabled and before DMA request signal asserts.

9.2.3.3 Fields

Field	Access	Description	Reset Value
0 DMAE0	RW	0b: Channel 0 DMA request is ignored. 1b: Channel 0 DMA request is enabled to service. NOTE: If DMA_CS0[REQDIS] is set, after channel 0 DMA transfer done, this bit will be automatically cleared by hardware.	0x0
1 DMAE1	RW	0b: Channel 1 DMA request is ignored. 1b: Channel 1 DMA request is enabled to service. NOTE: If DMA_CS1[REQDIS] is set, after channel 1 DMA transfer done, this bit will be automatically cleared by hardware.	0x0
2 DMAE2	RW	0b: Channel 2 DMA request is ignored. 1b: Channel 2 DMA request is enabled to service. NOTE: If DMA_CS2[REQDIS] is set, after channel 2 DMA transfer done, this bit will be automatically cleared by hardware.	0x0
3 DMAE3	RW	0b: Channel 3 DMA request is ignored. 1b: Channel 3 DMA request is enabled to service. NOTE: If DMA_CS3[REQDIS] is set, after channel 3 DMA transfer done, this bit will be automatically cleared by hardware.	0x0
4 DMAE4	RW	0b: Channel 4 DMA request is ignored. 1b: Channel 4 DMA request is enabled to service. NOTE: If DMA_CS4[REQDIS] is set, after channel 4	0x0

Field	Access	Description	Reset Value
		DMA transfer done, this bit will be automatically cleared by hardware.	
5 DMAE5	RW	0b: Channel 5 DMA request is ignored. 1b: Channel 5 DMA request is enabled to service. NOTE: If DMA_CS5[REQDIS] is set, after channel 5 DMA transfer done, this bit will be automatically cleared by hardware.	0x0
6 DMAE6	RW	0b: Channel 6 DMA request is ignored. 1b: Channel 6 DMA request is enabled to service. NOTE: If DMA_CS6[REQDIS] is set, after channel 6 DMA transfer done, this bit will be automatically cleared by hardware.	0x0
7 DMAE7	RW	0b: Channel 7 DMA request is ignored. 1b: Channel 7 DMA request is enabled to service. NOTE: If DMA_CS7[REQDIS] is set, after channel 7 DMA transfer done, this bit will be automatically cleared by hardware.	0x0
8 DMAE8	RW	0b: Channel 8 DMA request is ignored. 1b: Channel 8 DMA request is enabled to service. NOTE: If DMA_CS8[REQDIS] is set, after channel 8 DMA transfer done, this bit will be automatically cleared by hardware.	0x0
9 DMAE9	RW	0b: Channel 9 DMA request is ignored. 1b: Channel 9 DMA request is enabled to service. NOTE: If DMA_CS9[REQDIS] is set, after channel 9 DMA transfer done, this bit will be automatically cleared by hardware.	0x0
10 DMAE10	RW	0b: Channel 10 DMA request is ignored. 1b: Channel 10 DMA request is enabled to service. NOTE: If DMA_CS10[REQDIS] is set, after channel 10 DMA transfer done, this bit will be automatically cleared by hardware.	0x0
11 DMAE11	RW	0b: Channel 11 DMA request is ignored. 1b: Channel 11 DMA request is enabled to service. NOTE: If DMA_CS11[REQDIS] is set, after channel 11 DMA transfer done, this bit will be automatically cleared by hardware.	0x0
12 DMAE12	RW	0b: Channel 12 DMA request is ignored. 1b: Channel 12 DMA request is enabled to service. NOTE: If DMA_CS12[REQDIS] is set, after channel 12 DMA transfer done, this bit will be automatically cleared by hardware.	0x0
13 DMAE13	RW	0b: Channel 13 DMA request is ignored. 1b: Channel 13 DMA request is enabled to service. NOTE: If DMA_CS13[REQDIS] is set, after channel 13 DMA transfer done, this bit will be automatically cleared by hardware.	0x0
14 DMAE14	RW	0b: Channel 14 DMA request is ignored. 1b: Channel 14 DMA request is enabled to service. NOTE: If DMA_CS14[REQDIS] is set, after channel 14 DMA transfer done, this bit will be automatically cleared by hardware.	0x0
15 DMAE15	RW	0b: Channel 15 DMA request is ignored. 1b: Channel 15 DMA request is enabled to service. NOTE: If DMA_CS15[REQDIS] is set, after channel 15 DMA transfer done, this bit will be automatically cleared	0x0

Field	Access	Description	Reset Value
		by hardware.	
31:16 RSVD_31_16	RO	Reserved	0x0

9.2.4 DMA Error Interrupt Enable Register (DMA_EIE)

9.2.4.1 Offset

Register	Offset
DMA_EIE	0x0C

9.2.4.2 Function

This register provides the enable control for the interrupt generation of each channel when a channel error occurs.

9.2.4.3 Fields

Field	Access	Description	Reset Value
0 EIE0	RW	0b: Disable channel 0 error interrupt generation. 1b: Enable channel 0 error interrupt generation.	0x0
1 EIE1	RW	0b: Disable channel 1 error interrupt generation. 1b: Enable channel 1 error interrupt generation.	0x0
2 EIE2	RW	0b: Disable channel 2 error interrupt generation. 1b: Enable channel 2 error interrupt generation.	0x0
3 EIE3	RW	0b: Disable channel 3 error interrupt generation. 1b: Enable channel 3 error interrupt generation.	0x0
4 EIE4	RW	0b: Disable channel 4 error interrupt generation. 1b: Enable channel 4 error interrupt generation.	0x0
5 EIE5	RW	0b: Disable channel 5 error interrupt generation. 1b: Enable channel 5 error interrupt generation.	0x0
6 EIE6	RW	0b: Disable channel 6 error interrupt generation. 1b: Enable channel 6 error interrupt generation.	0x0
7 EIE7	RW	0b: Disable channel 7 error interrupt generation. 1b: Enable channel 7 error interrupt generation.	0x0
8 EIE8	RW	0b: Disable channel 8 error interrupt generation. 1b: Enable channel 8 error interrupt generation.	0x0
9 EIE9	RW	0b: Disable channel 9 error interrupt generation. 1b: Enable channel 9 error interrupt generation.	0x0
10 EIE10	RW	0b: Disable channel 10 error interrupt generation. 1b: Enable channel 10 error interrupt generation.	0x0
11 EIE11	RW	0b: Disable channel 11 error interrupt generation. 1b: Enable channel 11 error interrupt generation.	0x0
12 EIE12	RW	0b: Disable channel 12 error interrupt generation. 1b: Enable channel 12 error interrupt generation.	0x0
13 EIE13	RW	0b: Disable channel 13 error interrupt generation. 1b: Enable channel 13 error interrupt generation.	0x0
14 EIE14	RW	0b: Disable channel 14 error interrupt generation. 1b: Enable channel 14 error interrupt generation.	0x0
15 EIE15	RW	0b: Disable channel 15 error interrupt generation. 1b: Enable channel 15 error interrupt generation.	0x0
31:16 RSVD_31_16	RO	Reserved	0x0

9.2.5 DMA Global Channel Control Register (DMA_GCC)

9.2.5.1 Offset

Register	Offset
DMA_GCC	0x14

9.2.5.2 Function

This register provides global control to clear channels' status in CES/CIS registers, set channels' START bit to trigger software DMA request and clear DMA channel done flag. This register is write only. Reading this register will always return value 0.

9.2.5.3 Fields

Field	Access	Description	Reset Value
3:0 CCDONE	WO	This field defines which channel of the DONE bit is clear. 0000b: Clear DMA_CS0[DONE] bit. 0001b: Clear DMA_CS1[DONE] bit. 0010b: Clear DMA_CS2[DONE] bit. 0011b: Clear DMA_CS3[DONE] bit. 0100b: Clear DMA_CS4[DONE] bit. 0101b: Clear DMA_CS5[DONE] bit. 0110b: Clear DMA_CS6[DONE] bit. 0111b: Clear DMA_CS7[DONE] bit. 1000b: Clear DMA_CS8[DONE] bit. 1001b: Clear DMA_CS9[DONE] bit. 1010b: Clear DMA_CS10[DONE] bit. 1011b: Clear DMA_CS11[DONE] bit. 1100b: Clear DMA_CS12[DONE] bit. 1101b: Clear DMA_CS13[DONE] bit. 1110b: Clear DMA_CS14[DONE] bit. 1111b: Clear DMA_CS15[DONE] bit.	0x0
5:4 RSVD_5_4	RO	Reserved	0x0
6 CACDONE	WO	This bit defines whether all channels' DONE bits are cleared or not. 0b: Only clear one channel's DONE bit according to CCDONE configuration. 1b: Clear all channels' DONE bits of DMA_CS _n [DONE].	0x0
7 WPEN0	WO	Similar with WPEN3. This bit allows this register byte 0 bits to be write protected. When a value is written to DMA_GCC register, if the written value bit [7] is 1, writing to bit [6:0] will be ignored; if the written value bit [7] is 0, writing to bit [6:0] will make sense. Reading this bit will always return 0.	0x0
11:8 SCSTART	WO	This field defines which channel of the START bit is set. 0000b: Set DMA_CS0[START] bit. 0001b: Set DMA_CS1[START] bit. 0010b: Set DMA_CS2[START] bit. 0011b: Set DMA_CS3[START] bit. 0100b: Set DMA_CS4[START] bit.	0x0

Field	Access	Description	Reset Value
		0101b: Set DMA_CS5[START] bit. 0110b: Set DMA_CS6[START] bit. 0111b: Set DMA_CS7[START] bit. 1000b: Set DMA_CS8[START] bit. 1001b: Set DMA_CS9[START] bit. 1010b: Set DMA_CS10[START] bit. 1011b: Set DMA_CS11[START] bit. 1100b: Set DMA_CS12[START] bit. 1101b: Set DMA_CS13[START] bit. 1110b: Set DMA_CS14[START] bit. 1111b: Set DMA_CS15[START] bit.	
13:12 RSVD_13_12	RO	Reserved	0x0
14 SACSTART	WO	This bit defines whether all channels' START bits are set or not. 0b: Only set one channel's START bit according to SCSTART configuration. 1b: Set all channels' START bits of DMA_CS _n [START].	0x0
15 WPEN1	WO	Similar with WPEN3. This bit allows this register byte 1 bits to be write protected. When a value is written to DMA_GCC register, if the written value bit [15] is 1, writing to bit [14:8] will be ignored; if the written value bit [15] is 0, writing to bit [14:8] will make sense. Reading this bit will always return 0.	0x0
19:16 CCES	WO	This field defines which channel error status is cleared. 0000b: Clear ERR0 bit in CES register. 0001b: Clear ERR1 bit in CES register. 0010b: Clear ERR2 bit in CES register. 0011b: Clear ERR3 bit in CES register. 0100b: Clear ERR4 bit in CES register. 0101b: Clear ERR5 bit in CES register. 0110b: Clear ERR6 bit in CES register. 0111b: Clear ERR7 bit in CES register. 1000b: Clear ERR8 bit in CES register. 1001b: Clear ERR9 bit in CES register. 1010b: Clear ERR10 bit in CES register. 1011b: Clear ERR11 bit in CES register. 1100b: Clear ERR12 bit in CES register. 1101b: Clear ERR13 bit in CES register. 1110b: Clear ERR14 bit in CES register. 1111b: Clear ERR15 bit in CES register.	0x0
21:20 RSVD_21_20	RO	Reserved	0x0
22 CACES	WO	This bit defines whether all channels' error status is cleared or not. 0b: Only clear one ERR bit in CES register according to CCES configuration. 1b: Clear all channels' ERR bits in CES register.	0x0
23 WPEN2	WO	Similar with WPEN3. This bit allows this register byte 2 bits to be write protected. When a value is written to DMA_GCC register, if the written value bit [23] is 1, writing to bit [22:16] will be ignored. If the written value bit [23] is 0, writing to bit [22:16] will make sense. Reading this bit will always return 0.	0x0

Field	Access	Description	Reset Value
27:24 CCIS	WO	This field defines which channel interrupt status is cleared. 0000b: Clear INT0 bit in CIS register. 0001b: Clear INT1 bit in CIS register. 0010b: Clear INT2 bit in CIS register. 0011b: Clear INT3 bit in CIS register. 0100b: Clear INT4 bit in CIS register. 0101b: Clear INT5 bit in CIS register. 0110b: Clear INT6 bit in CIS register. 0111b: Clear INT7 bit in CIS register. 1000b: Clear INT8 bit in CIS register. 1001b: Clear INT9 bit in CIS register. 1010b: Clear INT10 bit in CIS register. 1011b: Clear INT11 bit in CIS register. 1100b: Clear INT12 bit in CIS register. 1101b: Clear INT13 bit in CIS register. 1110b: Clear INT14 bit in CIS register. 1111b: Clear INT15 bit in CIS register.	0x0
29:28 RSVD_29_28	RO	Reserved	0x0
30 CACIS	WO	This bit defines whether all channels' interrupt status is cleared or not. 0b: Only clear one INT bit in CIS register according to CCIS configuration. 1b: Clear all channels' INT bits in CIS register.	0x0
31 WPEN3	WO	This bit allows this register byte 3 bits to be write protected. When a value is written to DMA_GCC register, if the written value bit [31] is 1, writing to bit [30:24] will be ignored. If the written value bit [31] is 0, writing to bit [30:24] will make sense. Reading this bit will always return 0. This write protection mechanism allows the user to write 4-byte words to DMA_GCC register but protecting other bytes' fields. For example, if the user wants to configure CCIS to value 3, the user can write 0x03808080 to protect byte 0/1/2.	0x0

9.2.6 DMA Error Status Register (DMA_ES)

9.2.6.1 Offset

Register	Offset
DMA_ES	0x18

9.2.6.2 Function

The register provides information concerning the last recorded channel error, such as priority setting error, unaligned address error, and error channel ID.

9.2.6.3 Fields

Field	Access	Description	Reset Value
0 DBEF	RO	Read only bit. When destination address is invalid, a destination bus error occurs.	0x0

Field	Access	Description	Reset Value
		0b: No destination bus error. 1b: Bus error occurs on a destination write.	
1 SBEF	RO	Read only bit. When source address is invalid, source bus error occurs. 0b: No source bus error. 1b: Bus error occurs on a source read.	0x0
2 RSVD_2	RO	Reserved	0x0
3 NUMEF	RO	Read only bit. When number of transfer data is not multiple of SSIZE or DSIZE, number of transfer data error occur. 0b: No number of transfer data error. 1b: Number of transfer data configuration error.	0x0
4 DOEF	RO	Read only bit. When configured DOFF is not matched with DSIZE, destination address offset error occurs. 0b: No destination address offset error. 1b: Destination address offset configuration error.	0x0
5 DAEF	RO	Read only bit. When configured DADDR is not matched with DSIZE, destination address error occurs. 0b: No destination address error. 1b: Destination address configuration error.	0x0
6 SOEF	RO	Read only bit. When configured SOFF is not matched with SSIZE, source address offset error occurs. 0b: No source address offset error. 1b: Source address offset configuration error.	0x0
7 SAEF	RO	Read only bit. When configured SADDR is not matched with SSIZE, source address error occurs. 0b: No source address error. 1b: Source address configuration error.	0x0
11:8 CHEF	RO	Read only bit. This field records the last error channel number.	0x0
12 RSVD_12	RO	Reserved	0x0
13 PRIEF	RO	Read only bit. Every channel priority value should be unique, otherwise priority error occurs. 0b: No channel priority error. 1b: The last recorded error was channel priority configuration error.	0x0
14 RSVD_14	RO	Reserved	0x0
15 ERRF	RO	Read only bit. This bit is logic OR of all error status bits. 0b: No error status bit is set. 1b: At least one error status bit is set.	0x0
31:16 RSVD_31_16	RO	Reserved	0x0

9.2.7 DMA Channel Error Status Register (DMA_CES)

9.2.7.1 Offset

Register	Offset
DMA_CES	0x1C

9.2.7.2 Function

This register provides each channel error status. Every bit also can be cleared by writing the corresponding value to CACES or CCES fields of DMA_GCC register.

9.2.7.3 Fields

Field	Access	Description	Reset Value
0 ERR0	W1C	0b: No error occurred in channel 0 or the error occurred has been cleared. 1b: An error occurred in channel 0.	0x0
1 ERR1	W1C	0b: No error occurred in channel 1 or the error occurred has been cleared. 1b: An error occurred in channel 1.	0x0
2 ERR2	W1C	0b: No error occurred in channel 2 or the error occurred has been cleared. 1b: An error occurred in channel 2.	0x0
3 ERR3	W1C	0b: No error occurred in channel 3 or the error occurred has been cleared. 1b: An error occurred in channel 3.	0x0
4 ERR4	W1C	0b: No error occurred in channel 4 or the error occurred has been cleared. 1b: An error occurred in channel 4.	0x0
5 ERR5	W1C	0b: No error occurred in channel 5 or the error occurred has been cleared. 1b: An error occurred in channel 5.	0x0
6 ERR6	W1C	0b: No error occurred in channel 6 or the error occurred has been cleared. 1b: An error occurred in channel 6.	0x0
7 ERR7	W1C	0b: No error occurred in channel 7 or the error occurred has been cleared. 1b: An error occurred in channel 7.	0x0
8 ERR8	W1C	0b: No error occurred in channel 8 or the error occurred has been cleared. 1b: An error occurred in channel 8.	0x0
9 ERR9	W1C	0b: No error occurred in channel 9 or the error occurred has been cleared. 1b: An error occurred in channel 9.	0x0
10 ERR10	W1C	0b: No error occurred in channel 10 or the error occurred has been cleared. 1b: An error occurred in channel 10.	0x0
11 ERR11	W1C	0b: No error occurred in channel 11 or the error occurred has been cleared. 1b: An error occurred in channel 11.	0x0
12 ERR12	W1C	0b: No error occurred in channel 12 or the error occurred has been cleared. 1b: An error occurred in channel 12.	0x0
13 ERR13	W1C	0b: No error occurred in channel 13 or the error occurred has been cleared. 1b: An error occurred in channel 13.	0x0
14 ERR14	W1C	0b: No error occurred in channel 14 or the error occurred has been cleared. 1b: An error occurred in channel 14.	0x0
15 ERR15	W1C	0b: No error occurred in channel 15 or the error occurred has been cleared. 1b: An error occurred in channel 15.	0x0

Field	Access	Description	Reset Value
31:16 RSVD_31_16	RO	Reserved	0x0

9.2.8 DMA Channel Interrupt Status Register (DMA_CIS)

9.2.8.1 Offset

Register	Offset
DMA_CIS	0x20

9.2.8.2 Function

This register provides the interrupt status of each channel. Every bit also can be cleared by writing the corresponding value to the CACIS or CCIS fields of the DMA_GCC register.

9.2.8.3 Fields

Field	Access	Description	Reset Value
0 INT0	W1C	0b: No interrupt request generated in channel 0 or the interrupt status has been cleared. 1b: An interrupt request generated in channel 0.	0x0
1 INT1	W1C	0b: No interrupt request generated in channel 1 or the interrupt status has been cleared. 1b: An interrupt request generated in channel 1.	0x0
2 INT2	W1C	0b: No interrupt request generated in channel 2 or the interrupt status has been cleared. 1b: An interrupt request generated in channel 2.	0x0
3 INT3	W1C	0b: No interrupt request generated in channel 3 or the interrupt status has been cleared. 1b: An interrupt request generated in channel 3.	0x0
4 INT4	W1C	0b: No interrupt request generated in channel 4 or the interrupt status has been cleared. 1b: An interrupt request generated in channel 4.	0x0
5 INT5	W1C	0b: No interrupt request generated in channel 5 or the interrupt status has been cleared. 1b: An interrupt request generated in channel 5.	0x0
6 INT6	W1C	0b: No interrupt request generated in channel 6 or the interrupt status has been cleared. 1b: An interrupt request generated in channel 6.	0x0
7 INT7	W1C	0b: No interrupt request generated in channel 7 or the interrupt status has been cleared. 1b: An interrupt request generated in channel 7.	0x0
8 INT8	W1C	0b: No interrupt request generated in channel 8 or the interrupt status has been cleared. 1b: An interrupt request generated in channel 8.	0x0
9 INT9	W1C	0b: No interrupt request generated in channel 9 or the interrupt status has been cleared. 1b: An interrupt request generated in channel 9.	0x0
10 INT10	W1C	0b: No interrupt request generated in channel 10 or the interrupt status has been cleared. 1b: An interrupt request generated in channel 10.	0x0
11 INT11	W1C	0b: No interrupt request generated in channel 11 or the interrupt status has been cleared. 1b: An interrupt request generated in channel 11.	0x0

Field	Access	Description	Reset Value
12 INT12	W1C	0b: No interrupt request generated in channel 12 or the interrupt status has been cleared. 1b: An interrupt request generated in channel 12.	0x0
13 INT13	W1C	0b: No interrupt request generated in channel 13 or the interrupt status has been cleared. 1b: An interrupt request generated in channel 13.	0x0
14 INT14	W1C	0b: No interrupt request generated in channel 14 or the interrupt status has been cleared. 1b: An interrupt request generated in channel 14.	0x0
15 INT15	W1C	0b: No interrupt request generated in channel 15 or the interrupt status has been cleared. 1b: An interrupt request generated in channel 15.	0x0
31:16 RSVD_31_16	RO	Reserved	0x0

9.2.9 DMA Channel Hardware Request Status Register (DMA_CHRS)

9.2.9.1 Offset

Register	Offset
DMA_CHRS	0x24

9.2.9.2 Function

This register provides the hardware DMA request status for each channel. When a channel hardware DMA request is asserted, the corresponding HRS bit is set. After the service for the hardware request is done and the channel is free, the HRS bit is automatically cleared by hardware.

9.2.9.3 Fields

Field	Access	Description	Reset Value
0 HRS0	RO	Read only bit. 0b: No DMA hardware request is asserted in channel 0. 1b: A DMA hardware request is asserted in channel 0.	0x0
1 HRS1	RO	Read only bit. 0b: No DMA hardware request is asserted in channel 1. 1b: A DMA hardware request is asserted in channel 1.	0x0
2 HRS2	RO	Read only bit. 0b: No DMA hardware request is asserted in channel 2. 1b: A DMA hardware request is asserted in channel 2.	0x0
3 HRS3	RO	Read only bit. 0b: No DMA hardware request is asserted in channel 3. 1b: A DMA hardware request is asserted in channel 3.	0x0
4 HRS4	RO	Read only bit. 0b: No DMA hardware request is asserted in channel 4. 1b: A DMA hardware request is asserted in channel 4.	0x0
5 HRS5	RO	Read only bit. 0b: No DMA hardware request is asserted in channel 5. 1b: A DMA hardware request is asserted in channel 5.	0x0
6 HRS6	RO	Read only bit. 0b: No DMA hardware request is asserted in channel 6.	0x0

Field	Access	Description	Reset Value
		1b: A DMA hardware request is asserted in channel 6.	
7 HRS7	RO	Read only bit. 0b: No DMA hardware request is asserted in channel 7. 1b: A DMA hardware request is asserted in channel 7.	0x0
8 HRS8	RO	Read only bit. 0b: No DMA hardware request is asserted in channel 8. 1b: A DMA hardware request is asserted in channel 8.	0x0
9 HRS9	RO	Read only bit. 0b: No DMA hardware request is asserted in channel 9. 1b: A DMA hardware request is asserted in channel 9.	0x0
10 HRS10	RO	Read only bit. 0b: No DMA hardware request is asserted in channel 10. 1b: A DMA hardware request is asserted in channel 10.	0x0
11 HRS11	RO	Read only bit. 0b: No DMA hardware request is asserted in channel 11. 1b: A DMA hardware request is asserted in channel 11.	0x0
12 HRS12	RO	Read only bit. 0b: No DMA hardware request is asserted in channel 12. 1b: A DMA hardware request is asserted in channel 12.	0x0
13 HRS13	RO	Read only bit. 0b: No DMA hardware request is asserted in channel 13. 1b: A DMA hardware request is asserted in channel 13.	0x0
14 HRS14	RO	Read only bit. 0b: No DMA hardware request is asserted in channel 14. 1b: A DMA hardware request is asserted in channel 14.	0x0
15 HRS15	RO	Read only bit. 0b: No DMA hardware request is asserted in channel 15. 1b: A DMA hardware request is asserted in channel 15.	0x0
31:16 RSVD_31_16	RO	Reserved	0x0

9.2.10 DMA Channel Priority Register 0 (DMA_CPRI0)

9.2.10.1 Offset

Register	Offset
DMA_CPRI0	0x28

9.2.10.2 Function

This register provides the channel 0/1/2/3 priority configuration, suspension, and preemption properties. If a fixed channel priority arbitration is selected, DMA will execute channel service requests based on the priority configuration of the channel. It will execute the service request in the sequence of priority, starting from the service requests with high priority followed by the service requests with low priority.

If a round-robin channel arbitration is selected, DMA will execute channel service requests from channel 15 to channel 0. Every channel has its unique priority. Otherwise, a priority error will be generated. The suspension and preemption properties define the concurrent behavior of the channels.

9.2.10.3 Fields

Field	Access	Description	Reset Value

Field	Access	Description	Reset Value
3:0 CPRI3	RW	Channel 3 can configure the priority when fixed arbitration is enabled. CPRI3 reset value equals the corresponding channel number: CPRI3 = 4'b0011.	0x3
5:4 RSVD_5_4	RO	Reserved	0x0
6 CPDIS3	RW	Channel 3 preemption is disabled. This bit defines whether channel 3 can temporarily preempt a lower priority channel when channel 3 DMA request is asserted while a lower priority channel's DMA is servicing. 0b: Channel 3 DMA request can preempt a lower priority channel's service. 1b: Channel 3 DMA request can NOT preempt a lower priority channel's service.	0x0
7 CSE3	RW	Channel 3 suspension is enabled. This bit defines whether channel 3 DMA service can be temporarily suspended by higher priority channel when channel 3 DMA request is servicing and a higher priority channel's DMA request is asserted. 0b: Channel 3 DMA service can NOT be suspended by other channels. 1b: Channel 3 DMA service can be suspended by other channels.	0x0
11:8 CPRI2	RW	Channel 2 can configure the priority when fixed arbitration is enabled. CPRI2 reset value equals the corresponding channel number: CPRI2 = 4'b0010.	0x2
13:12 RSVD_13_12	RO	Reserved	0x0
14 CPDIS2	RW	Channel 2 preemption is disabled. This bit defines whether channel 2 can temporarily preempt a lower priority channel when channel 2 DMA request is asserted but a lower priority channel's DMA is servicing. 0b: Channel 2 DMA request can preempt a lower priority channel's service. 1b: Channel 2 DMA request can NOT preempt a lower priority channel's service.	0x0
15 CSE2	RW	Channel 2 suspension is enabled. This bit defines whether channel 2 DMA service can be temporarily suspended by higher priority channel when channel 2 DMA request is servicing and a higher priority channel's DMA request is asserted. 0b: Channel 2 DMA service can NOT be suspended by other channel. 1b: Channel 2 DMA service can be suspended by other channel.	0x0
19:16 CPRI1	RW	Channel 1 can configure the priority when fixed arbitration is enabled. CPRI1 reset value equals the corresponding channel number: CPRI1 = 4'b0001.	0x1
21:20 RSVD_21_20	RO	Reserved	0x0
22 CPDIS1	RW	Channel 1 preemption is disabled. This bit defines whether channel 1 can temporarily preempt a lower priority channel when channel 1 DMA request is asserted but a lower priority channel's DMA is servicing.	0x0

Field	Access	Description	Reset Value
		0b: Channel 1 DMA request can preempt a lower priority channel's service. 1b: Channel 1 DMA request can NOT preempt a lower priority channel's service.	
23 CSE1	RW	Channel 1 suspension is enabled. This bit defines whether channel 1 DMA service can be temporarily suspended by higher priority channel when channel 1 DMA request is servicing and a higher priority channel's DMA request is asserted. 0b: Channel 1 DMA service can NOT be suspended by other channel. 1b: Channel 1 DMA service can be suspended by other channel.	0x0
27:24 CPRI0	RW	Channel 0 can configure the priority when fixed arbitration is enabled. CPRI0 reset value equals the corresponding channel number: CPRI0 = 4'b0000.	0x0
29:28 RSVD_29_28	RO	Reserved	0x0
30 CPDIS0	RW	Channel 0 preemption is disabled. This bit defines whether channel 0 can temporarily preempt a lower priority channel when channel 0 DMA request is asserted but a lower priority channel's DMA is servicing. 0b: Channel 0 DMA request can preempt a lower priority channel's service. 1b: Channel 0 DMA request can NOT preempt a lower priority channel's service.	0x0
31 CSE0	RW	Channel 0 suspension is enabled. This bit defines whether channel 0 DMA service can be temporarily suspended by higher priority channel when channel 0 DMA request is servicing and a higher priority channel's DMA request is asserted. 0b: Channel 0 DMA service can NOT be suspended by other channel. 1b: Channel 0 DMA service can be suspended by other channel.	0x0

9.2.11 DMA Channel Priority Register 1 (DMA_CPRI1)

9.2.11.1 Offset

Register	Offset
DMA_CPRI1	0x2C

9.2.11.2 Function

This register provides the channel 4/5/6/7 priority configuration, suspension, and preemption properties.

9.2.11.3 Fields

Field	Access	Description	Reset Value
3:0 CPRI7	RW	Channel 7 can configure the priority when fixed arbitration is enabled.	0x7

Field	Access	Description	Reset Value
		CPRI7 reset value equals the corresponding channel number: CPRI7 = 4'b0111.	
5:4 RSVD_5_4	RO	Reserved	0x0
6 CPDIS7	RW	Channel 7 preemption is disabled. This bit defines when channel 7 DMA request is asserted but a lower priority channel's DMA is servicing, whether channel 7 can temporarily preempt a lower priority channel. 0b: channel 7 DMA request can preempt a lower priority channel's service. 1b: channel 7 DMA request can NOT preempt a lower priority channel's service.	0x0
7 CSE7	RW	Channel 7 suspension is enabled. This bit defines when channel 7 DMA request is servicing and a higher priority channel's DMA request is asserted, whether channel 7 DMA service can be temporarily suspended by higher priority channel. 0b: channel 7 DMA service can NOT be suspended by other channel. 1b: channel 7 DMA service can be suspended by other channel.	0x0
11:8 CPRI6	RW	Channel 6 can configure the priority when fixed arbitration is enabled. CPRI6 reset value is equal to the corresponding channel number, CPRI6 = 4'b0110.	0x6
13:12 RSVD_13_12	RO	Reserved	0x0
14 CPDIS6	RW	Channel 6 preemption is disabled. This bit defines when channel 6 DMA request is asserted but a lower priority channel's DMA is servicing, whether channel 6 can temporarily preempt a lower priority channel. 0b: channel 6 DMA request can preempt a lower priority channel's service. 1b: channel 6 DMA request can NOT preempt a lower priority channel's service.	0x0
15 CSE6	RW	Channel 6 suspension is enabled. This bit defines when channel 6 DMA request is servicing and a higher priority channel's DMA request is asserted, whether channel 6 DMA service can be temporarily suspended by higher priority channel. 0b: channel 6 DMA service can NOT be suspended by other channel. 1b: channel 6 DMA service can be suspended by other channel.	0x0
19:16 CPRI5	RW	Channel 5 can configure the priority when fixed arbitration is enabled. CPRI5 reset value is equal to the corresponding channel number, CPRI5 = 4'b0101.	0x5
21:20 RSVD_21_20	RO	Reserved	0x0
22 CPDIS5	RW	Channel 5 preemption is disabled. This bit defines when channel 5 DMA request is asserted but a lower priority channel's DMA is servicing, whether channel 5 can temporarily preempt a lower priority channel. 0b: channel 5 DMA request can preempt a lower priority channel's service.	0x0

Field	Access	Description	Reset Value
		1b: channel 5 DMA request can NOT preempt a lower priority channel's service.	
23 CSE5	RW	Channel 5 suspension is enabled. This bit defines when channel 5 DMA request is servicing and a higher priority channel's DMA request is asserted, whether channel 5 DMA service can be temporarily suspended by higher priority channel. 0b: channel 5 DMA service can NOT be suspended by other channel. 1b: channel 5 DMA service can be suspended by other channel.	0x0
27:24 CPRI4	RW	Channel 4 can configure the priority when fixed arbitration is enabled. CPRI4 reset value is equal to the corresponding channel number, CPRI4 = 4'b0100.	0x4
29:28 RSVD_29_28	RO	Reserved	0x0
30 CPDIS4	RW	Channel 4 preemption is disabled. This bit defines when channel 4 DMA request is asserted but a lower priority channel's DMA is servicing, whether channel 4 can temporarily preempt a lower priority channel. 0b: channel 4 DMA request can preempt a lower priority channel's service. 1b: channel 4 DMA request can NOT preempt a lower priority channel's service.	0x0
31 CSE4	RW	Channel 4 suspension is enabled. This bit defines when channel 4 DMA request is servicing and a higher priority channel's DMA request is asserted, whether channel 4 DMA service can be temporarily suspended by higher priority channel. 0b: channel 4 DMA service can NOT be suspended by other channel. 1b: channel 4 DMA service can be suspended by other channel.	0x0

9.2.12 DMA Channel Priority Register 2 (DMA_CPRI2)

9.2.12.1 Offset

Register	Offset
DMA_CPRI2	0x30

9.2.12.2 Function

This register provides the channel 8/9/10/11 priority configuration, suspension and preemption properties.

9.2.12.3 Fields

Field	Access	Description	Reset Value
3:0 CPRI11	RW	Channel 11 can configure the priority when fixed arbitration is enabled. CPRI11 reset value equals the corresponding channel number: CPRI11 = 4'b1011.	0xB

Field	Access	Description	Reset Value
5:4 RSVD_5_4	RO	Reserved	0x0
6 CPDIS11	RW	Channel 11 preemption is disabled. This bit defines whether channel 11 can temporarily preempt a lower priority channel when channel 11 DMA request is asserted while a lower priority channel's DMA is servicing. 0b: Channel 11 DMA request can preempt a lower priority channel's service. 1b: Channel 11 DMA request can NOT preempt a lower priority channel's service.	0x0
7 CSE11	RW	Channel 11 suspension is enabled. This bit defines whether channel 11 DMA service can be temporarily suspended by higher priority channel when channel 11 DMA request is servicing and a higher priority channel's DMA request is asserted. 0b: Channel 11 DMA service can NOT be suspended by other channel. 1b: Channel 11 DMA service can be suspended by other channel.	0x0
11:8 CPRI10	RW	Channel 10 can configure the priority when fixed arbitration is enabled. CPRI10 reset value equals the corresponding channel number: CPRI10 = 4'b1010.	0xA
13:12 RSVD_13_12	RO	Reserved	0x0
14 CPDIS10	RW	Channel 10 preemption is disabled. This bit defines whether channel 10 can temporarily preempt a lower priority channel when channel 10 DMA request is asserted while a lower priority channel's DMA is servicing. 0b: Channel 10 DMA request can preempt a lower priority channel's service. 1b: Channel 10 DMA request can NOT preempt a lower priority channel's service.	0x0
15 CSE10	RW	Channel 10 suspension is enabled. This bit defines whether channel 10 DMA service can be temporarily suspended by higher priority channel when channel 10 DMA request is servicing and a higher priority channel's DMA request is asserted. 0b: Channel 10 DMA service can NOT be suspended by other channel. 1b: Channel 10 DMA service can be suspended by other channel.	0x0
19:16 CPRI9	RW	Channel 9 can configure the priority when fixed arbitration is enabled. CPRI9 reset value equals the corresponding channel number: CPRI9 = 4'b1001.	0x9
21:20 RSVD_21_20	RO	Reserved	0x0
22 CPDIS9	RW	Channel 9 preemption is disabled. This bit defines whether channel 9 can temporarily preempt a lower priority channel when channel 9 DMA request is asserted but a lower priority channel's DMA is servicing. 0b: Channel 9 DMA request can preempt a lower priority channel's service.	0x0

Field	Access	Description	Reset Value
		1b: Channel 9 DMA request can NOT preempt a lower priority channel's service.	
23 CSE9	RW	Channel 9 suspension is enabled. This bit defines whether channel 9 DMA service can be temporarily suspended by higher priority channel when channel 9 DMA request is servicing and a higher priority channel's DMA request is asserted. 0b: Channel 9 DMA service can NOT be suspended by other channel. 1b: Channel 9 DMA service can be suspended by other channel.	0x0
27:24 CPRI8	RW	Channel 8 can configure the priority when fixed arbitration is enabled. CPRI8 reset value equals the corresponding channel number: CPRI8 = 4'b1000.	0x8
29:28 RSVD_29_28	RO	Reserved	0x0
30 CPDIS8	RW	Channel 8 preemption is disabled. This bit defines whether channel 8 can temporarily preempt a lower priority channel when channel 8 DMA request is asserted while a lower priority channel's DMA is servicing. 0b: Channel 8 DMA request can preempt a lower priority channel's service. 1b: Channel 8 DMA request can NOT preempt a lower priority channel's service.	0x0
31 CSE8	RW	Channel 8 suspension is enabled. This bit defines whether channel 8 DMA service can be temporarily suspended by higher priority channel when channel 8 DMA request is servicing and a higher priority channel's DMA request is asserted. 0b: Channel 8 DMA service can NOT be suspended by other channel. 1b: Channel 8 DMA service can be suspended by other channel.	0x0

9.2.13 DMA Channel Priority Register 3 (DMA_CPRI3)

9.2.13.1 Offset

Register	Offset
DMA_CPRI3	0x34

9.2.13.2 Function

This register provides the channel 12/13/14/15 priority configuration, suspension and preemption properties.

9.2.13.3 Fields

Field	Access	Description	Reset Value
3:0 CPRI15	RW	Channel 15 can configure the priority when fixed arbitration is enabled. CPRI15 reset value equals the corresponding channel number: CPRI15 = 4'b1111.	0xF

Field	Access	Description	Reset Value
5:4 RSVD_5_4	RO	Reserved	0x0
6 CPDIS15	RW	Channel 15 preemption is disabled. This bit defines whether channel 15 can temporarily preempt a lower priority channel when channel 15 DMA request is asserted but a lower priority channel's DMA is servicing. 0b: Channel 15 DMA request can preempt a lower priority channel's service. 1b: Channel 15 DMA request can NOT preempt a lower priority channel's service.	0x0
7 CSE15	RW	Channel 15 suspension is enabled. This bit defines whether channel 15 DMA service can be temporarily suspended by higher priority channel when channel 15 DMA request is servicing and a higher priority channel's DMA request is asserted. 0b: Channel 15 DMA service can NOT be suspended by other channel. 1b: Channel 15 DMA service can be suspended by other channel.	0x0
11:8 CPRI14	RW	Channel 14 can configure the priority when fixed arbitration is enabled. CPRI14 reset value equals the corresponding channel number: CPRI14 = 4'b1110.	0xE
13:12 RSVD_13_12	RO	Reserved	0x0
14 CPDIS14	RW	Channel 14 preemption is disabled. This bit defines whether channel 14 can temporarily preempt a lower priority channel when channel 14 DMA request is asserted but a lower priority channel's DMA is servicing. 0b: Channel 14 DMA request can preempt a lower priority channel's service. 1b: Channel 14 DMA request can NOT preempt a lower priority channel's service.	0x0
15 CSE14	RW	Channel 14 suspension is enabled. This bit defines whether channel 14 DMA service can be temporarily suspended by higher priority channel when channel 14 DMA request is servicing and a higher priority channel's DMA request is asserted. 0b: Channel 14 DMA service can NOT be suspended by other channel. 1b: Channel 14 DMA service can be suspended by other channel.	0x0
19:16 CPRI13	RW	Channel 13 can configure the priority when fixed arbitration is enabled. CPRI13 reset value equals the corresponding channel number: CPRI13 = 4'b1101.	0xD
21:20 RSVD_21_20	RO	Reserved	0x0
22 CPDIS13	RW	Channel 13 preemption is disabled. This bit defines whether channel 13 can temporarily preempt a lower priority channel when channel 13 DMA request is asserted but a lower priority channel's DMA is servicing. 0b: Channel 13 DMA request can preempt a lower priority	0x0

Field	Access	Description	Reset Value
		channel's service. 1b: Channel 13 DMA request can NOT preempt a lower priority channel's service.	
23 CSE13	RW	Channel 13 suspension is enabled. This bit defines whether channel 13 DMA service can be temporarily suspended by higher priority channel when channel 13 DMA request is servicing and a higher priority channel's DMA request is asserted. 0b: Channel 13 DMA service can NOT be suspended by other channel. 1b: Channel 13 DMA service can be suspended by other channel.	0x0
27:24 CPRI12	RW	Channel 12 can configure the priority when fixed arbitration is enabled. CPRI12 reset value equals the corresponding channel number: CPRI12 = 4'b1100.	0xC
29:28 RSVD_29_28	RO	Reserved	0x0
30 CPDIS12	RW	Channel 12 preemption is disabled. This bit defines whether channel 12 can temporarily preempt a lower priority channel when channel 12 DMA request is asserted but a lower priority channel's DMA is servicing. 0b: Channel 12 DMA request can preempt a lower priority channel's service. 1b: Channel 12 DMA request can NOT preempt a lower priority channel's service.	0x0
31 CSE12	RW	Channel 12 suspension is enabled. This bit defines whether channel 12 DMA service can be temporarily suspended by higher priority channel when channel 12 DMA request is servicing and a higher priority channel's DMA request is asserted. 0b: Channel 12 DMA service can NOT be suspended by other channel. 1b: Channel 12 DMA service can be suspended by other channel.	0x0

9.2.14 DMA Channel n Source Address Register (DMA_SADDRn)

9.2.14.1 Offset

Register	Offset
DMA_SADDRn	0x200 + 0x20*n(n=0~15)

9.2.14.2 Function

DMA_SADDRn(n=0~15) register provides channel n source address. For the first DMA transfer in channel n, DMA reads data from this source address.

9.2.14.3 Fields

Field	Access	Description	Reset Value
31:0 SADDR	RW	DMA transfers read data from this source address.	0x0

9.2.15 DMA Channel n Destination Address Register (DMA_DADDRn)

9.2.15.1 Offset

Register	Offset
DMA_DADDRn	$0x204 + 0x20 \times n (n=0 \sim 15)$

9.2.15.2 Function

DMA_DADDRn(n=0~15) register provides channel n destination address. For the first DMA transfer in channel n, DMA writes data to this destination address.

9.2.15.3 Fields

Field	Access	Description	Reset Value
31:0 DADDR	RW	DMA transfers write data to this destination address.	0x0

9.2.16 DMA Channel n Address Offset Register (DMA_AOFFn)

9.2.16.1 Offset

Register	Offset
DMA_AOFFn	$0x208 + 0x20 \times n (n=0 \sim 15)$

9.2.16.2 Function

DMA_AOFFn(n=0~15) register provides the source and destination address offset value in channel n in a minor loop. For channel n, DMA transfers data in a minor loop. The next transfer source/destination address depends on the offset value of this source/destination. DAOFF and SAOFF are both 16-bit signed number with the MSB bit representing the signed bit. When MSB is set to 1, it represents that the offset value is negative.

9.2.16.3 Fields

Field	Access	Description	Reset Value
15:0 SAOFF	RW	This is a 16-bit signed number, the MSB bit represents the signed bit. In a minor loop, DMA next transfers read data from source address plus this source address offset.	0x0
31:16 DAOFF	RW	This is a 16-bit signed number, the MSB bit represents the signed bit. In a minor loop, DMA next transfers write data to destination address plus this destination address offset.	0x0

9.2.17 DMA Channel n Major Loop Address Offset Register (DMA_MLAOFFn)

9.2.17.1 Offset

Register	Offset
DMA_MLAOFFn	0x20C + 0x20*n(n=0~15)

9.2.17.2 Function

DMA_MLAOFFn(n=0~15) register provides the source and destination address offset value in channel n between the two major loops. For channel n, DMA transfers data for the next major loop. The first transfer source/destination address depends on the offset value of this major loop source/destination.

MLDAOFF and MLSAOFF are both 16-bit signed numbers with the MSB bit representing the signed bit. When MSB is set to 1, it represents that the offset value is negative.

9.2.17.3 Fields

Field	Access	Description	Reset Value
15:0 MLSAOFF	RW	This is a 16-bit signed number, the MSB bit represents the signed bit. After one major loop is done, DMA next transfers read data from source address plus this source address offset.	0x0
31:16 MLDAOFF	RW	This is a 16-bit signed number, the MSB bit represents the signed bit. After one major loop is done, DMA next transfers write data to destination address plus this destination address offset.	0x0

9.2.18 DMA Channel n Number of Transfer Data (DMA_NUMn)

9.2.18.1 Offset

Register	Offset
DMA_NUMn	0x210 + 0x20*n(n=0~15)

9.2.18.2 Function

DMA_NUMn(n=0~15) register provides the number of data that should be transferred in a minor loop for channel n.

9.2.18.3 Fields

Field	Access	Description	Reset Value
31:0 NUM	RW	This field defines how many bytes of data are transferred for one DMA request service in a minor loop. If this field = 0 and DMA transfer starts, error occurs.	0x0

9.2.19 DMA Channel n Iteration Number Register (DMA_ITERn)

9.2.19.1 Offset

Register	Offset
DMA_ITERn	0x214 + 0x20*n(n=0~15)

9.2.19.2 Function

DMA_ITERn(n=0~15) register provides the minor loop iteration number in one major loop for channel n.

9.2.19.3 Fields

Field	Access	Description	Reset Value
15:0 MLITER	RW	This field defines the iteration number of minor loop that was included in one major loop. Once one minor loop was done for one DMA request service, DMA will assert DMA done to peripheral to deassert DMA request. When the next DMA request is asserted, the second minor loop will start. After the iteration of minor loop was done, the major loop was completed. The DMA_CS[DONE] flag will be set and an interrupt can be generated. NOTE If this field is configured as 0, DMA will not service any DMA request, and DMA is always IDLE.	0x1
31:16 MLSTA	RO	This is the read only field. Reading this field will return the number of iterations of minor loop that were still not finished in one major loop. For example, if MLITER is configured as 3, MLSTA will also be 3 before the first minor loop is done. After the first minor loop is done, MLSTA is set to 2. After the second minor loop is done, MLSTA is 1. After the third minor loop is done, MLSTA will resume to 3 again.	0x1

9.2.20 DMA Channel n Control And Status Register (DMA_CSn)

9.2.20.1 Offset

Register	Offset
DMA_CSn	0x218 + 0x20*n(n=0~15)

9.2.20.2 Function

DMA_CSn(n=0~15) register can configure the source/destination data transfer size of channel n, enable the interrupt after major loop is done, and disable channel n hardware request after DMA transfer is done. It also provides the software DMA request control. It contains the DMA transfer done flag and busy flag.

9.2.20.3 Fields

Field	Access	Description	Reset Value
0 START	RW	When the bit is set, it indicates that this channel is requesting DMA service via software. This bit is asserted by either writing 1 to this field, or writing the corresponding value to SACSTART or SCSTART fields. This bit was cleared automatically by hardware when this channel begins execution. 0b: Channel n is not requesting start service. 1b: Channel n is requesting start service via a software request.	0x0

Field	Access	Description	Reset Value
1 INTE	RW	This bit defines if DMA interrupt is generated when channel n major loop is done. 0b: Disable channel n major loop done interrupt generation. 1b: Enable channel n major loop done interrupt generation.	0x0
2 RSVD_2	RO	Reserved	0x0
3 REQDIS	RW	When this bit is set, after channel n DMA transfer is done, channel n DMA request enable bit DMA_DMAE[DMAEn] will be automatically cleared by hardware. The next DMA request will not be serviced. 0b: DMA_DMAE[DMAEn] is not cleared after DMA transfer is done. 1b: DMA_DMAE[DMAEn] is cleared after DMA transfer is done.	0x0
5:4 RSVD_5_4	RO	Reserved	0x0
6 BUSY	RO	Read only bit. 0b: Channel n is idle 1b: Channel n is busy and the data transfer is executed.	0x0
7 DONE	RW	When this channel major loop was completed, this flag was set. It is cleared by software through three methods: (1) writing 0 to this bit. (2) writing 1 to DMA_GCC[CACDONE] bit. (3) writing corresponding channel value to DMA_GCC[CCDONE] bit.	0x0
15:8 RSVD_15_8	RO	Reserved	0x0
17:16 DSIZE	RW	This field defines the destination data size for one DMA transfer. 00b: 1-byte destination data write transfer 01b: 2-byte destination data write transfer 10b: 4-byte destination data write transfer 11b: Reserved	0x0
23:18 RSVD_23_18	RO	Reserved	0x0
25:24 SSIZE	RW	This field defines source data size for one DMA transfer. 00b: 1-byte source data read transfer 01b: 2-byte source data read transfer 10b: 4-byte source data read transfer 11b: Reserved	0x0
31:26 RSVD_31_26	RO	Reserved	0x0

9.3 Function description

This section provides the functional information of the DMA module.

9.3.1 DMA transactions

The DMA controller contains two bus interfaces: AHB master bus and APB slave bus. AHB master bus is shared with CPU system bus, and it is used to transfer data between peripherals

and memories. APB slave bus is used for DMA register read and write access. The DMA transaction is completed by the following four steps, DMA request->DMA response->DMA transfer->DMA done.

- **DMA request:** The DMA request can be software DMA request or hardware DMA request. DMA software request is asserted by software setting DMA_CS_n[START] bit to start a software request. DMA hardware request is asserted by peripherals and memories, DMA_DMAE[DMAEn] bit should be set to enable channel n DMA hardware request. Prior to DMA request assertion, CPU should initialize the DMA controller first.
- **DMA response:** When the channel n DMA request is asserted, the DMA should arbitrate channel priority and then request CPU to release the system bus for DMA data transfer.
- **DMA transfer:** After DMA controller gets the control of AHB master bus, DMA starts to transfer data. DMA transfers read data from the source address and write data to the destination address. For every DMA read/write data transfer, the transfer data size is programmable. The next transfer address will increment based on source address and destination address offset.
- **DMA done:** After the expected number of data is transferred, which means the DMA transfer is completed on the channel, the DMA controller sends the channel DMA done signal to peripheral to deassert this channel DMA request, and then releases the AHB master bus for CPU. If there are more requests on this channel, the peripheral can initialize the next transaction.

9.3.2 Major loop and minor loop

The DMA only needs to be configured once to implement the major loop and minor loop functions that support continuous DMA request services. DMA services one DMA request, which is called one minor loop. A major loop can contain several minor loops.

For one minor loop, the number of transfer data is determined by DMA_NUM_n[NUM], the data transfer size for one DMA transfer is determined by DMA_CS_n[SSIZE] and DMA_CS_n[DSIZE], the address offset during one minor loop is determined by DMA_AOFF_n[SAOFF] and DMA_AOFF_n[DAOFF]. The address offset between two minor loops is also determined by DMA_AOFF_n[SAOFF] and DMA_AOFF_n[DAOFF].

For one major loop, the number of minor loops is determined by DMA_ITER_n[MLITER]. The address offset between two major loops is determined by DMA_MLAOFF_n[MLSAOFF] and MA_MLAOFF_n[MLDAOFF].

For example, when

- DMA_CS_n[SSIZE]=2'b10
- DMA_CS_n[DSIZE]=2'b10
- DMA_NUM_n[NUM]=12
- DMA_ITER_n[MLITER]=2

The major loop and minor loop structures are shown as following figure.

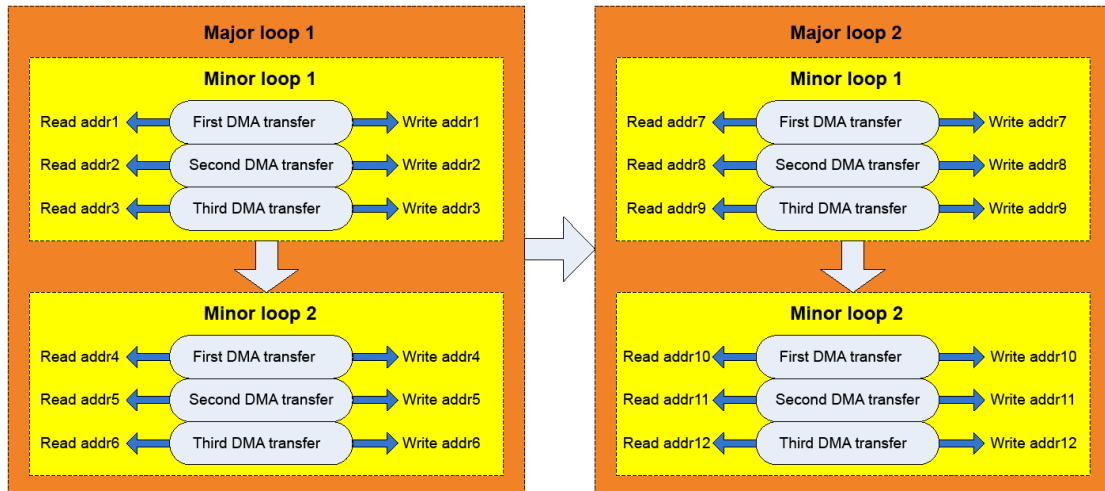


Figure 9-2 major loop and minor loop structures

Read address of every data transfer is shown as following, write address is similar with read address.

- read addr1 = DMA_SADDRn[SADDR]
- read addr2 = read addr1 + DMA_AOFF[SAOFF]
- read addr3 = read addr2 + DMA_AOFF[SAOFF]
- read addr4 = read addr3 + DMA_AOFF[SAOFF]
- read addr5 = read addr4 + DMA_AOFF[SAOFF]
- read addr6 = read addr5 + DMA_AOFF[SAOFF]
- read addr7 = read addr6 + DMA_AOFF[SAOFF] + DMA_MLAOFF[MLSAOFF]
- read addr8 = read addr7 + DMA_AOFF[SAOFF]
- read addr9 = read addr8 + DMA_AOFF[SAOFF]
- read addr10 = read addr9 + DMA_AOFF[SAOFF]
- read addr11 = read addr10 + DMA_AOFF[SAOFF]
- read addr12 = read addr11 + DMA_AOFF[SAOFF]

The major loop transaction sequence is shown as following:

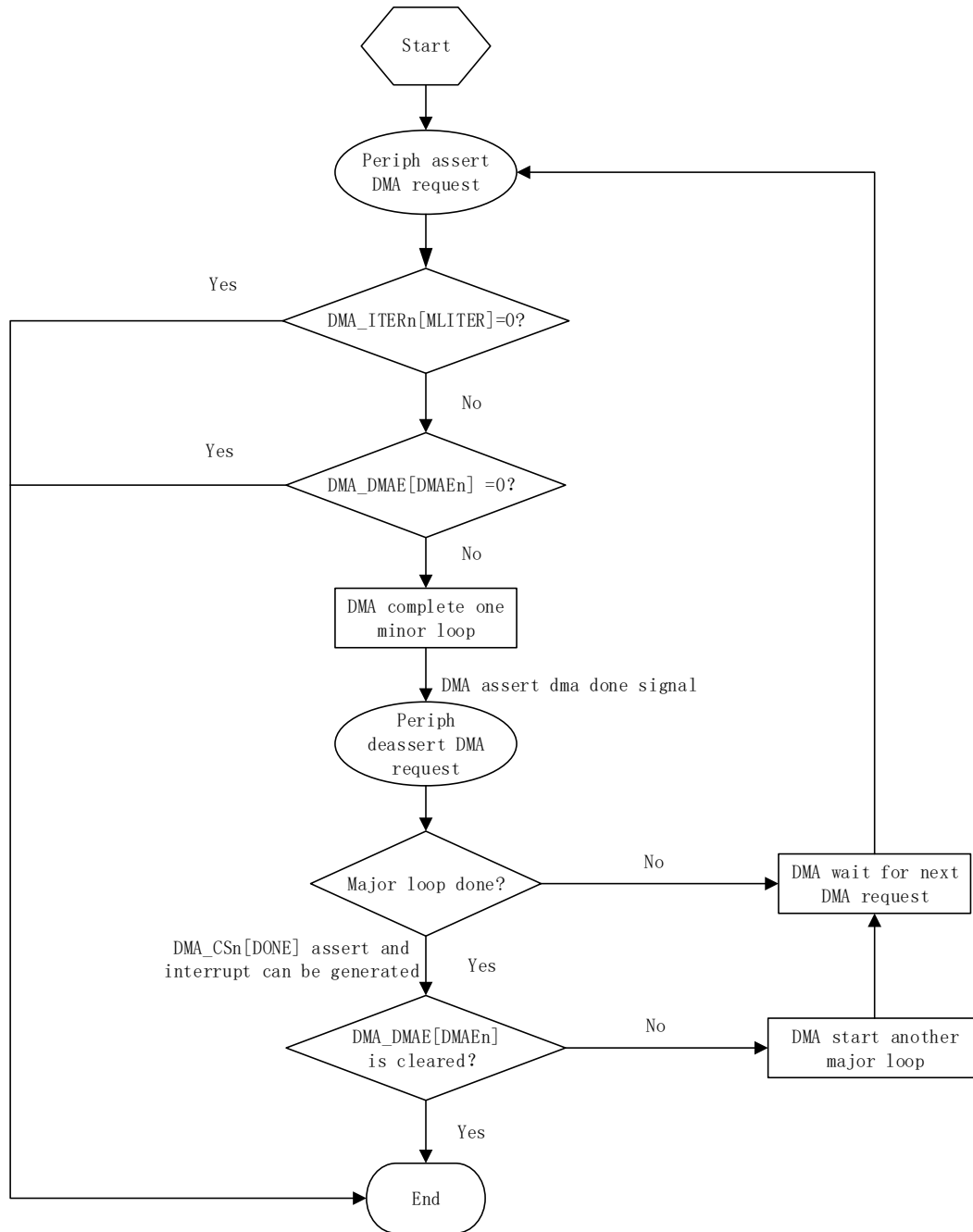


Figure 9-3 DMA major loop transaction sequence

For DMA hardware request service, when channel n DMA hardware request is asserted by peripheral, the DMA starts to service first minor loop. After first minor loop is completed, DMA will feed back channel n DMA done signal to peripheral to deassert the DMA hardware request. But the `DMA_CSn[DONE]` flag will not assert at this time if the major loop contains more than one minor loop. When channel n DMA hardware request is asserted again, the DMA continues to service the second minor loop, and so on.

After DMA transfer for one major loop is completed, the `DMA_CSn[DONE]` flag is set and a major loop done interrupt can be generated. And the `DMA_DMAE[DMAEn]` bit can be automatically cleared by hardware if `DMA_CSn[REQDIS]` is set to 1.

9.3.3 DMA channel priority arbitration

The DMA controller includes 16 independent channels to service 16 DMA requests. But only

one channel's DMA request can be serviced at any time. During this channel service time, other channels' request is holding. So if more than one channel's DMA request is asserted, the DMA channel priority arbiter must arbitrate first and then launch the DMA service sequences. The DMA arbiter supports two types of channel priority arbitration: round-robin priority arbitration and fixed-priority arbitration.

Round-robin priority arbitration is selected when DMA_CONF[RRCAE] is set to 1. In this mode, channels are serviced from the highest channel number to the lowest channel number without considering the channel priority levels, that is, CH15>CH14>.....>CH1>CH0.

Fixed-priority arbitration is selected when DMA_CONF[RRCAE] is set to 0. In this mode, channels are serviced from the highest priority level channel to the lowest priority level channel. Channel priority level is configured by DMA_CPRIx[CPRIn] fields. The CPRIn ranges from 0 to 15. Value 15 indicates highest priority and value 0 indicates lowest priority. Each channel priority level must be configured as unique value, otherwise priority error will occur. After chip reset, the default priority level of each channel is equal to its channel number.

9.3.4 DMA channel program

The 16 DMA channels are independent. Each channel contains 7 registers to program the data transfer read and write control.

- DMA_SADDRn is used to program the channel n source address. DMA transfers the first read data from this source address. The next read address will increment from this source address
- DMA_DADDRn is used to program the channel n destination address. DMA transfers the first write data to this destination address. The next write address will increment from this destination address
- DMA_AOFFn is used to program the source/destination address offset in channel n in a minor loop. The next transfer read/write address during a minor loop will offset based on this SAOFF and DAOFF
- DMA_MLAOFFn is used to program the source/destination address offset in channel n between two major loops. For next major loop transfer, the first read/write address will offset based on MLSAOFF and MLDAOFF
- DMA_NUMn is used to program the number of bytes of channel n transfer data in a minor loop. This register defines how many bytes of data are transferred for one DMA request service in a minor loop
- DMA_ITERn is used to program the minor loop iteration number of channel n in a major loop
- DMA_CSn is used to program the source/destination transfer size of channel n. DMA supports 1-byte/2-byte/4-byte data size for one DMA transfer. DMA may need to complete several DMA transfer for one DMA request service. This register also includes the DMA done and DMA busy status flag. A software DMA request can be started by setting START bit. REQDIS bit defines whether this channel DMA request is disabled after DMA major loop transfer is done. INTE bit enables this channel interrupt after DMA major loop transfer is done

9.3.5 DMA channel error

When channel is programmed with invalid value, channel program errors are reported in the Error status register (DMA_ES). A channel program error is reported with following cases:

- a) Source address error: source address is not aligned with source transfer size SSIZE. For example, SSIZE defines 4-byte data transfer size, then source address must be multiple of 4, otherwise source address error occurs.
- b) Destination address error: destination address is not aligned with destination transfer size DSIZE.

- c) Source address offset error: source address offset or major loop source address offset is not aligned with source transfer size SSIZE.
- d) Destination address offset error: destination address offset or major loop destination address offset is not aligned with destination transfer size DSIZE.
- e) Number of transfer data error: number of transfer data is not multiple of SSIZE or DSIZE.
- f) Channel priority error: channels' priority levels are not unique in fixed-priority arbitration mode.

In addition, an error termination to a bus master read or write cycle also leads to source bus error or destination bus error.

When a channel program error occurs, the appropriate error flag is set and this channel DMA request will not be serviced. If a system read or write bus termination error occurs during the data transfer, the appropriate bus error flag is set and the data transfer will be stopped.

Channel error interrupt is generated when any type error flag is set and the corresponding channel error interrupt enable bit DMA_EIE[EIE_n] is enabled.

Note

When DMA priority error occurs, an error flag will be recorded in DMA_ES[PRIEF]; however, the error channel number reported in DMA_ES[CHEF] and DMA_CES[ERR_n] may not be the right active channel because of priority configuration error. When all of the channel priorities are not unique, the channel number selected by arbitration is undetermined. Software should check all of channel priority configuration whether it is unique when error happens.

9.3.6 Interrupts

The DMA module can generate two interrupts:

- channel DMA done interrupt. This interrupt is generated when DMA_CS_n[INTE] is enabled and DMA_CS_n[DONE] flag is asserted
- channel error interrupt. This interrupt is generated when DMA_EIE[EIE_n] is enabled and any error flag in DMA_ES register is asserted

When DMA done interrupt occurs, software can clear DMA_CS_n[DONE] and DMA_CIS[INT_n] flag by DMA_GCC register. When channel error interrupt occurs, software can clear DMA_CES[ERR_n] and DMA_ES[ERRF] flag by DMA_GCC register. See DMA_GCC register for more details.

9.4 Initialization information

9.4.1 DMA initialization

Before a DMA request starts, the DMA should be initialized with following steps:

1. write to DMA_CONF register to define basic operating configuration.
2. write to DMA_DMAE register to enable channel DMA request if hardware DMA request is desired.
3. write to DMA_EIE register to enable channel error interrupt.
4. write to DMA_CPRIn register to configure channel priority level if fixed-priority arbitration is selected.
5. write to channel n SADDR/DADDR/AOFF/MLAOFF/ITER/NUM registers to configure source address, destination address, source address offset, destination address offset, iteration number and number of transfer data bytes.

6. write to channel n CS register to configure SSIZE and DSIZE, enable channel interrupt, and set DMA_CS_n[REQDIS] to 1 if the user wants to disable DMA request enable bit automatically after DMA transfer is done.
7. wait DMA request start via two ways:
 - a) software DMA request: setting the DMA_CS_n[START] bit.
 - b) hardware DMA request: slave device asserting the DMA request.

Chapter 10

Direct Memory Access Multiplexer (DMAMUX)

10.1 Introduction

The purpose of this chapter is to define Direct Memory Access Multiplexer (DMAMUX). The Direct Memory Access Multiplexer (DMAMUX) module routes DMA input sources to any of the 16 DMA channels.

10.1.1 Features

The DMAMUX supports the following key features:

- It supports up to 94 peripheral input sources and up to 2 always-on input sources.
- 16 independently selectable DMA_MUX_OUT channels to actual DMA controller.
- Each channel of DMA_MUX_OUT can be assigned to one of the possible peripheral input sources or to one of the always-on input sources.

10.1.2 Block diagram

The following figure is a block diagram of the Direct Memory Access Multiplexer (DMAMUX) module.

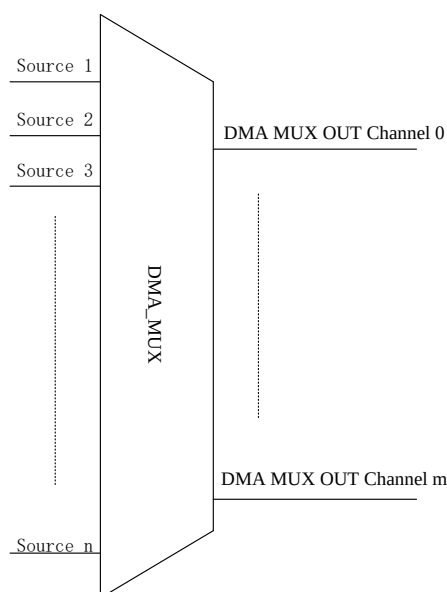


Figure 10-1 DMAMUX block diagram

10.2 Memory map and register definition

This section provides the memory map and register definitions of DMAMUX module. Each DMA_MUX_OUT channel has its own DMA CHANNEL MUX CONFIGURE register.

10.2.1 Memory map

DMAMUX base address: 0x40009000h

Table 10-1 DMAMUX memory map

Offset	Register	Width (In bits)	Reset Value

Offset	Register	Width (In bits)	Reset Value
0x00 + (n * 0x4), (n = 0~15)	DMA Channel n Mux Configuration Register (DMA_MUX_CHn_CFG)	32	0x00000000

10.2.2 DMA Channel n Mux Configure Register (DMA_MUX_CHn_CFG)

10.2.2.1 Offset

Register	Offset
DMA_MUX_CHn_CFG	For n = 0 to 15: 0x00 + (n * 0x4)

10.2.2.2 Function

Each DMA_MUX_OUT channel can be independently enabled/disabled and assigned to one of the peripheral input sources or the always-on input sources.

10.2.2.3 Fields

Field	Access	Description	Reset Value
0 ENABLE	RW	DMAMUX has a total of 16 output channels 0b: means disable DMAMUX output channel 1b: means enable DMAMUX output channel	0x0
7:1 SRC_MUX	RW	DMA CHANNEL MUX has a total of 96 input sources 0: Invalid input source 1: Invalid input source 2: UART0 dma tx request 3: UART0 dma rx request 4: UART1 dma tx request 5: UART1 dma rx request 6: UART2 dma tx request 7: UART2 dma rx request 8: UART3 dma tx request 9: UART3 dma rx request 10: UART4 dma tx request 11: UART4 dma rx request 12: UART5 dma tx request 13: UART5 dma rx request 14: SPI0 dma tx request 15: SPI0 dma rx request 16: SPI1 dma tx request 17: SPI1 dma rx request 18: SPI2 dma tx request 19: SPI2 dma rx request 20: SPI3 dma tx request 21: SPI3 dma rx request 22: STIM dma request 0 23: STIM dma request 1 24: STIM dma request 2 25: STIM dma request 3 26: I2C0 dma tx request 27: I2C0 dma rx request	0x0

Field	Access	Description	Reset Value
		28: I2C1 dma tx request 29: I2C1 dma rx request 30: CAN0 dma request 31: CAN1 dma request 32: CAN2 dma request 33: CAN3 dma request 34: CAN4 dma request 35: CAN5 dma request 36: CAN6 dma request 37: CAN7 dma request 38: TIM0 channel 0 dma request 39: TIM0 channel 1 dma request 40: TIM0 channel 2 dma request 41: TIM0 channel 3 dma request 42: TIM0 channel 4 dma request 43: TIM0 channel 5 dma request 44: TIM0 channel 6 dma request 45: TIM0 channel 7 dma request 46: TIM1 channel 0 dma request 47: TIM1 channel 1 dma request 48: TIM1 channel 2 dma request 49: TIM1 channel 3 dma request 50: TIM1 channel 4 dma request 51: TIM1 channel 5 dma request 52: TIM1 channel 6 dma request 53: TIM1 channel 7 dma request 54: TIM2 channel 0 dma request 55: TIM2 channel 1 dma request 56: TIM2 channel 2 dma request 57: TIM2 channel 3 dma request 58: TIM2 channel 4 dma request 59: TIM2 channel 5 dma request 60: TIM2 channel 6 dma request 61: TIM2 channel 7 dma request 62: TIM3 channel 0 dma request 63: TIM3 channel 1 dma request 64: TIM3 channel 2 dma request 65: TIM3 channel 3 dma request 66: TIM3 channel 4 dma request 67: TIM3 channel 5 dma request 68: TIM3 channel 6 dma request 69: TIM3 channel 7 dma request 70: MPWM0 channel 0 dma request 71: MPWM0 channel 1 dma request 72: MPWM0 channel 2 dma request 73: MPWM0 channel 3 dma request 74: MPWM0 channel 4 dma request 75: MPWM0 channel 5 dma request 76: MPWM0 channel 6 dma request 77: MPWM0 channel 7 dma request 78: MPWM1 channel 0 dma request 79: MPWM1 channel 1 dma request 80: MPWM1 channel 2 dma request 81: MPWM1 channel 3 dma request 82: MPWM1 channel 4 dma request 83: MPWM1 channel 5 dma request	

Field	Access	Description	Reset Value
		84: MPWM1 channel 6 dma request 91: PORTA dma request 92: PORTB dma request 93: PORTC dma request 94: PORTD dma request 95: PORTE dma request 96: DMAMUX always-on channel 0 97: DMAMUX always-on channel 1 98: Invalid input source ... 127: Invalid input source	
31:8 RSVD_31_8	RO	Reserved	0x0

10.3 Function description

This section provides the functional information of the DMAMUX module.

10.3.1 Normal routing from peripheral

The main purpose of the DMAMUX is to support flexible system use of DMA channels. For normal routing case, software needs to configure certain peripheral to generate DMA request, then route this request to DMA_MUX_OUT channel via DMAMUX to trigger the DMA transfer.

10.3.2 Always-on routing channel

There are 2 always-on input sources that are always enabled. For always-on routing case, it will be very useful as following cases without waiting for peripheral generating DMA request:

- Perform DMA transfer from memory to memory — moving data from memory to memory, typically as fast as possible
- Perform DMA transfer from/to GPIO — moving data from/to one or more GPIO pins

10.4 Initialization/Application information

This section provides the application information of the DMAMUX module.

10.4.1 Peripheral DMA routing

1. Clear the DMA_MUX_CHn_CFG[ENABLE] field of the DMA channel.
2. Select the input source to be routed to the DMA_MUX_OUT channel by writing to the DMA_MUX_CHn_CFG[SRC_MUX] field.
3. Set the DMA_MUX_CHn_CFG[ENABLE] field to enable DMA_MUX_OUT channel.
4. Configure source/destination transfer parameters related to DMA controller.
5. Start a DMA transfer.

10.4.2 Always-on DMA routing

1. Clear the DMA_MUX_CHn_CFG[ENABLE] field of the DMA channel.

2. Select one of the always-on input sources to be routed to the DMA_MUX_OUT channel by writing to the DMA_MUX_CHn_CFG[SRC_MUX] field.
3. Set the DMA_MUX_CHn_CFG[ENABLE] field to enable DMA_MUX_OUT channel.
4. Configure source/destination transfer parameters related to DMA controller.
5. Start a DMA transfer.

Chapter 11

Trigger Mux Unit (TMU)

11.1 Introduction

The chapter defines Trigger Mux Unit (TMU). The TMU module allows software to select the trigger source for peripherals.

11.1.1 Block diagram

The following figure is a block diagram of the Trigger Mux Unit (TMU) module.

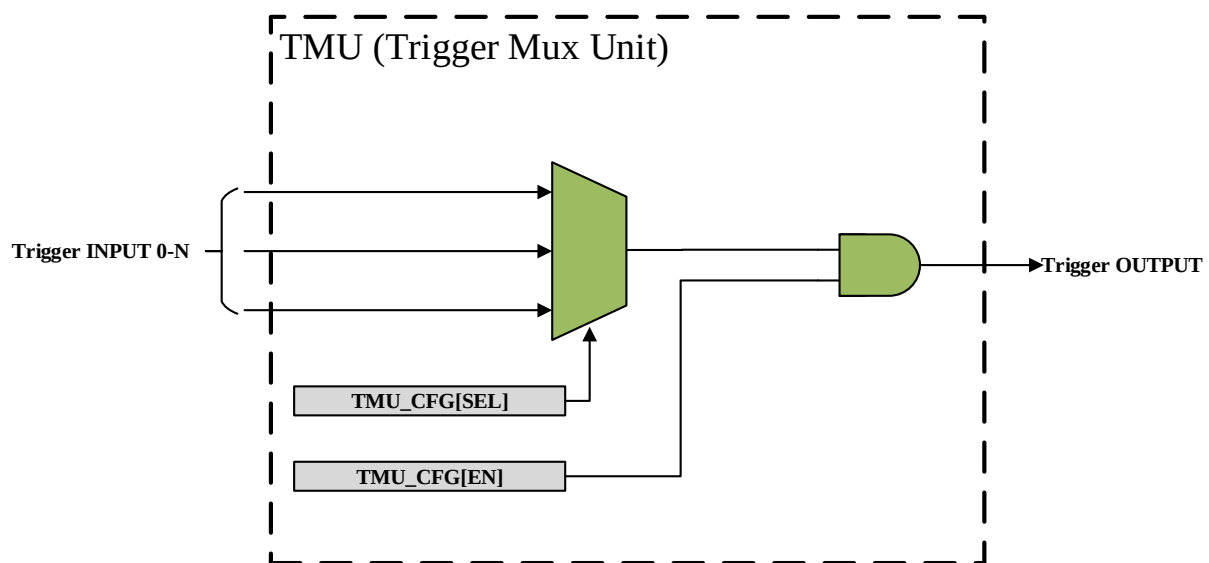


Figure 11-1 TMU block diagram

11.2 Signal description

The section describes the signals of TMU module.

Table 11-1 TMU signal descriptions

Signal	Description	I/O
TMU_OUTPUT	TRIGGERMUX_OUTPUT	Output
TMU_INPUT	TRIGGERMUX_INPUT	Input

11.3 Memory map and register definition

This section provides the memory map and register definitions of the TMU module.

11.3.1 Memory map

TMU base address: 0x40063000

Table 11-2 TMU memory map

Offset	Register	Width (In bits)	Reset Value

Offset	Register	Width (In bits)	Reset Value
0x0 + 0x4*n, (n=0~39)	Trigger Mux Unit Configuration (TMU_CFGn)	32	0x80000000

11.3.2 Trigger Mux Unit Configuration (TMU_CFGn)

11.3.2.1 Offset

Register	Offset
TMU_CFGn	0x0 + 0x4*n, (n=0~39)

11.3.2.2 Function

This register is used for TMU configuration.

11.3.2.3 Fields

Field	Access	Function	Reset Value
15:0 SEL	RW	Select one from 83 input channels to port to mux output.	0x0
16 EN	RW	0b: Mux output disable. 1b: Mux output enable.	0x0
30:17 RSVD_30_17	RO	Reserved	0x0
31 LOCK	RW	Write 0x5B to TMU_CFG[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

11.4 Function description

This section provides the functional information of the TMU module.

11.4.1 TMU input channel description

TMU has 83 input channels. Their detail descriptions are shown as following:

Table 11-3 TMU input channel

Trigger Mux Input Channel	Source Module	Trigger Mux Description
0	VSS	VSS
1	VDD	VDD
2	IO	TMU_INPUT0
3	IO	TMU_INPUT1
4	IO	TMU_INPUT2
5	IO	TMU_INPUT3
6	IO	TMU_INPUT4
7	IO	TMU_INPUT5

Trigger Mux Input Channel	Source Module	Trigger Mux Description
8	IO	TMU_INPUT6
9	IO	TMU_INPUT7
10	IO	TMU_INPUT8
11	IO	TMU_INPUT9
12	IO	TMU_INPUT10
13	IO	TMU_INPUT11
14	RTC	RTC_second
15	RTC	RTC_alarm
16	STIM	Stim_trigger_output[0]
17	STIM	Stim_trigger_output[1]
18	STIM	Stim_trigger_output[2]
19	STIM	Stim_trigger_output[3]
20	TIM0	Match trigger
21	TIM0	Initialization trigger
22	TIM1	Match trigger
23	TIM1	Initialization trigger
24	TIM2	Match trigger
25	TIM2	Initialization trigger
26	TIM3	Match trigger
27	TIM3	Initialization trigger
28	ADC0	Conversion Complete
29	ADC1	Conversion Complete
30	CMP0	CMP out
31	TDG0	TDG0 channel 0 trigger output
32	TDG0	TDG0 channel 1 trigger output
33	TDG0	TDG0 channel 2 trigger output
34	TDG0	TDG0 channel 3 trigger output
35	TDG0	TDG0 channel 4 trigger output
36	TDG0	TDG0 channel 5 trigger output
37	TDG1	TDG1 channel 0 trigger output
38	TDG1	TDG1 channel 1 trigger output
39	TDG1	TDG1 channel 2 trigger output
40	TDG1	TDG1 channel 3 trigger output
41	TDG1	TDG1 channel 4 trigger output
42	TDG1	TDG1 channel 5 trigger output
43	SPI0	SPI_RX_trigger
44	SPI0	SPI_TX_trigger
45	SPI1	SPI_RX_trigger

Trigger Mux Input Channel	Source Module	Trigger Mux Description
46	SPI1	SPI_TX_trigger
47	SPI2	SPI_RX_trigger
48	SPI2	SPI_TX_trigger
49	SPI3	SPI_RX_trigger
50	SPI3	SPI_TX_trigger
51	I2C0	I2C_RX_trigger
52	I2C0	I2C_TX_trigger
53	I2C1	I2C_RX_trigger
54	I2C1	I2C_TX_trigger
55	SCM	SW_TRIGGER: miscctl_reg[0]
56	SCM	SW_TRIGGER: miscctl_reg[1]
57	SCM	SW_TRIGGER: miscctl_reg[2]
58	SCM	SW_TRIGGER: miscctl_reg[3]
59	MCPWM0	MATCH_TRIGGER0
60	MCPWM0	MATCH_TRIGGER1
61	MCPWM0	MATCH_TRIGGER2
62	MCPWM0	MATCH_TRIGGER3
63	MCPWM0	MATCH_TRIGGER4
64	MCPWM0	MATCH_TRIGGER5
65	MCPWM0	MATCH_TRIGGER6
66	MCPWM0	MATCH_TRIGGER7
67	MCPWM0	INIT_TRIGGER0
68	MCPWM0	INIT_TRIGGER1
69	MCPWM0	INIT_TRIGGER2
70	MCPWM0	INIT_TRIGGER3
71	MCPWM1	MATCH_TRIGGER0
72	MCPWM1	MATCH_TRIGGER1
73	MCPWM1	MATCH_TRIGGER2
74	MCPWM1	MATCH_TRIGGER3
75	MCPWM1	MATCH_TRIGGER4
76	MCPWM1	MATCH_TRIGGER5
77	MCPWM1	MATCH_TRIGGER6
78	MCPWM1	MATCH_TRIGGER7
79	MCPWM1	INIT_TRIGGER0
80	MCPWM1	INIT_TRIGGER1
81	MCPWM1	INIT_TRIGGER2
82	MCPWM1	INIT_TRIGGER3

11.4.2 TMU output channel description

TMU has 40 registers as output channel. Detail descriptions are shown as following:

Table 11-4 TMU output channel

Trigger Mux Output Channel	Source Module	Trigger Mux Description
0	IO	TMU_OUTPUT0
1	IO	TMU_OUTPUT1
2	IO	TMU_OUTPUT2
3	IO	TMU_OUTPUT3
4	IO	TMU_OUTPUT4
5	IO	TMU_OUTPUT5
6	IO	TMU_OUTPUT6
7	IO	TMU_OUTPUT7
8	ADC0	ADC HW trigger OPT0
9	ADC0	ADC HW trigger OPT1
10	ADC0	ADC HW trigger OPT2
11	ADC0	ADC HW trigger OPT3
12	ADC1	ADC HW trigger OPT0
13	ADC1	ADC HW trigger OPT1
14	ADC1	ADC HW trigger OPT2
15	ADC1	ADC HW trigger OPT3
16	CMP0	CMP sample
17	CMP0	CMP external enable
18	TDG0	TDG0 trigger input
19	TDG1	TDG1 trigger input
20	TIM0	Fault input channel 0
21	TIM0	Fault input channel 1
22	TIM1	Fault input channel 0
23	TIM1	Fault input channel 1
24	TIM2	Fault input channel 0
25	TIM2	Fault input channel 1
26	TIM3	Fault input channel 0
27	TIM3	Fault input channel 1
28	MCPWM0	Fault input channel 0
29	MCPWM0	Fault input channel 1
30	MCPWM0	Fault input channel 2
31	MCPWM0	Fault input channel 3
32	MCPWM1	Fault input channel 0
33	MCPWM1	Fault input channel 1
34	MCPWM1	Fault input channel 2

Trigger Mux Output Channel	Source Module	Trigger Mux Description
35	MCPWM1	Fault input channel 3
36	STIM	Stim_trigger_input[0]
37	STIM	Stim_trigger_input[1]
38	STIM	Stim_trigger_input[2]
39	STIM	Stim_trigger_input[3]

11.5 Initialization application information

This section provides the application information of the TMU module. The TMU module should be applied with following steps:

1. Set output module unlock
2. Set input source for output module
3. Enable target module
4. Set lock for output module

Chapter 12

Advanced Encryption Standard Accelerator (AES)

12.1 Introduction

The AES hardware accelerator can be used to both encipher and decipher data with 128-bit keys by using AES algorithm as defined by Federal Information Processing Standards Publication.

12.1.1 Features

- The AES supports the following key features:
- Encryption/decryption using AES Rijndael block cipher algorithm
- Electronic codebook (ECB), cipher block chaining (CBC), Output Feedback (OFB), counter mode (CTR), Cipher Feedback (CFB), Galois counter mode (GCM), counter with CBC-MAC (CCM), Tweakable Block Cipher with Ciphertext Stealing (XTS) and cipher message authentication code mode (CMAC) supported
- 128-bit key length
- 128-bit data block processing
- A 128-bit initialization vector when CBC, OFB, CTR or CFB mode is selected
- A 128-bit tweak when XTS mode is selected
- An initialization vector when GCM mode is selected
- 7-13 bytes initialization vector when CCM mode is selected

12.1.2 Modes of operation

The AES operates in the following modes:

- WAIT mode:
AES is enabled in WAIT mode
- STOP mode:
AES is disabled in STOP mode

12.2 Memory map and register definition

This section provides the memory map and register definitions of AES module.

12.2.1 Memory map

AES base address: 0x40033000

Table 12-1 AES memory map

Offset	Register	Width (In bits)	Reset Value
0x0004	Status Register (AES_STATUS)	32	0x00000001
0x0008	Control Register (AES_CONTROL)	32	0x00000000
0x000c	Config Register (AES_CONFIG)	32	0x00000000
0x001c	Data Input 0 Register (AES_DATA_I0)	32	0x00000000
0x0020	Data Input 1 Register	32	0x00000000

Offset	Register	Width (In bits)	Reset Value
	(AES_DATA_I1)		
0x0024	Data Input 2 Register (AES_DATA_I2)	32	0x00000000
0x0028	Data Input 3 Register (AES_DATA_I3)	32	0x00000000
0x002c	Data Output 0 Register (AES_DATA_O0)	32	0x00000000
0x0030	Data Output 1 Register (AES_DATA_O1)	32	0x00000000
0x0034	Data Output 2 Register (AES_DATA_O2)	32	0x00000000
0x0038	Data Output 3 Register (AES_DATA_O3)	32	0x00000000
0x003c	Key 0 Register (AES_KEY0)	32	0x00000000
0x0040	Key 1 Register (AES_KEY1)	32	0x00000000
0x0044	Key 2 Register (AES_KEY2)	32	0x00000000
0x0048	Key 3 Register (AES_KEY3)	32	0x00000000
0x005c	Initialization Vector (AES_IV0)	32	0x00000000
0x0060	Initialization Vector (AES_IV1)	32	0x00000000
0x0064	Initialization Vector (AES_IV2)	32	0x00000000
0x0068	Initialization Vector (AES_IV3)	32	0x00000000

12.2.2 Status Register (AES_STATUS)

12.2.2.1 Offset

Register	Offset
AES_STATUS	0x0004

12.2.2.2 Function

The Status Register (AES_STATUS) provides useful information regarding the IP.

12.2.2.3 Fields

Field	Access	Description	Reset Value
0 R	RO	When this bit is '0b1', then the Block Cipher IP is ready to accept an operation.	0x1
8:1 ERR	RO	Valid when R is '0b1'. When different from 0x0, an error event occurred: "0x1":UNAUTHORIZED_ALGORITHM "0x2":UNAUTHORIZED_KEY_SIZE_FOR_ALGORITHM "0x3":UNAUTHORIZED_CIPHER_DIRECTION "0x4":UNAUTHORIZED_MODE	0x0

Field	Access	Description	Reset Value
		"0x5":UNAUTHORIZED_ALGORITHM_FOR_MODE "0x6":UNAUTHORIZED_DIRECTION_FOR_MODE "0x7":UNAUTHORIZED_KEY_SIZE_FOR_MODE "0xff":UNKNOWN_SETUP_ERROR. These bits are cleared by a start.	
31:9 RSVD_31_9	RO	Reserved	0x0

12.2.3 Control Register (AES_CONTROL)

12.2.3.1 Offset

Register	Offset
AES_CONTROL	0x0008

12.2.3.2 Function

This register is useful to start an AES operation and setup the random value to be used by the countermeasure. It also drives the key selection signal as well as the key register lock signals. Reading this register will cause bus fault.

12.2.3.3 Fields

Field	Access	Description	Reset Value
0 S	WO	Writing '0b1' at this register bit starts an AES operation.	0x0
4:1 RND	WO	4-bit random data to be used by the Block Cipher IP SCA countermeasure.	0x0
31:5 RSVD_31_5	WO	Reserved	0x0

12.2.4 Config Register (AES_CONFIG)

12.2.4.1 Offset

Register	Offset
AES_CONFIG	0x000c

12.2.4.2 Function

This register is useful to setup the AES operation to be performed (key size, encryption/decryption/mode). Reading this register will cause bus fault.

12.2.4.3 Fields

Field	Access	Description	Reset Value
0 CD	WO	0b0: Encryption 0b1: Decryption	0x0
2:1 KSS	WO	"0x0": The Block Cipher IP is used in the AES-128 mode.	0x0
3	WO	Reserved	0x0

Field	Access	Description	Reset Value
RSVD_3			
7:4 RSVD_7_4	WO	Reserved	0x0
15:8 MODE	WO	"0x0": ECB_INIT "0x1": ECB_UPDATE "0x10": CBC_INIT "0x11": CBC_UPDATE "0x20": OFB_INIT "0x21": OFB_UPDATE "0x30": CTR_INIT "0x31": CTR_UPDATE "0x40": CFB_INIT "0x41": CFB_UPDATE "0x50": GCM_INIT "0x51": GCM_GHASH "0x52": GCM_UPDATE "0x53": GCM_FINISH_IV "0x54": GCM_FINISH "0x60": CCM_INIT "0x61": CCM_MAC "0x62": CCM_UPDATE "0x63": CCM_FINISH "0x70": XTS_TWEAK_ENC "0x71": XTS_INIT "0x72": XTS_UPDATE "0x80": CMAC_INIT "0x81": CMAC_UPDATE "0x82": CMAC_FINISH_K1 "0x83": CMAC_FINISH_K2	0x0
23:16 BS	WO	Optional block size in bits (CMAC_FINISH_K1, CMAC_FINISH_K2, GCM_UPDATE, GCM_FINISH, CCM_UPDATE and CCM_FINISH modes of operation only). It is used to perform automatic LSB padding on the output. Its use is not mandatory and it must be set to 0x80 if not used.	0x0
31:24 ALGO	WO	"0x0": AES	0x0

12.2.5 Data Input 0 Register (AES_DATA_I0)

12.2.5.1 Offset

Register	Offset
AES_DATA_I0	0x001c

12.2.5.2 Function

Data input register contains the data to be processed by the Block Cipher IP. This register is wired to data_i [31:0]. Reading this register will cause bus fault.

12.2.5.3 Fields

Field	Access	Description	Reset Value

Field	Access	Description	Reset Value
31:0 DATA_I	WO	data_i [31:0] signal	0x0

12.2.6 Data Input 1 Register (AES_DATA_I1)

12.2.6.1 Offset

Register	Offset
AES_DATA_I1	0x0020

12.2.6.2 Function

Data input register contains the data to be processed by the Block Cipher IP. This register is wired to data_i [63:32]. Reading this register will cause bus fault.

12.2.6.3 Fields

Field	Access	Description	Reset Value
31:0 DATA_I	WO	data_i [63:32] signal	0x0

12.2.7 Data Input 2 Register (AES_DATA_I2)

12.2.7.1 Offset

Register	Offset
AES_DATA_I2	0x0024

12.2.7.2 Function

Data input register contains the data to be processed by the Block Cipher IP. This register is wired to data_i [95:64]. Reading this register will cause bus fault.

12.2.7.3 Fields

Field	Access	Description	Reset Value
31:0 DATA_I	WO	data_i [95:64] signal	0x0

12.2.8 Data Input 3 Register (AES_DATA_I3)

12.2.8.1 Offset

Register	Offset
AES_DATA_I3	0x0028

12.2.8.2 Function

Data input register contains the data to be processed by the Block Cipher IP. This register is wired to data_i [127:96]. Reading this register will cause bus fault.

12.2.8.3 Fields

Field	Access	Description	Reset Value
31:0 DATA_I	WO	data_i [127:96] signal	0x0

12.2.9 Data Output 0 Register (AES_DATA_O0)

12.2.9.1 Offset

Register	Offset
AES_DATA_O0	0x002c

12.2.9.2 Function

Data output register contains the data processed by the Block Cipher IP. This register is wired to data_o [31:0].

12.2.9.3 Fields

Field	Access	Description	Reset Value
31:0 DATA_O	RO	data_o [31:0] signal	0x0

12.2.10 Data Output 1 Register (AES_DATA_O1)

12.2.10.1 Offset

Register	Offset
AES_DATA_O1	0x0030

12.2.10.2 Function

Data output register contains the data processed by the Block Cipher IP. This register is wired to data_o [63:32].

12.2.10.3 Fields

Field	Access	Description	Reset Value
31:0 DATA_O	RO	data_o [63:32] signal	0x0

12.2.11 Data Output 2 Register (AES_DATA_O2)

12.2.11.1 Offset

Register	Offset
AES_DATA_O2	0x0034

12.2.11.2 Function

Data output register contains the data processed by the Block Cipher IP. This register is wired to data_o [95:64].

12.2.11.3 Fields

Field	Access	Description	Reset Value
31:0 DATA_O	RO	data_o [95:64] signal	0x0

12.2.12 Data Output 3 Register (AES_DATA_O3)

12.2.12.1 Offset

Register	Offset
AES_DATA_O3	0x0038

12.2.12.2 Function

Data output register contains the data processed by the Block Cipher IP. This register is wired to data_o [127:96].

12.2.12.3 Fields

Field	Access	Description	Reset Value
31:0 DATA_O	RO	data_o [127:96] signal	0x0

12.2.13 Key 0 Register (AES_KEY0)

12.2.13.1 Offset

Register	Offset
AES_KEY0	0x003c

12.2.13.2 Function

Key input register contains the key. This register is wired to key_i [31:0].

12.2.13.3 Fields

Field	Access	Description	Reset Value
31:0 KEY0	RW	key_i [31:0] signal	0x0

12.2.14 Key 1 Register (AES_KEY1)

12.2.14.1 Offset

Register	Offset

Register	Offset
AES_KEY1	0x0040

12.2.14.2 Function

Key input register contains the key. This register is wired to key_i [63:32].

12.2.14.3 Fields

Field	Access	Description	Reset Value
31:0 KEY1	RW	key_i [63:32] signal	0x0

12.2.15 Key 2 Register (AES_KEY2)

12.2.15.1 Offset

Register	Offset
AES_KEY2	0x0044

12.2.15.2 Function

Key input register contains the key. This register is wired to key_i [95:64].

12.2.15.3 Fields

Field	Access	Description	Reset Value
31:0 KEY2	RW	key_i [95:64] signal	0x0

12.2.16 Key 3 Register (AES_KEY3)

12.2.16.1 Offset

Register	Offset
AES_KEY3	0x0048

12.2.16.2 Function

Key input register contains the key. This register is wired to key_i [127:96].

12.2.16.3 Fields

Field	Access	Description	Reset Value
31:0 KEY3	RW	key_i [127:96] signal	0x0

12.2.17 Initialization Vector (AES_IV0)

12.2.17.1 Offset

Register	Offset
AES_IV0	0x005c

12.2.17.2 Function

Initialization Vector (IV) input register contains the Initialization Vector (IV). This register is wired to iv_i [31:0]. Reading this register will cause bus fault.

12.2.17.3 Fields

Field	Access	Description	Reset Value
31:0 IV0	WO	iv_i [31:0] signal	0x0

12.2.18 Initialization Vector (AES_IV1)

12.2.18.1 Offset

Register	Offset
AES_IV1	0x0060

12.2.18.2 Function

Initialization Vector (IV) input register contains the Initialization Vector (IV). This register is wired to iv_i [63:32]. Reading this register will cause bus fault.

12.2.18.3 Fields

Field	Access	Description	Reset Value
31:0 IV1	WO	iv_i [63:32] signal	0x0

12.2.19 Initialization Vector (AES_IV2)

12.2.19.1 Offset

Register	Offset
AES_IV2	0x0064

12.2.19.2 Function

Initialization Vector (IV) input register contains the Initialization Vector (IV). This register is wired to iv_i [95:64]. Reading this register will cause bus fault.

12.2.19.3 Fields

Field	Access	Description	Reset Value
31:0 IV2	WO	iv_i [95:64] signal	0x0

12.2.20 Initialization Vector (AES_IV3)

12.2.20.1 Offset

Register	Offset
AES_IV3	0x0068

12.2.20.2 Function

Initialization Vector (IV) input register contains the Initialization Vector (IV). This register is wired to iv_i [127:96]. Reading this register will cause bus fault.

12.2.20.3 Fields

Field	Access	Description	Reset Value
31:0 IV3	WO	iv_i [127:96] signal	0x0

12.3 Function Description

This section provides the functional information of the AES module.

12.3.1 ECB

1. Write Key vector to AES_KEY0 – AES_KEY3;
2. Write Data block i to AES_DATA_I0 - AES_DATA_I3;
3. Configure AES_CONFIG:
ALGO: 0x0;
MODE: 0x0(ECB_INIT);
BS: 128;
KSS: 0(AES-128 mode);
CD: 0x0/0x01
4. Reseed and start operation by writing AES_CONTROL;
5. Poll AES_STATUS until the operation is completed;
6. Read Data block i from AES_DATA_O0 - AES_DATA_O3;
7. Repeat step 2 through step 6 (configure AES_CONFIG[MODE] with ECB_UPDATE(0x1)) until the operation of all data blocks is completed.

12.3.2 CBC

1. Write Key vector to AES_KEY0 – AES_KEY3;
2. Write Initialization Vector to AES_IV0 - AES_IV3;
3. Write Data block i to AES_DATA_I0 - AES_DATA_I3;
4. Configure AES_CONFIG:
ALGO: 0x0;
MODE: 0x10(CBC_INIT);
BS: 128;

KSS: 0(AES-128 mode);

CD: 0x0/0x01

5. Reseed and start operation by writing AES_CONTROL;
6. Poll AES_STATUS until the operation is completed;
7. Read Data block i from AES_DATA_O0 - AES_DATA_O3;
8. Repeat step 3 through step 7 (configure AES_CONFIG[MODE] with CBC_UPDATE(0x11)) until the operation of all data blocks is completed.

12.3.3 OFB

1. Write Key vector to AES_KEY0 – AES_KEY3;
2. Write Initialization Vector to AES_IV0 - AES_IV1;
3. Write Data block i to AES_DATA_I0 - AES_DATA_I3;
4. Configure AES_CONFIG:

ALGO: 0x0;

MODE: 0x20(OFB_INIT);

BS: 128;

KSS: 0(AES-128 mode);

CD: 0x0/0x01
5. Reseed and start operation by writing AES_CONTROL;
6. Poll AES_STATUS until the operation is completed;
7. Read Data block i from AES_DATA_O0 - AES_DATA_O3;
8. Repeat step 3 through step 7 (configure AES_CONFIG[MODE] with OFB_UPDATE(0x21)) until the operation of all data blocks is completed.

12.3.4 CTR

1. Write Key vector to AES_KEY0 – AES_KEY3;
2. Write Initialization Vector to AES_IV0 - AES_IV1;
3. Write Data block i to AES_DATA_I0 - AES_DATA_I3;
4. Configure AES_CONFIG:

ALGO: 0x0;

MODE: 0x30(CTR_INIT);

BS: 128;

KSS: 0(AES-128 mode);

CD: 0x0/0x01
5. Reseed and start operation by writing AES_CONTROL;
6. Poll AES_STATUS until the operation is completed;
7. Read Data block i from AES_DATA_O0 - AES_DATA_O3;
8. Repeat step 3 through step 7 (configure AES_CONFIG[MODE] with CTR_UPDATE(0x31)) until the operation of all data blocks is completed.

12.3.5 CFB

1. Write Key vector to AES_KEY0 – AES_KEY3;

2. Write Initialization Vector to AES_IV0 - AES_IV1;
3. Write Data block i to AES_DATA_I0 - AES_DATA_I3;
4. Configure AES_CONFIG:
ALGO: 0x0;
MODE: 0x40(CFB_INIT);
BS: 128;
KSS: 0(AES-128 mode);
CD: 0x0/0x01
5. Reseed and start operation by writing AES_CONTROL;
6. Poll AES_STATUS until the operation is completed;
7. Read Data block i from AES_DATA_O0 - AES_DATA_O3;
8. Repeat step 3 through step 7 (configure AES_CONFIG[MODE] with CFB_UPDATE(0x41)) until the operation of all data blocks is completed.

12.3.6 GCM

When the len(IV)≠96 bits:

1. Write Key vector to AES_KEY0 – AES_KEY3;
2. Configure AES_CONFIG:
ALGO: 0x0;
MODE: 0x50(GCM_INIT);
BS: 128;
KSS: 0(AES-128 mode);
CD: 0x0/0x01
3. Reseed and start operation by writing AES_CONTROL;
4. Poll AES_STATUS until the operation is completed;
5. Hash the Initialization Vector down;
6. Write Initialization Vector to AES_DATA_I0 - AES_DATA_I3;
7. Configure AES_CONFIG:
ALGO: 0x0;
MODE: 0x51(GCM_GHASH);
BS: 128;
KSS: 0(AES-128 mode);
CD: 0x0/0x01
8. Reseed and start operation by writing AES_CONTROL;
9. Poll AES_STATUS until the operation is completed (When len(IV) > 128 bits, repeat steps 6-9 until all the left IV data is handled.);
10. Write len(IV) to AES_DATA_I0 - AES_DATA_I3;
11. Configure AES_CONFIG:
ALGO: 0x0;
MODE: 0x51(GCM_GHASH);
BS: 128;
KSS: 0(AES-128 mode);
CD: 0x0/0x01

12. Reseed and start operation by writing AES_CONTROL;
13. Poll AES_STATUS until the operation is completed;
14. Configure AES_CONFIG:
ALGO: 0x0;
MODE: 0x53(GCM_FINISH_IV);
BS: 128;
KSS: 0(AES-128 mode);
CD: 0x0/0x01
15. Reseed and start operation by writing AES_CONTROL;
16. Poll AES_STATUS until the operation is completed;
17. Write additional authenticated data(AAD) block i to AES_DATA_I0 - AES_DATA_I3;
18. Configure AES_CONFIG:
ALGO: 0x0;
MODE: 0x51(GCM_GHASH);
BS: 128;
KSS: 0(AES-128 mode);
CD: 0x0/0x01
19. Reseed and start operation by writing AES_CONTROL;
20. Poll AES_STATUS until the operation is completed;
21. Repeat step 17 through step 20 until the operation of all AAD data blocks is completed;
22. Write data block i to AES_DATA_I0 - AES_DATA_I3;
23. Configure AES_CONFIG:
ALGO: 0x0;
MODE: 0x52(GCM_UPDATE);
BS: 128(when the length of data block is 128 bits), actual data block length(when the length of data block < 128 bits);
KSS: 0(AES-128 mode);
CD: 0x0/0x01
24. Reseed and start operation by writing AES_CONTROL;
25. Poll AES_STATUS until the operation is completed;
26. Read data block i from AES_DATA_O0 - AES_DATA_O3;
27. Repeat step 22 through step 26 until the operation of all data blocks is completed;
28. Write len(AAD)_64||len(Cipher)_64 value to AES_DATA_I0 - AES_DATA_I3;
29. Configure AES_CONFIG:
ALGO: 0x0;
MODE: 0x51(GCM_GHASH);
BS: 128;
KSS: 0(AES-128 mode);
CD: 0x0/0x01
30. Reseed and start operation by writing AES_CONTROL;
31. Poll AES_STATUS until the operation is completed;
32. Configure AES_CONFIG:
ALGO: 0x0;

- MODE: 0x54(GCM_FINISH);
 BS: 128 or optional length of tag that user needs;
 KSS: 0(AES-128 mode);
 CD: 0x0/0x01
33. Reseed and start operation by writing AES_CONTROL;
 34. Poll AES_STATUS until the operation is completed;
 35. Read Tag from AES_DATA_O0 - AES_DATA_O3.

When the len(IV)=96 bits:

1. Write Key vector to AES_KEY0 – AES_KEY3;
2. Write initialization vector to AES_IV0 – AES_IV3;
3. Configure AES_CONFIG:
 ALGO: 0x0;
 MODE: 0x50(GCM_INIT);
 BS: 128;
 KSS: 0(AES-128 mode);
 CD: 0x0/0x01
4. Reseed and start operation by writing AES_CONTROL;
5. Poll AES_STATUS until the operation is completed;
6. Write additional authenticated data(AAD) block i to AES_DATA_I0 - AES_DATA_I3;
7. Configure AES_CONFIG:
 ALGO: 0x0;
 MODE: 0x51(GCM_GHASH);
 BS: 128;
 KSS: 0(AES-128 mode);
 CD: 0x0/0x01
8. Reseed and start operation by writing AES_CONTROL;
9. Poll AES_STATUS until the operation is completed;
10. Write data block i to AES_DATA_I0 - AES_DATA_I3;
11. Configure AES_CONFIG:
 ALGO: 0x0;
 MODE: 0x52(GCM_UPDATE);
 BS: 128(when the length of data block is 128 bits), actual data block length(when the length of data block < 128 bits);
 KSS: 0(AES-128 mode);
 CD: 0x0/0x01
12. Reseed and start operation by writing AES_CONTROL;
13. Poll AES_STATUS until the operation is completed;
14. Read data block i from AES_DATA_O0 - AES_DATA_O3;
15. Repeat step 10 through step 14 until the operation of all data blocks is completed;
16. If plaintext/ciphertext is empty, ignore step 10 to step 14;
17. Write len(AAD)_64||len(cipher)_64 to AES_I0-AES_I3;

18. Configure AES_CONFIG:

ALGO: 0x0;
MODE: 0x51(GCM_GHASH);
BS: 128;
KSS: 0(AES-128 mode);
CD: 0x0/0x01

19. Reseed and start operation by writing AES_CONTROL;**20. Poll AES_STATUS until the operation is completed;****21. Configure AES_CONFIG:**

ALGO: 0x0;
MODE: 0x54(GCM_FINISH);
BS: 128 or optional length of tag that user needs;
KSS: 0(AES-128 mode);
CD: 0x0/0x01

22. Reseed and start operation by writing AES_CONTROL;**23. Poll AES_STATUS until the operation is completed;****24. Read tag from AES_DATA_O0 - AES_DATA_O3.**

The user should perform LSB padding on IV, ADD, and data when they are not 128-bit aligned.

12.3.7 CCM

1. Write Key vector to AES_KEY0 – AES_KEY3;**2. Write formatted counter ctr 0 to AES_IV0 – AES_IV3;****3. Write formatted data block 0 to AES_DATA_I0 - AES_DATA_I3;****4. Configure AES_CONFIG:**

ALGO: 0x0;
MODE: 0x60(CCM_INIT);
BS: 128;
KSS: 0(AES-128 mode);
CD: 0x0/0x01

5. Reseed and start operation by writing AES_CONTROL;**6. Poll AES_STATUS until the operation is completed;****7. Write formatted AAD block i to AES_DATA_I0 - AES_DATA_I3;****8. Configure AES_CONFIG:**

ALGO: 0x0;
MODE: 0x61(CCM_MAC);
BS: 128;
KSS: 0(AES-128 mode);
CD: 0x0/0x01

9. Reseed and start operation by writing AES_CONTROL;**10. Poll AES_STATUS until the operation is completed;**

11. Repeat step 7 through step 10 until the operation of all AAD blocks is completed;
12. If the AAD is empty, ignore step 7 to step 11;
13. Write formatted data block i to AES_DATA_I0 - AES_DATA_I3;
14. Configure AES_CONFIG:
ALGO: 0x0;
MODE: 0x62(CCM_UPDATE);
BS: 128(when the length of data block is 128 bits), actual data block length(when the length of data block less than 128 bits);
KSS: 0(AES-128 mode);
CD: 0x0/0x01
15. Reseed and start operation by writing AES_CONTROL;
16. Poll AES_STATUS until the operation is completed;
17. Read data block i from AES_DATA_O0 - AES_DATA_O3;
18. Repeat step 13 through step 17 until the operation of all data blocks is completed;
19. Configure AES_CONFIG:
ALGO: 0x0;
MODE: 0x63(CCM_FINISH);
BS: 128 or optional length of tag that user needs;
KSS: 0(AES-128 mode);
CD: 0x0/0x01
20. Reseed and start operation by writing AES_CONTROL;
21. Poll AES_STATUS until the operation is completed;
22. Read tag from AES_DATA_O0 - AES_DATA_O3.

12.3.8 CMAC

1. Write Key vector to AES_KEY0 – AES_KEY3;
2. Configure AES_CONFIG:
ALGO: 0x0;
MODE: 0x80(CMAC_INIT);
BS: 128;
KSS: 0(AES-128 mode);
CD: 0x0/0x01
3. Reseed and start operation by writing AES_CONTROL;
4. Poll AES_STATUS until the operation is completed;
5. Write data block i to AES_DATA_I0 - AES_DATA_I3;
6. Configure AES_CONFIG:
ALGO: 0x0;
MODE: 0x81(CMAC_UPDATE);
BS: 128;

KSS: 0(AES-128 mode);

CD: 0x0/0x01

7. Reseed and start operation by writing AES_CONTROL;
8. Poll AES_STATUS until the operation is completed;
9. Repeat step 5 through step 8 until the last data block remains;

10. Configure AES_CONFIG:

ALGO: 0x0;

MODE: 0x82 (If data is 128-bit aligned) / 0x83 (If data is not 128-bit aligned);

BS: 128 or optional length of tag that user needs;

KSS: 0(AES-128 mode);

CD: 0x0/0x01

11. Reseed and start operation by writing AES_CONTROL;
12. Poll AES_STATUS until the operation is completed;
13. Read mac from AES_DATA_O0 - AES_DATA_O3.

The user should perform LSB padding on data according to the CMAC algorithm when it is not 128-bit aligned.

12.3.9 XTS

1. Write tweak key vector to AES_KEY0 – AES_KEY3;
2. Write tweak to AES_IV0 – AES_IV3;
3. Configure AES_CONFIG:

ALGO: 0x0;

MODE: 0x70(XTS_TWEAK_ENC);

BS: 128;

KSS: 0(AES-128 mode);

CD: 0x0/0x01

4. Reseed and start operation by writing AES_CONTROL;
5. Poll AES_STATUS until the operation is completed;
6. Write data key vector to AES_KEY0 – AES_KEY3;
7. Write data block i to AES_DATA_I0 - AES_DATA_I3;
8. Configure AES_CONFIG:

ALGO: 0x0;

MODE: 0x71(XTS_INIT);

BS: 128;

KSS: 0(AES-128 mode);

CD: 0x0/0x01

9. Reseed and start operation by writing AES_CONTROL;
10. Poll AES_STATUS until the operation is completed;
11. Read data block i from AES_DATA_O0 - AES_DATA_O3;

12. Repeat step 7 through step 11 (configure AES_CONFIG[MODE] with XTS_UPDATE(0x72)) until the operation of all data blocks is completed.

12.3.10 Interrupts

The AES module can generate interrupt when operation is completed, software should check AES_STATUS[R] flag to verify that the operation is done when AES interrupt occurs.

Chapter 13

System Memory Protect Unit (SMPU)

13.1 Introduction

This chapter defines a System Memory Protect Unit (SMPU).

System Memory Protect Unit (SMPU) provides the memory region access protection at the level of AHB X-bar. Each AHB X-bar master access rights to each AHB X-bar slave ports can be defined in Memory Region Configuration Word 2 Register (SMPU_MRCn_W2) (n=0~15). When SMPU is enabled, memory access that has sufficient access control rights is allowed to complete, while memory access that has insufficient rights will be terminated with a protection error response.

13.1.1 SMPU slave port assignments

The memory-mapped resources protected by SMPU for Z20K14xMC series is shown as table 13-1.

Table 13-1 SMPU Slave Ports assignment for Z20K14xMC series

Source	SMPU slave port	Destination
Crossbar slave port 0	SMPU slave port 0	Flash controller
Crossbar slave port 1	SMPU slave port 1	SRAML
Crossbar slave port 2	SMPU slave port 2	SRAMU

The logic bus master assignment for Z20K14xMC series is shown as table 13-2.

Table 13-2 SMPU logic bus master assignment for Z20K14xMC series

SMPU logic bus master ID	Bus Master	Possible Access Type						
		User	Supervisor	Data	Instruction	Read	Write	Execute
0	Core	√	√	√	√	√	√*	√
1	debugger	√	√	√	√	√	√*	√
2	DMA	×	√	√	×	√	√*	×

Note: “√*” means that flash cannot be written by the corresponding master.

13.1.2 Features

The SMPU supports the following key features:

- Implement 16 independent memory region protect units to assign different access rights for different space regions.
 - Each memory region protect unit can define a protected memory space.
 - The defined memory space can overlap in different memory region protect units, access to the overlapped space is allowed to complete if there is at least one memory region protect unit defined with sufficient rights.
 - Each memory region protect unit can define the master read/write/execute rights in supervisor and user modes.
 - Each memory region protect unit has enable/disable control bit.
- If the access has insufficient rights, or the access address is not mapped to any memory region protect unit, the access will be terminated with a protection error response.
- SMPU will capture the last error access address, master/slave port number, error access

- attributes and other information.
- Implement global SMPU enable control bit.
- SMPU register can only be accessed in supervisor mode.

13.1.3 Block diagram

The following diagram is a block diagram of the System Memory Protect Unit (SMPU) module.

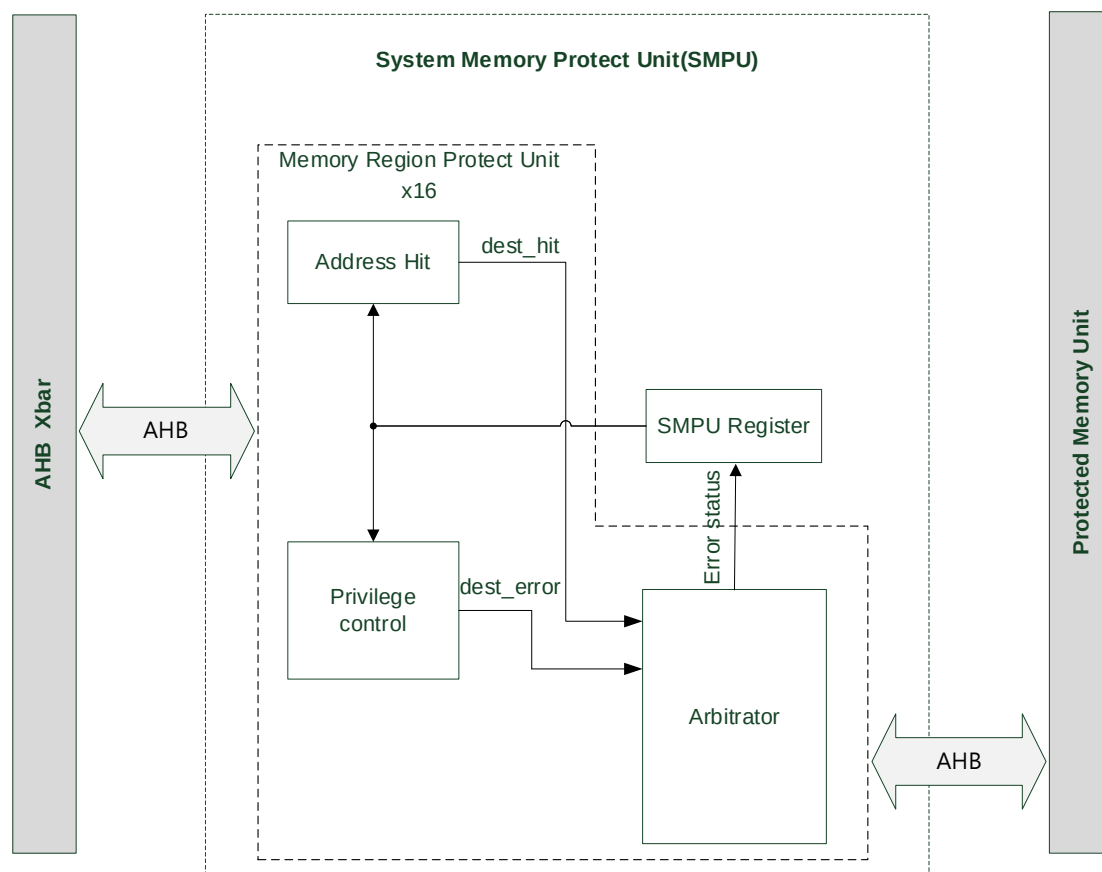


Figure 13-1 SMPU block diagram

13.2 Memory map and register definition

This section provides the memory map and register definitions of the SMPU module.

13.2.1 Memory map

SMPU base address: 0x4000A000

Table 13-3 SMPU memory map

Offset	Register	Width (In bits)	Reset Value
0x04	Control and Status Register (SMPU_CS)	32	0x00000000
0x10+4*n (n=0~2)	Slave Port n Error Address Register (SMPU_Sn_EA)	32	0x00000000
0x30+4*n (n=0~2)	Slave Port n Error Status Register (SMPU_Sn_ES)	32	0x00000000
0x100+0x10*n (n=0~15)	Memory Region n Configuration Word 0 Register	32	0x00000000

Offset	Register	Width (In bits)	Reset Value
	(SMPU_MRCn_W0)		
0x104+0x10*n (n=0~15)	Memory Region n Configuration Word 1 Register (SMPU_MRCn_W1)	32	0x0000001F
0x108+0x10*n (n=0~15)	Memory Region n Configuration Word 2 Register (SMPU_MRCn_W2)	32	0x00000000
0x10C+0x10*n (n=0~15)	Memory Region n Configuration Word 3 Register (SMPU_MRCn_W3)	32	0x00000000

13.2.2 Control and Status Register (SMPU_CS)

13.2.2.1 Offset

Register	Offset
SMPU_CS	0x04

13.2.2.2 Fields

Field	Access	Description	Reset Value
0 S0ERR	W1C	This bit indicates the slave port 0 access error is captured. This bit is automatically set by hardware when slave port 0 access error occurs, writing 1 to this field will clear this bit. 0b: Slave port 0 access error didn't occur. 1b: Slave port 0 access error occurred.	0x0
1 S1ERR	W1C	This bit indicates the slave port 1 access error is captured. This bit is automatically set by hardware when slave port 1 access error occurs, writing 1 to this field will clear this bit. 0b: Slave port 1 access error didn't occur. 1b: Slave port 1 access error occurred.	0x0
2 S2ERR	W1C	This bit indicates the slave port 2 access error is captured. This bit is automatically set by hardware when slave port 2 access error occurs, writing 1 to this field will clear this bit. 0b: Slave port 2 access error didn't occur. 1b: Slave port 2 access error occurred.	0x0
7:3 RSVD_7_3	RO	Reserved	0x0
8 GLBEN	RW	Global SMPU enable 0b: SMPU is disabled. 1b: SMPU is enabled.	0x0
31:9 RSVD_31_9	RO	Reserved	0x0

13.2.3 Slave Port n Error Address Register (SMPU_Sn_EA)

13.2.3.1 Offset

Register	Offset
SMPU_Sn_EA	$0x10+4*n(n=0\sim2)$

13.2.3.2 Function

This register records the captured slave port n error information.

13.2.3.3 Fields

Field	Access	Description	Reset Value
31:0 EADDR	RO	This field indicates the captured slave port n error access address.	0x0

13.2.4 Slave Port n Error Status Register (SMPU_Sn_ES)

13.2.4.1 Offset

Register	Offset
SMPU_Sn_ES	$0x30+4*n(n=0\sim2)$

13.2.4.2 Fields

Field	Access	Description	Reset Value
15:0 EMRC	RO	This field records the error is detected by which MRC (memory region configuration). EMRC bit n indicates error detected by MRCn(n=0~15).	0x0
23:16 RSVD_23_16	RO	Reserved	0x0
27:24 EMN	RO	Indicate the bus master that causes the access error. 00b: Bus master 0 causes the access error. 01b: Bus master 1 causes the access error. 10b: Bus master 2 causes the access error. 11b: Reserved.	0x0
28 EATTR	RO	Indicate the error type is instruction or data access. 0b: Instruction access error. 1b: Data access error.	0x0
29 EUS	RO	Indicate the error type is user or supervisor access. 0b: User mode access error. 1b: Supervisor mode access error.	0x0
30 RSVD_30	RO	Reserved	0x0
31 ERW	RO	Indicate the error type is read or write access. 0b: Read access error. 1b: Write access error.	0x0

13.2.5 Memory Region n Configuration Word 0 Register (SMPU_MRCn_W0)

13.2.5.1 Offset

Register	Offset
SMPU_MRCn_W0	0x100+0x10*n(n=0~15)

13.2.5.2 Fields

Field	Access	Description	Reset Value
4:0 RSVD_4_0	RO	Reserved	0x0
31:5 SADDR	RW	This field defines the most significant bit of start address protected by MRCn(n=0~15).	0x0

13.2.6 Memory Region n Configuration Word 1 Register (SMPU_MRCn_W1)

13.2.6.1 Offset

Register	Offset
SMPU_MRCn_W1	0x104+0x10*n(n=0~15)

13.2.6.2 Fields

Field	Access	Description	Reset Value
4:0 RSVD_4_0	RW	Reserved	0x1F
31:5 EADDR	RW	This field defines the most significant bit of end address protected by MRCn(n=0~15).	0x0

13.2.7 Memory Region n Configuration Word 2 Register (SMPU_MRCn_W2)

13.2.7.1 Offset

Register	Offset
SMPU_MRCn_W2	0x108+0x10*n (n=0~15)

13.2.7.2 Fields

Field	Access	Description	Reset Value
0 MOUX	RW	This bit defines the execute access control for bus master 0 in user mode. 0b: Execute is not allowed. 1b: Execute is allowed.	0x0
1 MOUW	RW	This bit defines the write access control for bus master 0 in user mode. 0b: Write is not allowed. 1b: Write is allowed.	0x0

Field	Access	Description	Reset Value
2 M0UR	RW	This bit defines the read access control for bus master 0 in user mode. 0b: Read is not allowed. 1b: Read is allowed.	0x0
4:3 M0SM	RW	This field defines the access control for bus master 0 in supervisor mode. 00b: R/W/X, read/write/execute are all allowed. 01b: R/X, read/execute are allowed, write is not allowed. 10b: R/W, read/write are allowed, execute is not allowed. 11b: Access controls are same with user mode.	0x0
5 RSVD_5	RO	Reserved	0x0
6 M1UX	RW	This bit defines the execute access control for bus master 1 in user mode. 0b: Execute is not allowed. 1b: Execute is allowed.	0x0
7 M1UW	RW	This bit defines the write access control for bus master 1 in user mode. 0b: Write is not allowed. 1b: Write is allowed.	0x0
8 M1UR	RW	This bit defines the read access control for bus master 1 in user mode. 0b: Read is not allowed. 1b: Read is allowed.	0x0
10:9 M1SM	RW	This field defines the access control for bus master 1 in supervisor mode. 00b: R/W/X, read/write/execute are all allowed. 01b: R/X, read/execute are allowed, write is not allowed. 10b: R/W, read/write are allowed, execute is not allowed. 11b: Access controls are same with user mode.	0x0
11 RSVD_11	RO	Reserved	0x0
12 M2UX	RW	This bit defines the execute access control for bus master 2 in user mode. 0b: Execute is not allowed. 1b: Execute is allowed.	0x0
13 M2UW	RW	This bit defines the write access control for bus master 2 in user mode. 0b: Write is not allowed. 1b: Write is allowed.	0x0
14 M2UR	RW	This bit defines the read access control for bus master 2 in user mode. 0b: Read is not allowed. 1b: Read is allowed.	0x0
16:15 M2SM	RW	This field defines the access control for bus master 2 in supervisor mode. 00b: R/W/X, read/write/execute are all allowed. 01b: R/X, read/execute are allowed, write is not allowed. 10b: R/W, read/write are allowed, execute is not allowed. 11b: Access controls are same with user mode.	0x0

Field	Access	Description	Reset Value
31:17 RSVD_31_17	RO	Reserved	0x0

13.2.8 Memory Region n Configuration Word 3 Register (SMPU_MRCn_W3)

13.2.8.1 Offset

Register	Offset
SMPU_MRCn_W3	0x10C+0x10*n(n=0~15)

13.2.8.2 Fields

Field	Access	Description	Reset Value
15:0 RSVD_15_0	RO	Reserved	0x0
16 EN	RW	This bit defines whether such memory region configuration is enabled. 0b: Memory region n configuration is disabled. 1b: Memory region n configuration is enabled.	0x0
31:17 RSVD_31_17	RO	Reserved	0x0

13.3 Function description

This section provides the functional information of the SMPU module.

13.3.1 Memory region protect unit

The SMPU implements 16 independent memory region protect units to assign different access rights for different memory regions. Memory region protect unit n (n=0~15) is configured by SMPU_MRCn_W0~SMPU_MRCn_W3 registers. Memory region protect unit n is enabled only when SMPU_MRCn_W3[EN] bit is set.

Every memory region protect unit includes two parts, address hit and privilege control.

13.3.2 Memory protect address

The addresses protected by the SMPU are as follows:

- 128K SRAM_L, from 0x1FFE_0000 to 0x1FFF_FFFF
- 128K SRAM_U, from 0x2000_0000 to 0x2001_FFFF
- 2M program flash, from 0x0000_0000 to 0x001F_FFFF
- 128K data flash, from 0x0100_0000 to 0x0101_FFFF
- 32KB program flash IFR, from 0x0200_0000 to 0x0200_7FFF

Note: IP registers' protection can only be implemented by PARCC, not SMPU.

13.3.3 Bus master

In SMPU, the bus masters are defined as follows:

- Bus master 0 indicates the core
- Bus master 1 indicates the debug

- Bus master 2 indicates the DMA

NOTE

The bus master here is not the same as the master port in Figure 2-1.

13.3.4 Slave port

In SMPU, the slave ports are defined as follows:

- Slave port 0 indicates the flash
- Slave port 1 indicates the SRAML
- Slave port 2 indicates the SRAMU

13.3.5 Address hit

SMPU_MRCn_W0 register configures the start address of protected memory region, SMPU_MRCn_W1 register configures the end address of protected memory region. When master accessed slave memory address is mapped to this memory region, that is, $\text{SMPU\_MRCn\_W0[SADDR]} \leq \text{ACCESS\_ADDR}[31:5] \leq \text{SMPU\_MRCn\_W1[EADDR]}$, then address is hit.

NOTE

SADDR and EADDR only define the most significant bit [31:5] of start and end address. If $\text{SMPU\_MRCn\_W0[SADDR]} > \text{SMPU\_MRCn\_W0[EADDR]}$, no any address will hit.

13.3.6 Privilege control

The privilege control unit will determine the current access is allowed or not allowed by the permissions defined by SMPU_MRCn_W2 register. Read, write and execute access rights of all bus masters in supervisor and user modes can be configured.

In SMPU_MRCn_W2 register, MxUR/MxUW/MxUX bits define the bus master x (x = 0~2) read/write/execute rights in user mode. The following table shows the master access permission in user mode.

Table 13-4 Master access permission in user mode

Master Access Type	MxUR	MxUW	MxUX	Access Is Allowed?
Data Read	0	-	-	No
	1	-	-	Yes
Data Write	-	0	-	No
	-	1	-	Yes
Instruction Fetch Read	-	-	0	No
	-	-	1	Yes

In SMPU_MRCn_W2 register, MxSM [1:0] field defines the bus master x access rights in supervisor mode. The following table shows the master access permission in supervisor mode.

Table 13-5 Master access permission in supervisor mode

MxSM[1:0]	Master Access Is Allowed?		
	Read Data	Write Data	Instruction Fetch Read
00	Yes	Yes	Yes
01	Yes	No	Yes
10	Yes	Yes	No

MxSM[1:0]	Master Access Is Allowed?		
	Read Data	Write Data	Instruction Fetch Read
11	same with user mode, determined by MxUR/MxUW/MxUX		

13.3.7 Access arbitration

The access arbitration module will put all memory region protect unit outputs together and determine if the access is allowed by SMPU. After SMPU global enable bit SMPU_CS[GLBEN] is set, the SMPU will monitor all bus master access to each slave port. If the access is not allowed, the SMPU will terminate this access with a protection error response to master. In following three conditions, the access is not allowed.

- The access address does not hit any memory region protect unit.
- The access address hits a single memory region protect unit, and the access is not allowed by this single memory region protect unit.
- The access address hits multiple memory region protect units, and the access is not allowed by all these memory region protect units.

NOTE

If the access address hits multiple memory region protect units, but the access is allowed by some units and not allowed by some units, the privilege arbitration result is the access that is allowed by SMPU. When the access is terminated by SMPU, the SMPU will feed back a protection error response to master by AHB bus signals.

13.3.8 Error report

When the access is terminated by SMPU, the SMPU will capture the error status and record the last error access information in SMPU_Sn_EA and SMPU_Sn_ES (n=0~2) registers. The error information includes master port number, slave port number, error access address, supervisor or user mode access, access attributes, and which memory protect unit to capture this error access. If the error access hits multiple memory region protect units, SMPU_Sn_ES[EMRC] (n=0~2) multiple bits will assert.

13.4 Initialization application information

To protect a memory space, the SMPU should be initialized with following steps:

1. Select one memory region protect unit to allow code to execute in flash or sram, such as select memory region protect unit 0.
 - a) Write SMPU_MRC0_W0 and SMPU_MRC0_W1 registers to configure code execute memory region.
 - b) Write SMPU_MRC0_W2 register to allow instruction fetch read by core.
 - c) Write SMPU_MRC0_W3 register to enable this memory region protect unit.
2. Select one memory region protect unit to allow stack and variables to be stored in sram.
3. Select one memory region to protect expected memory space.
4. Write SMPU_CS register to enable GLBEN bit.

Chapter 14

Watchdog Timer (WDOG)

14.1 Introduction

The Watchdog Timer (WDOG) is an independent timer for monitoring system behavior. It provides a safety measure that ensures the software is running as planned and the CPU is not stuck in an infinite loop or running in an unintended code. If the WDOG module is not refreshed within a certain period, it resets the CPU.

14.2 Features

The WDOG supports the following key features:

- Configurable clock source to drive counter:
 - OSC Clock
 - FIRC Clock
 - BUS Clock
 - LPO Clock
- Programmable timeout period
 - Support 32-bit timeout value
- Robust refresh mechanism for counter clear
 - Refresh sequence of writing 0xA0C4B1D6 and then 0x1E0D0C7B
- Optional window mode for the acceptance of refresh sequence
 - Support 32-bit window value
 - Refresh the operation that is not within window and trigger a reset
 - Monitor the program execution that is faster than expected
- Optional interrupt before triggering CPU reset to allow post-processing for handling critical schemes
 - Interrupt request to CPU with interrupt vector for an interrupt service routine (ISR)
 - Forced reset occurs 256 bus clocks after the interrupt vector is fetched
- After unlocking, the configuration registers are write-once registers to make sure the watchdog configuration cannot be overwritten by mistake
- Robust unlock mechanism for reconfiguration without resetting
 - Unlock sequence of writing 0xB0D9A1C4 and then 0x1A1E3B0F for unlocking the locked configuration bits
 - Software must complete updates within 128 bus clocks after unlocking. WDOG automatically locks all the configuration bits once the window counter reaches to 128.
 - Each register can be written once within 128 bus clocks after unlocking.

14.2.1 Modes of operation

The WDOG operates in the following modes:

- DEBUG mode:
 - WDOG operation is configurable in DEBUG mode by WDOG_CS register:
 - If WDOG_CS[DEBUGE] is cleared, the WDOG counter halted in the DEBUG mode.
 - If WDOG_CS[DEBUGE] is set, the WDOG counter continues to count up in a DEBUG mode.

- **WAIT mode:**
WDOG operation is configurable in WAIT mode by the WDOG_CS register:
 - If WDOG_CS[WAITE] is cleared, the WDOG counter halted in the WAIT mode.
 - If WDOG_CS[WAITE] is set, the WDOG counter continues to count up in the WAIT mode.
- **STOP mode:**
If the selected function clock source remains enabled, WDOG operation is configurable in the STOP mode by the WDOG_CS register:
 - If WDOG_CS[STOPE] is cleared, the WDOG counter halted in the STOP mode.
 - If WDOG_CS[STOPE] is set, the WDOG counter continues to count up in the STOP mode.

NOTE

To make the WDOG keeps on working, it is necessary to ensure that the clock source selected by WDOG_CS[CLKS] is active in both WAIT and STOP modes.

14.2.2 Block diagram

The following figure is a block diagram of the Watchdog Timer (WDOG) module.

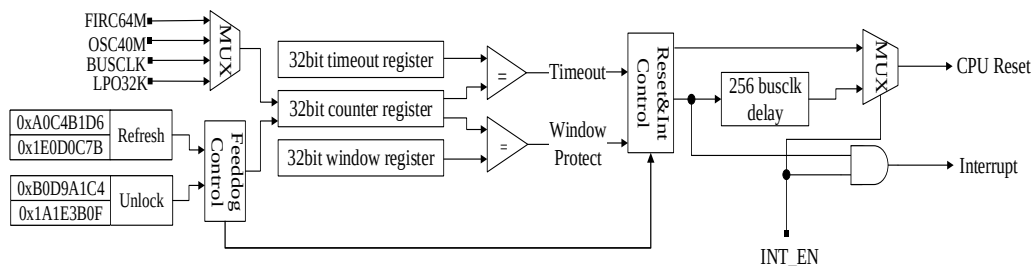


Figure 14-1 WDOG block diagram

14.3 Memory map and register definition

This section provides the memory map and register definitions of the WDOG module.

14.3.1 Memory map

WDOG base address: 0x40052000

Table 14-1 WDOG memory map

Offset	Register	Width (In bits)	Reset Value
0x4	Watchdog Parameter (WDOG_PARAM)	32	0x00000001
0x8	Watchdog Control and Status Register (WDOG_CS)	32	0x01040041
0xC	Watchdog Timeout Register (WDOG_TMO)	32	0x0001FFFF
0x10	Watchdog Window Register (WDOG_WIN)	32	0x00000000
0x14	Watchdog Counter Register (WDOG_CNT)	32	0x00000000

14.3.2 Watchdog Parameter (WDOG_PARAM)

14.3.2.1 Offset

Register	Offset
WDOG_PARAM	0x4

14.3.2.2 Function

This register is used to indicate design feature implementation.

14.3.2.3 Fields

Field	Access	Description	Reset Value
0 INT_PRESENT	RO	0b: No interrupt function. 1b: Interrupt function is available.	0x1
31:1 RSVD_31_1	RO	Reserved	0x0

14.3.3 Watchdog Control and Status Register (WDOG_CS)

14.3.3.1 Offset

Register	Offset
WDOG_CS	0x8

14.3.3.2 Function

The register defines the basic operation configuration of the WDOG, including function clock source selection, WDOG behavior in debug mode and low power mode, etc.

14.3.3.3 Fields

Field	Access	Description	Reset Value
0 WDOGE	RW	This bit enables the watchdog. 0b: Watchdog disabled. 1b: Watchdog enabled.	0x1
1 DEBUGE	RW	This bit enables the watchdog counter to count up in debug mode. 0b: Watchdog counter halted in debug mode. 1b: Watchdog counter continues to count up in debug mode.	0x0
2 WAITE	RW	This bit enables the watchdog counter to count up in WAIT mode. 0b: Watchdog counter halted in WAIT mode. 1b: Watchdog counter continues to count up in WAIT mode.	0x0
3 STOPE	RW	This bit enables the watchdog counter to count up in STOP mode. 0b: Watchdog counter halted in STOP mode. 1b: Watchdog counter continues to count up in STOP mode.	0x0
4 INTE	RW	This bit enables the watchdog to generate an interrupt request before forcing a reset. 0b: Interrupt is disabled. 1b: Interrupt is enabled.	0x0

Field	Access	Description	Reset Value
5 WINE	RW	This bit enables window mode. 0b: Window mode is disabled. 1b: Window mode is enabled.	0x0
6 CFGUA	RW	This bit allows software to update configuration without a reset. 0b: Update not allowed. The configuration cannot be modified again once the initial configuration is done. 1b: Update allowed. Registers can be modified after the unlock write sequence is executed.	0x1
7 RSVD_7	RO	Reserved	0x0
9:8 CLKS	RW	The bits are used to select clock source to drive the watchdog counter. 00b: LPO clock 01b: BUS clock 10b: FIRC64M clock 11b: OSC40M clock	0x0
15:10 RSVD_15_10	RO	Reserved	0x0
18:16 TSTM	RW	000b: Test mode0 is enabled. 001b: Test mode1 is enabled. 010b: Test mode2 is enabled. 011b: Test mode3 is enabled. 100b: Normal mode is enabled. Software should use this value to indicate that the watchdog is running in normal function mode. Note: This field becomes 100b after a power-on reset, and is not affected by other resets.	0x4
23:19 RSVD_23_19	RO	Reserved	0x0
24 CFGUF	RO	0b: New configurations update is on-going. 1b: New configurations take effect. When writing unlock sequence to WDOG_CNT, this field becomes 0b. After 128 bus clocks plus 4 watchdog clocks, this field becomes 1b automatically.	0x1
25 UNLKF	RO	0b: All registers are locked. 1b: All registers are unlocked. When writing unlock sequence to WDOG_CNT, this field becomes 1b. After 128 bus clocks, this field becomes 0b automatically.	0x0
26 INTF	W1C	0b: No interrupt occurred. 1b: An interrupt occurred.	0x0
31:27 RSVD_31_27	RO	Reserved	0x0

14.3.4 Watchdog Timeout Register (WDOG_TMO)

14.3.4.1 Offset

Register	Offset
WDOG_TMO	0xC

14.3.4.2 Function

This register is used to configure the timeout value.

14.3.4.3 Fields

Field	Access	Description	Reset Value
31:0 TMO	RW	The value of watchdog counter(WDOG_CNT) is compared with the timeout value(WDOG_TMO) continuously. If the counter value reaches the timeout value, the watchdog generates a reset event. To balance the coverage and time of a test, it is recommended to configure the timeout register with 0xFFFFFFFF in any test mode.	0x1FFFF

14.3.5 Watchdog Window Register (WDOG_WIN)

14.3.5.1 Offset

Register	Offset
WDOG_WIN	0x10

14.3.5.2 Function

This register is used to configure the window value.

14.3.5.3 Fields

Field	Access	Description	Reset Value
31:0 WIN	RW	If the window mode is enabled (WDOG_CS[WINE]=1b), a refresh sequence is considered valid if it occurs when WDOG_CNT is greater than WDOG_WIN. Otherwise, it is an invalid refresh sequence which will result in a reset event.	0x0

14.3.6 Watchdog Counter Register (WDOG_CNT)

14.3.6.1 Offset

Register	Offset
WDOG_CNT	0x14

14.3.6.2 Function

This register is counter register.

14.3.6.3 Fields

Field	Access	Description	Reset Value
31:0 CNT	RW	This register provides a read to the value of the free-running watchdog counter. This register cannot be written directly. But two write	0x0

Field	Access	Description	Reset Value
		sequences are allowed for special functions. 1. The refresh sequence is used to reset the counter to 0. The refresh sequence can be two 32-bit writes (0xA0C4B1D6, 0x1E0D0C7B) to the WDOG_CNT. 2. The unlock sequence is used to unlock registers for reconfiguration. The unlock sequence can be two 32-bit writes (0xB0D9A1C4, 0x1A1E3B0F) to the WDOG_CNT.	

14.4 Function description

This section provides the functional information of the WDOG module. The WDOG module has a fail-safe feature to guarantee that the system can be reset to a known state of operation in case of system failure, such as CPU clock stops or there is a runaway condition in the software code which is detected. The watchdog counter runs continuously off a selectable clock source and expects to be serviced (refreshed) periodically. Otherwise, it generates a reset triggering event.

14.4.1 Watchdog refresh mechanism

The watchdog has a robust refresh mechanism with writing predefined magic words according to the predetermined order within fixed window time. This watchdog refresh mechanism makes sure that the watchdog will not be refreshed by runaway code. The watchdog refresh mechanism resets the MCU if the watchdog counter is not refreshed before reaching the timeout value.

Software must perform a refresh write sequence to refresh the watchdog counter before the timeout period expires. Besides, software could perform the refresh sequence after the counter register value is greater than the value in the WDOG_WIN register if window mode is enabled.

14.4.1.1 Window mode

When the software completes the main control loop faster than expected, this could indicate the system faults. By taking this into consideration, the WDOG can be programmed to force a reset when refresh attempts are early.

When the window mode is enabled, the watchdog refresh write must be performed after the counter has reached the value of WDOG_WIN register. Otherwise, the watchdog resets the MCU. Setting WDOG_CS[WINE] enables the window mode.

14.4.1.2 Refreshing the watchdog

The refresh write sequence is writing 0xA0C4B1D6 and then 0x1E0D0C7B to the WDOG_CNT register. The refresh write must occur before the WDOG times out. Otherwise, the watchdog resets the MCU. Writing wrong refresh magic words will cause the watchdog to generate a reset.

14.4.2 Configure the watchdog

14.4.2.1 Configure the watchdog once after reset

All watchdog registers, including WDOG_CS, WDOG_TMO, and WDOG_WIN, are locked after reset. The software can perform an unlock sequence to make them writable. The user must ensure that WDOG_CS[CFGUA] is set to 0. The unlock sequence guarantees that the configuration will not be modified again unless a reset occurs.

14.4.2.2 Reconfigure the watchdog

In some scenarios, the user may want to reconfigure the watchdog without forcing a reset first. The user can reconfigure the watchdog at any time by executing an unlock sequence by setting WDOG_CS[CFGUA] to 1 on the watchdog's initial configuration after a reset.

14.4.2.3 Unlock the watchdog

The unlock sequence is a write to the WDOG_CNT register of 0xB0D9A1C4 followed by 0x1A1E3B0F at any time after configuring the watchdog. The user must reconfigure the watchdog within 128 bus clocks after completing unlock sequence. Otherwise, the watchdog closes the unlock window and locks all of registers. Writing wrong and unlock magic word causes watchdog to generate reset.

14.4.3 Interrupt

When an interrupt is enabled (WDOG_CS[INTE] = 1), the watchdog first produces an interrupt request after a reset-triggering event. After that, the watchdog delays 256 bus clock cycles before generating a reset to allow the interrupt service routine (ISR) to execute codes. When interrupt is disabled (WDOG_CS[INTE] = 0), the watchdog immediately generates a reset without delay once a reset-triggering event occurs.

14.4.4 Relief reset

The relief reset function is a safeguard feature that independently generates a reset if the main WDOG logic loses its clock (the bus clock) and can no longer make timeout comparisons and monitor feed dog operation.

If the watchdog counter overflows twice consecutively (without an intervening reset), the relief reset function takes effect and generates a reset.

The relief reset becomes valid when an interrupt is enabled. If an interrupt is enabled, the normal watchdog reset will be blocked when the bus clock is cut off prior to exiting the interrupt routine. In this case, the second overflow will cause a relief reset directly.

14.4.5 Quick testing of the watchdog counter

The quick testing feature allows the watchdog to shorten the test time before the application code starts up. It reduces the test time to 512 clocks (not including overhead, such as user configuration and reset vector fetches). Using a faster clock (such as the bus clock) for the counter drive can further speed up the testing.

The watchdog splits the counter into its constituent byte-wide stages when this feature is enabled.

On a power-on reset, the POR bit in the system reset register is set, indicating the software should perform the WDOG fast test.

Select one from the four modes by configuring WDOG_CS[TSTM] to test each byte.

The following descriptions show how the watchdog counter behaves in different test modes:

- In test mode 0, byte1 and byte0 run as a 16-bit counter and start from 0xFE00, the value of byte3 and byte2 is frozen at 0xFF.
- In test mode 1, byte2 and byte1 run as a 16-bit counter and start from 0xFE00, the value of byte3 and byte0 is frozen at 0xFF.
- In test mode 2, byte3 and byte2 run as a 16-bit counter and start from 0xFE00, the value of byte1 and byte0 is frozen at 0xFF.
- In test mode 3, byte3 runs as an 8-bit counter from 0x00, the value of byte2, byte1 and byte0 is frozen at 0xFF.

In any test mode, the watchdog compares the 32-bit counter against the 32-bit timeout value register. Software should configure the timeout register with a proper value before testing. To balance the coverage and time of a test, it is recommended to configure the timeout register

with 0xFFFFFFFF in any test mode.

14.4.5.1 Test each byte of the counter

The following is the test procedure:

1. Configure the WDOG_TMO register with 0xFFFFFFFF.
2. Configure the WDOG_CS[TSTM] with 000b to select the test mode 0.
3. Wait for the watchdog time out and force a reset.
4. After reset, configure the WDOG_TMO register with 0xFFFFFFFF and configure the WDOG_CS[TSTM] with 001b to select the test mode 1.
5. Wait for the watchdog time out and force a reset.
6. After reset, configure the WDOG_TMO register with 0xFFFFFFFF and configure the WDOG_CS[TSTM] with 010b to select the test mode 2.
7. Wait for the watchdog time out and force a reset.
8. After reset, configure the WDOG_TMO register with 0xFFFFFFFF and configure the WDOG_CS[TSTM] with 011b to select the test mode 3.
9. Wait for the watchdog time out and force a reset.
10. After reset, configure WDOG_CS[TSTM] with 100b to enter normal mode. Then users can configure WDOG according to actual needs.

14.4.6 Clock source

The watchdog counter has the following clock source options selected by programming WDOG_CS[CLKS]:

- OSC40M clock
- FIRC64M clock
- BUS clock
- LPO clock

These options allow the software to select a clock source independent of the bus clock for applications that need to meet more robust safety requirements. Using a clock source other than the bus clock ensures that the watchdog counter continues to run if the bus clock is somehow stopped.

14.5 Application information

This section provides the application information of the WDOG module.

14.5.1 WDOG application sequence

The WDOG application follows the following sequence:

1. Write an unlock sequence to the WDOG_CNT register to unlock WDOG registers.
2. Write to the WDOG_CS register to define WDOG operating configuration, including the function clock source selection, WDOG behavior in debug mode and low-power modes, window mode control, interrupt mask, etc.
3. Write to the WDOG_TMO register to set the timeout period of the WDOG.
4. If window mode is enabled, write to the WDOG_WIN register to set the window value. Note that the software must reconfigure the watchdog within 128 bus clocks after completing the unlock sequence.
5. Set the WDOG_CS[WDOGE] field to enable the WDOG counter.
6. Periodically perform a refresh operation to reset the WDOG counter when its count value is between window value and a timeout value.

Chapter 15

External Watchdog Timer (EWDT)

15.1 Introduction

In some safety-critical applications, it is necessary to monitor external circuits and ensure they enter into safe states when there is application code runaway or the execution flow is not running as planned.

External Watchdog Timer (EWDT) can solve this problem. It detects the abnormal behavior of MCU by monitoring the software flow, similar to the internal watchdog (WDOG).

Unlike the WDOG, the EWDT does not reset the CPU and peripherals of the MCU. Instead, the EWDT provides an independent EWDT_OUT signal which resets or places an external circuit into a safe state when asserted. The EWDT_OUT signal is asserted upon the EWDT counter time-out or error service detection. An optional external input EWDT_IN allows additional control of the assertion of the EWDT_OUT signal.

15.1.1 Features

The EWDT supports the following key features:

- Configurable clock source to drive counter:
 - See the PARCC section for details
- Programmable time-out period
 - Support 32-bit time-out value
- Programmable service window
 - Support 32-bit window value
 - Provide robust check that program flow is faster than expected
 - Service outside window leads to the assertion of EWDT_OUT
- Robust service mechanism
 - Write values of 0xB2C5D3A8 and 0x1D7E4C5F to EWDT Counter Register within 16 bus clock cycles
- One output port, the EWDT_OUT, is used to reset or place the external circuit into a safe state when asserted
- One input port, the EWDT_IN, allows an external circuit to control the assertion of the EWDT_OUT

15.1.2 Modes of operation

The EWDT operates in the following modes:

- **DEBUG mode:**
EWDT operation is configurable in DEBUG mode by the EWDT_CS register:
 - If EWDT_CS[DEBUGE] is cleared, the EWDT counter halted in the DEBUG mode
 - If EWDT_CS[DEBUGE] is set, the EWDT counter continues to count up in the DEBUG mode
- **WAIT mode:**
EWDT operation is configurable in the WAIT mode by the EWDT_CS register:
 - If EWDT_CS[WAITE] is cleared, the EWDT counter halted in the WAIT mode
 - If EWDT_CS[WAITE] is set, the EWDT counter continues to count up in the WAIT mode
- **STOP mode:**
EWDT is disabled in the STOP mode

NOTE

To make sure the EWDT keeps working, ensure the function clock source of EWDT is active in the WAIT mode.

15.1.3 Block diagram

The following figure is a block diagram of the External Watchdog Timer (EWDT) module.

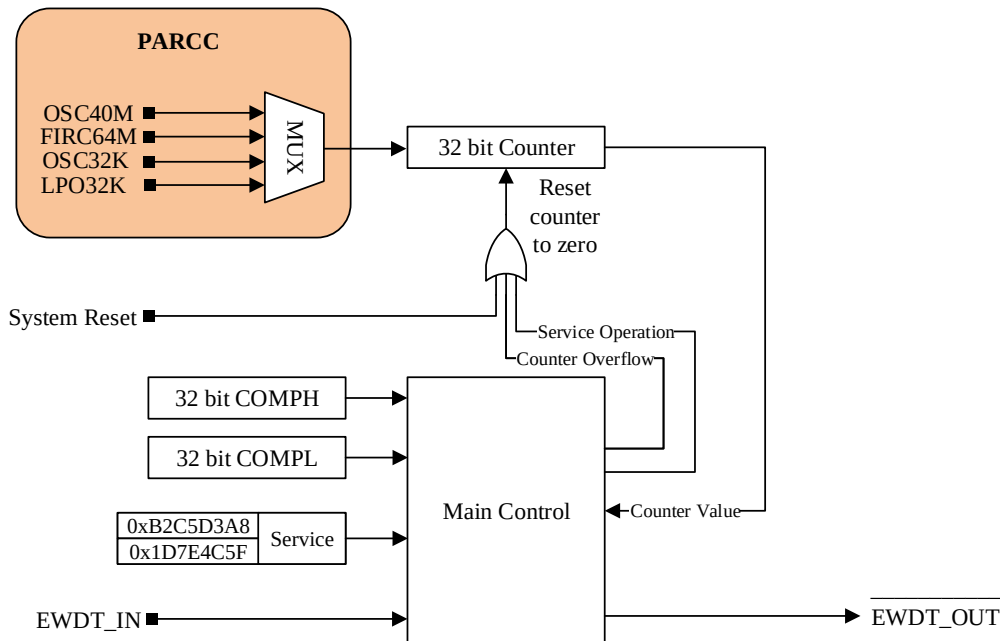


Figure 15-1 EWDT block diagram

15.2 Signal description

The EWDT module has two I/O signals connected to the external chip pins. These signals are summarized in the following table.

Table 15-1 EWDT signal descriptions

Signal	Description	I/O
EWDT_OUT	EWDT reset output signal	Output
EWDT_IN	EWDT input for safety control from external circuits.	Input

15.2.1 The EWDT_OUT signal

The EWDT_OUT is a digital output signal used to place an external circuit (application customization) in a safe state. 0 state is EWDT_OUT_b assertion, and 1 state is EWDT_OUT_b deassertion.

15.2.2 The EWDT_IN signal

The EWDT_IN is a digital input signal for safety control from external safety circuits, which allows an external circuit to control the assertion of the EWDT_OUT signal. The active polarity of EWDT_IN is programmable using the EWDT_CS [INPP] bit. The default active polarity is low.

15.3 Memory map and register definition

This section provides the memory map and register definitions of the EWDT module.

15.3.1 Memory map

EWDT base address: 0x40061000

Table 15-2 EWDT memory map

Offset	Register	Width (In bits)	Reset Value
0x4	External Watchdog Timer Version Parameter Register (EWDT_PARAM)	32	0x00000000
0x8	External Watchdog Timer Control and Status Register (EWDT_CS)	32	0x00000000
0xC	External Watchdog Timer Compare High Register (EWDT_COMPH)	32	0xFFFFFFFF
0x10	External Watchdog Timer Compare Low Register (EWDT_COMPL)	32	0x00000000
0x14	External Watchdog Timer Counter Register (EWDT_CNT)	32	0x00000000

15.3.2 External Watchdog Timer Version Parameter Register (EWDT_PARAM)

15.3.2.1 Offset

Register	Offset
EWDT_PARAM	0x4

15.3.2.2 Function

This register is for indicating the design feature.

15.3.2.3 Fields

Field	Access	Description	Reset Value
31:0 RSVD_31_0	RO	Reserved	0x0

15.3.3 External Watchdog Timer Control and Status Register (EWDT_CS)

15.3.3.1 Offset

Register	Offset
EWDT_CS	0x8

15.3.3.2 Function

This register is for indicating the configuration and status.

15.3.3.3 Fields

Field	Access	Description	Reset Value
0 EWDTE	RW	This write-once bit enables the EWDT. 0b: EWDT disabled. 1b: EWDT enabled.	0x0
1 DEBUGE	RW	This write-once bit enables the EWDT counter to count up in debug mode. 0b: EWDT counter halted in the debug mode. 1b: EWDT counter continues to count up in the debug mode.	0x0
2 WAITE	RW	This write-once bit enables the EWDT counter to count up in WAIT mode. 0b: EWDT counter halted in the WAIT mode. 1b: EWDT counter continues to count up in the WAIT mode.	0x0
3 RSVD_3	RO	Reserved	0x0
4 INTE	RW	This is write-once bit. This write-once bit enables the EWDT to generate an interrupt request when the EWDT_OUT_b is asserted.	0x0
5 INPE	RW	This write-once bit enables the input port. 0b: Disable EWDT_IN port. 1b: Enable EWDT_IN port.	0x0
6 INPP	RW	This write-once bit programs the input port polarity. 0b: 0 state is active assertion of EWDT_IN. 1b: 1 state is active assertion of EWDT_IN.	0x0
23:7 RSVD_23_7	RO	Reserved	0x0
24 INTF	W1C	Flag indicates an interrupt request is generated because the EWDT_OUT_b is asserted. 0b: No interrupt occurred. 1b: An interrupt occurred.	0x0
31:25 RSVD_31_25	RO	Reserved	0x0

15.3.4 External Watchdog Timer Compare High Register (EWDT_COMPH)

15.3.4.1 Offset

Register	Offset
EWDT_COMPH	0xC

15.3.4.2 Function

This register is for determining the high threshold for the counter comparison.

15.3.4.3 Fields

Field	Access	Description	Reset Value
31:0 COMPH	RW	The COMPH register can be written once only after a reset to prevent runaway code from changing it. This register value determines the latest refresh time.	0xFFFFFFFF

15.3.5 External Watchdog Timer Compare Low Register (EWDT_COMPL)

15.3.5.1 Offset

Register	Offset
EWDT_COMPL	0x10

15.3.5.2 Function

This register is for determining the low threshold for the counter comparison.

15.3.5.3 Fields

Field	Access	Description	Reset Value
31:0 COMPL	RW	The COMPL register can be written only once after a reset to prevent runaway code from changing it. This register value determines the earliest refresh time.	0x0

15.3.6 External Watchdog Timer Counter Register (EWDT_CNT)

15.3.6.1 Offset

Register	Offset
EWDT_CNT	0x14

15.3.6.2 Function

This is a counter register.

15.3.6.3 Fields

Field	Access	Description	Reset Value
31:0 CNT	RW	This register provides a read to the value of the free-running EWDT counter. This register cannot be written directly. However, a write sequence named refresh is allowed for resetting the counter to zero. The refresh sequence can be two 32-bit writes (0xB2C5D3A8, 0x1D7E4C5F) to the EWDT_CNT. The refresh sequence writes will occur when the counter value is within the value of COMPH and COMPL, and EWDT_IN is not asserted if INPE is set. Otherwise, it results in EWDT_OUT_b assertion.	0x0

15.4 Function description

This section provides the functional information of the EWDT module.

15.4.1 The EWDT_OUT signal

The EWDT_OUT is a digital output signal used to place an external circuit (application customization) in safe state.

The EWDT_OUT signal remains deasserted when the EWDT has been regularly refreshed by the software within the configurable refresh window, indicating that the application code runs as expected.

Any of the following behavior results in the EWDT_OUT signal assertion:

- The EWDT service occurs if the counter value is less than the EWDT_COMPL register value
- The EWDT counter value reaches the EWDT_COMPH register value, and no EWDT refresh has occurred
- The functionality of EWDT_IN pin is enabled by setting EWDT_CS[INPE] to 1, and EWDT_IN pin is asserted while EWDT service occurs

After any reset (by the external pull-down resistor on the EWDT_OUT pin), the EWDT_OUT is asserted after any reset by the external pull-down resistor on the EWDT_OUT pad. Enable EWDT by setting EWDT_CS[EWDTTE] bit to 1b to deassert EWDT_OUT.

15.4.2 The EWDT_IN signal

The EWDT_IN is a digital input signal for safety control from external safety circuits that allows an external circuit to control the assertion of the EWDT_OUT signal. The EWDT_IN signal is ignored if the functionality of EWDT is disabled or if the EWDT_CS[INPE] bit is cleared.

When EWDT is enabled by setting the EWDT_CS[EWDTTE] bit, and the functionality of EWDT_IN is enabled by setting the EWDT_CS[INPE] bit, the EWDT_IN signal must be deasserted before the CPU starts the EWDT service. This ensures that the EWDT_OUT stays in the deasserted state. Otherwise, the EWDT_OUT signal is asserted.

15.4.3 EWDT counter

EWDT has a 32-bit ripple counter with a configurable width range time-out period by selecting a different clock source.

The counter is reset to zero after the system resets, or EWDT service operation is performed, or counter value reaches the EWDT_COMPH register value. CPU reads the counter value.

15.4.4 EWDT compare registers

EWDT has two compare registers EWDT_COMPH and EWDT_COMPL. They are write-once after a system resets and become read-only after the first write until another system reset occurs.

The compare registers create a service window for the service operation.

You are not allowed to configure both EWDT_COMPH and EWDT_COMPL with the same value. In this condition, as soon as the counter reaches (EWDT_COMPL + 1), EWDT_OUT is asserted.

15.4.5 EWDT service operation

The service operation is used to clear the EWDT_CNT register. The CPU must write the EWDT_CNT register with a correct write sequence of unique word within the window time

frame as determined by the EWDT_COMPH and EWDT_COMPL registers for the correct EWDT service operation. The three possible conditions can occur:

Table 15-3 EWDT service conditions

Condition	Operation results
An EWDT service occurs when: $COMPL < Counter < COMPH$.	EWDT counter is reset to zero. The EWDT_OUT output remains in the deasserted state if the EWDT_IN input has been in a deasserted state.
An EWDT service occurs when: $Counter < COMPL$.	The service occurred before the windowed time frame. The counter is reset to zero, and the EWDT_OUT output is asserted regardless of the state of the EWDT_IN input.
No service occurred when the counter value equals to COMPH.	The EWDT counter is reset to zero and the EWDT_OUT output is asserted regardless of the state of the EWDT_IN input.

15.4.6 EWDT interrupts

An interrupt request (EWDT_CS[INTF]) is generated to indicate the assertion of the EWDT reset out signal, when EWDT_OUT is asserted. This interrupt is enabled by setting EWDT_CS[INTE] to 1b.

Write 1b to this bit (EWDT_CS[INTF]) to clear the interrupt request but does not affect EWDT_OUT. The EWDT_OUT signal can be deasserted only by forcing a system reset.

15.5 Application information

This section provides the application information of the EWDT module.

15.5.1 EWDT application sequence

Run the following steps to initialize the EWDT application:

1. Set the PORT pin multiplexer to the EWDT_OUT, EWDT_IN mode.
2. Write to the EWDT_COMPH register to set the maximum time for the CPU to refresh the EWDT counter.
3. Write to the EWDT_COMPL register to set the minimum time for the CPU to refresh the EWDT counter.
4. Write to the EWDT_CS register to define the operating configuration of EWDT, including EWDT behavior in debug mode and low-power modes, input port signal configuration, interrupt mask and EWDT counter enabling.
5. Periodically perform a refresh operation to reset the EWDT counter when its count value is between the EWDT_COMPL value and the EWDT_COMPH value. The EWDT_IN signal must be deasserted when the EWDT service starts.

Chapter 16

Clock Monitor Unit (CMU)

16.1 Introduction

CMU modules are required to measure monitored clock frequency or ensure clock signal integrity on the chip. In measure mode, CMU uses reference clock to calculate the frequency of monitored clock. The measurement gets started by user trigger and gets stopped at the end of measure. In monitor mode, CMU monitors if the frequency of a monitored clock is within a programmable frequency range specified by the user. If the frequency is outside the specified limits, it could lead to performance, protocol, and timing failures. CMU requires a clock that is used as a base reference for the frequency check operation. This reference clock must be within documented limits for correct CMU operation. If a reference clock is outside operating parameters, CMU behavior cannot be guaranteed.

16.1.1 Features

The CMU supports the following key features:

- If a monitored clock frequency is above high frequency threshold (FAHT), an event occurs.
- If a monitored clock frequency is below low frequency threshold (FBLT), an event occurs.
- Programmable duration of reference clock cycles.
- Time out functionality to generate an FBLT event when a monitored clock is lost.
- Masking of interrupt.
- Optional monitor mode or measure mode.

16.1.2 Block diagram

The following figure is a block diagram of the Clock Monitor Unit (CMU) module.

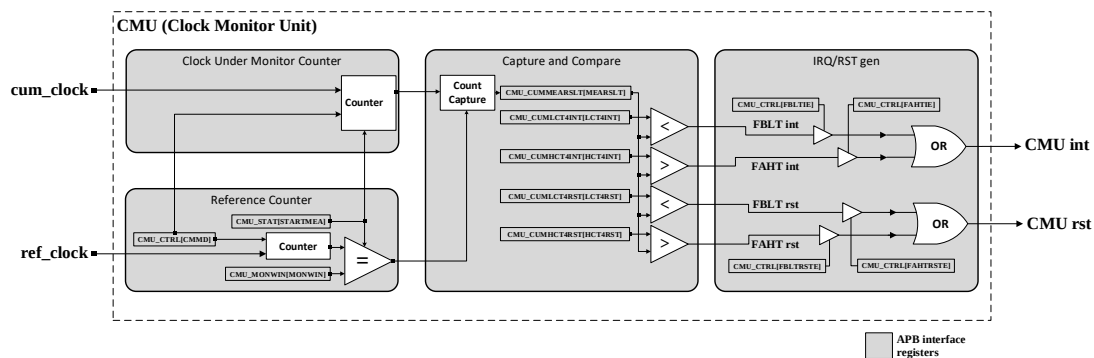


Figure 16-1 CMU block diagram

16.2 Memory map and register definition

This section provides the memory map and register definitions of the CMU module.

16.2.1 Memory map

CMU0 base address: 0x4003E000

CMU1 base address: 0x40042000

CMU2 base address: 0x40043000

Table 16-1 CMU memory map

Offset	Register	Width (In bits)	Reset Value
0x4	Clock Monitor Unit Parameter (CMU_PARAM)	32	0x00000001
0x8	Clock Monitor Unit Control (CMU_CTRL)	32	0x80000000
0xC	Clock Monitor Unit Status (CMU_STAT)	32	0x00000000
0x10	Clock Monitor Unit Monitor Window (CMU_MONWIN)	32	0x00000000
0x14	Clock Monitor Unit Clock under Monitor High Compare Threshold for Interrupt (CMU_CUMHCT4INT)	32	0x00000000
0x18	Clock Monitor Unit Clock under Monitor Low Compare Threshold for Interrupt (CMU_CUMLCT4INT)	32	0x00000000
0x1C	Clock Monitor Unit Clock under Monitor High Compare Threshold for Reset (CMU_CUMHCT4RST)	32	0x00000000
0x20	Clock Monitor Unit Clock under Monitor Low Compare Threshold for Reset (CMU_CUMLCT4RST)	32	0x00000000
0x24	Clock Monitor Unit Clock under Monitor Measurement Result (CMU_CUMMEARSLT)	32	0x00000000

16.2.2 Clock Monitor Unit Parameter (CMU_PARAM)

16.2.2.1 Offset

Register	Offset
CMU_PARAM	0x04

16.2.2.2 Fields

Field	Access	Function	Reset Value
0 RST_IMP	RO	0b: Reset generation is unimplemented. 1b: Reset generation is implemented.	0x1
31:1 RSVD_31_1	RO	Reserved	0x0

16.2.3 Clock Monitor Unit Control (CMU_CTRL)

16.2.3.1 Offset

Register	Offset
CMU_CTRL	0x08

16.2.3.2 Function

This register is used for CMU control.

16.2.3.3 Fields

Field	Access	Function	Reset Value
0 CME	RW	All configuration registers are write lock when this bit is set. 0b: Monitor is disabled. 1b: Monitor is enabled.	0x0
3:1 RSVD_3_1	RO	Reserved	0x0
4 CMMD	RW	0b: Monitor mode. 1b: Measure mode.	0x0
7:5 RSVD_7_5	RO	Reserved	0x0
8 FAHTRSTE	RW	0b: Reset is disabled. 1b: Reset is enabled.	0x0
9 FBLTRSTE	RW	0b: Reset is disabled. 1b: Reset is enabled.	0x0
15:10 RSVD_15_10	RO	Reserved	0x0
16 FAHTIE	RW	0b: Interrupt is disabled. 1b: Interrupt is enabled.	0x0
17 FBLTIE	RW	0b: Interrupt is disabled. 1b: Interrupt is enabled.	0x0
30:18 RSVD_30_18	RO	Reserved	0x0
31 LOCK	RW	Write 0x5B to CMU_CTRL[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. LOCK bit only affects write for CMU_CTRL[CME]. 0b: Register is writable. 1b: Register is read-only.	0x1

16.2.4 Clock Monitor Unit Status (CMU_STAT)

16.2.4.1 Offset

Register	Offset
CMU_STAT	0x0C

16.2.4.2 Function

This register is used to indicate status.

16.2.4.3 Fields

Field	Access	Function	Reset Value
1:0 MONST	RO	00b: Configuration is updatable. 01b: Configuration is locked. 10b: Monitor is running.	0x0
3:2 RSVD_3_2	RO	Reserved	0x0
4 STARTMEA	RW	0b: Measurement is completed. 1b: Start to measure and measurement is in	0x0

Field	Access	Function	Reset Value
		progress.	
15:5 RSVD_15_5	RO	Reserved	0x0
16 FAHTIF	W1C	Write 1 to clear. 0b: Interrupt does not happen. 1b: Interrupt happens.	0x0
17 FBLTIF	W1C	Write 1 to clear. 0b: Interrupt does not happen. 1b: Interrupt happens.	0x0
31:18 RSVD_31_18	RO	Reserved	0x0

16.2.5 Clock Monitor Unit Monitor Window (CMU_MONWIN)

16.2.5.1 Offset

Register	Offset
CMU_MONWIN	0x10

16.2.5.2 Function

This register is used for monitor window configuration.

16.2.5.3 Fields

Field	Access	Function	Reset Value
31:0 MONWIN	RW	CMU will capture the value of clock under monitor counter when reference counter reaches up this value.	0x0

16.2.6 Clock Monitor Unit Clock under Monitor High Compare Threshold for Interrupt (CMU_CUMHCT4INT)

16.2.6.1 Offset

Register	Offset
CMU_CUMHCT4INT	0x14

16.2.6.2 Function

This register is used for high comparison threshold configuration for interrupt.

16.2.6.3 Fields

Field	Access	Function	Reset Value
31:0 HCT4INT	RW	It is used for high comparison threshold configuration for interrupt.	0x0

16.2.7 Clock Monitor Unit Clock under Monitor Low Compare

Threshold for Interrupt (CMU_CUMLCT4INT)

16.2.7.1 Offset

Register	Offset
CMU_CUMLCT4INT	0x18

16.2.7.2 Function

This register is used for low comparison threshold configuration for interrupt.

16.2.7.3 Fields

Field	Access	Function	Reset Value
31:0 LCT4INT	RW	It is used for low comparison threshold configuration for interrupt.	0x0

16.2.8 Clock Monitor Unit Clock under Monitor High Compare Threshold for Reset (CMU_CUMHCT4RST)

16.2.8.1 Offset

Register	Offset
CMU_CUMHCT4RST	0x1C

16.2.8.2 Function

This register is used for high comparison threshold configuration.

16.2.8.3 Field

Field	Access	Description	Reset Value
31:0 HCT4RST	RW	It is used for high comparison threshold configuration for reset.	0x0

16.2.9 Clock Monitor Unit Clock under Monitor Low Compare Threshold for Reset (CMU_CUMLCT4RST)

16.2.9.1 Offset

Register	Offset
CMU_CUMLCT4RST	0x20

16.2.9.2 Function

This register is used for low comparison threshold configuration for reset.

16.2.9.3 Fields

Field	Access	Function	Reset Value
31:0 LCT4RST	RW	It is used for low comparison threshold configuration for reset.	0x0

16.2.10 Clock Monitor Unit Clock under Monitor Measurement Result (CMU_CUMMEARSLT)

16.2.10.1 Offset

Register	Offset
CMU_CUMMEARSLT	0x24

16.2.10.2 Function

This register is read-only.

16.2.10.3 Fields

Field	Access	Function	Reset Value
31:0 MEARSLT	RO	Get measurement result by reading this register when CMU_STAT[STARTMEA] returns to 0.	0x0

16.3 Function description

This section provides the functional information of the CMU module.

16.3.1 Clock monitor mode

Clock monitor starts when software writes CMU_CTRL[CME] = 1. The CMU requires a fixed number of reference clock cycles prior to the start of hardware operations. The status field CMU_STAT[MONST] = 2'b10 indicates monitor operation is started. Safety critical applications must poll CMU_STAT[MONST] to determine when a check operation starts. CMU performs continuous periodic clock checking of the monitored clock.

Each checking window contains the following operations:

Phase 1 - Reference clock counter runs for CMU_MONWIN[MONWIN] reference clock cycles. Monitored clock counter runs in parallel for the same time duration as reference clock counter.

Phase 2 - After phase 1, the module compares the final monitored clock count against the programmed thresholds.

If the monitored clock count is greater than the high threshold value in CMU_CUMHCT4INT[HCT4INT], an FAHT interrupt event is generated. If the monitored clock count is lower than the low threshold value in CMU_CUMLCT4INT[LCT4INT], an FBLT interrupt event is generated.

If the monitored clock count is greater than the high threshold value in CMU_CUMHCT4RST[HCT4RST], an FAHT reset event is generated. If the monitored clock count is lower than the low threshold value in CMU_CUMLCT4RST[LCT4RST], an FBLT reset event is generated. The reset event will be sent to SERU, then SERU decides to generate interrupt or reset. So when using CMU reset function, users should configure SERU first, including SERU_CFG_CHn[CH_EN] to enable channel of the specific CMU, SERU_CFG_CHn[CH_RESET_TYP] to select reset type, SERU_CFG_CHn[CH_RESET_DLY] to select delay before reset occurs,

SERU_CFG_CHn[CH_RESET_CFG] to enable reset. (Note: Reset function is unsupported by CMU1).

The error of both monitored clock and reference clock shall be considered while calculating low threshold and high threshold values. For the threshold calculation formula, please refer to the frequency calculation formula in next section.

$$TH = window\_value * \frac{f_{cum\_clk}}{f_{ref\_clk}}$$

For example:

$$\begin{aligned} f_{ref\_clk} &= 40M(39M \sim 41M) \\ f_{cum\_clk} &= 64M(63.6976M \sim 64.9974M) \end{aligned}$$

$$window\_value = 10000$$

$$TH_{high} = 10000 * \frac{64.9974}{39} = 16666$$

$$TH_{low} = 10000 * \frac{63.6976}{41} = 15366$$

The CMU measurement error can be negligible. It is recommended that users set the threshold range larger than what is calculated above.

16.3.2 Frequency measure mode

When the reset signal is released and the CMU_CTRL[CMMD] bit is set to 1, the CMU will work in measure mode. Frequency measurement starts when software writes CMU_STAT[STARTMEA] = 1 with configuration CMU_CTRL[CMMD] = 1 and CMU_CTRL[CME] = 1. User needs to poll CMU_STAT[STARTMEA], once the CMU_STAT[STARTMEA] returns to 0, the CUM(clock under monitor) measurement result will be stored in CMU_CUMMEARSLT register. The result of measuring this clock is:

$$f_{cum\_clk} = f_{ref\_clk} * \frac{measure\_value}{window\_value}$$

Use the measure mode, there are some errors in calculating the CUM clock. The magnitude of this error is determined by the frequency ratio of the reference clock to the CUM clock.

$$n = \frac{f_{ref\_clk}}{f_{cum\_clk}}$$

If the $n \leq 1$, the max_error_value is:

$$max_{error\_value} = 2$$

If the $n > 1$, the max_error_value is:

$$max_{error\_value} = 2n - 2$$

The measure error:

$$max_{error} = f_{ref\_clk} * \frac{max_{error\_value}}{window\_value}$$

For example:

$$\begin{aligned} f_{ref\_clk} &= 40M \\ f_{cum\_clk} &= 64M \\ window\_value &= 10000 \\ max_{error\_value} &= 2 \end{aligned}$$

$$\max_{error} = 40M * \frac{2}{10000} = 8k$$

So, in order to reduce the error, users can set a larger window value, especially when the frequency ratio of the reference clock to the CMU clock is large.

16.3.3 Monitored clock and reference clock

The monitored clock and the reference clock are fixed for each CMU as shown in the following Table 16-2. Note that the monitored clock and the reference clock must be enabled prior to using CMU.

Table 16-2 Monitored clock and reference clock of CMU

	Monitored clock	Reference clock
CMU0	Bus clock	FIRC64M clock
CMU1	FIRC64M clock	OSC40M clock
CMU2	LPO32K clock	OSC40M clock

Chapter 17

Error Injection and Report Unit (EIRU)

17.1 Introduction

The purpose of this chapter is to define an Error Injection and Report Unit (EIRU). Error Injection and Report Unit (EIRU) is designed for function safety purpose of Error Correction Code (ECC) self-diagnostic. It provides single-bit and multi-bit error injection on SRAM read data and ECC check bits by inverting the data buses. EIRU is implemented between SRAM array and SRAM controller to fulfil the ECC error injection. It also reports the ECC error information when ECC error occurs on SRAM.

17.1.1 Features

The EIRU supports the following key features:

- ECC error injection.
 - Supports error injection for two SRAM.
 - Supports error injection on SRAM read data and check bits.
 - Supports single-bit and multi-bit error injection.
 - The error injection bits on read data and check bits are configurable.
 - The error injection address of SRAM is configurable.
 - Provides error injection enable/disable control.
- ECC error reports.
 - Captures the ECC fault address and fault data when ECC error is detected.
 - Captures the ECC fault information, such as fault access size, fault access attributes, single-bit ECC error and multi-bit ECC error status.
 - Provides the single-bit and multi-bit ECC error report enable/disable control.
 - Provides write ECC error and read ECC error report enable/disable control.
- Provide global lock bit for write protection.

17.1.2 Block diagram

The following figure is a block diagram of the Error Injection and Report Unit (EIRU) module.

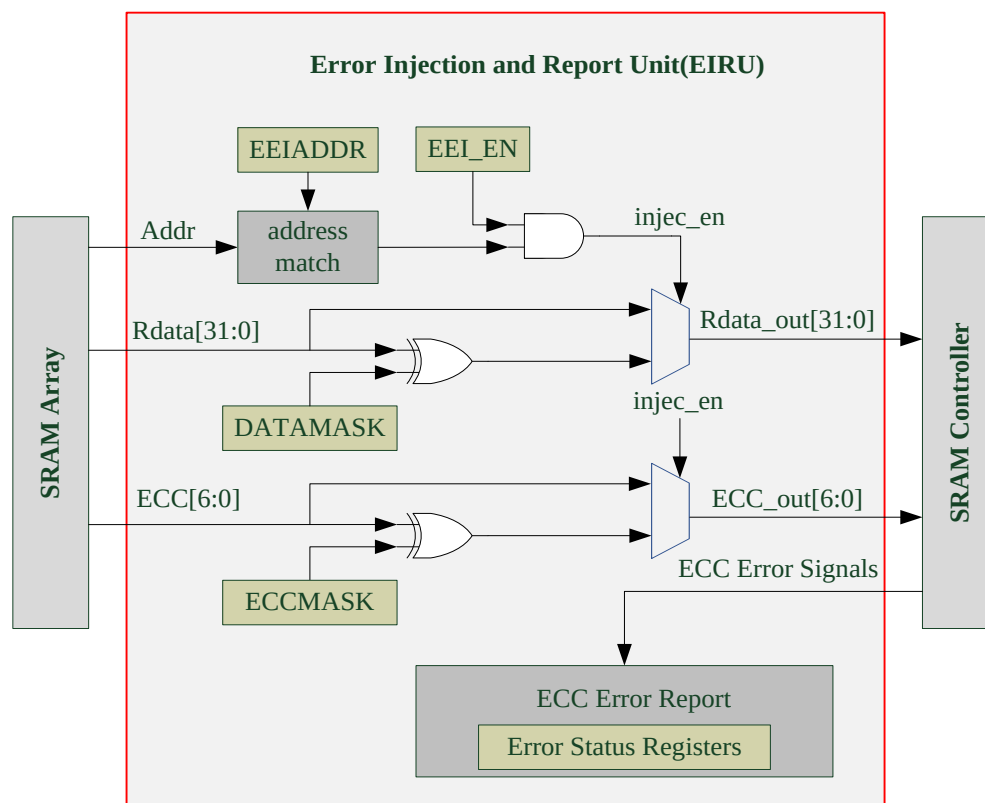


Figure 17-1 EIRU block diagram

17.2 Memory map and register definition

This section provides the memory map and register definitions of the EIRU module.

17.2.1 Memory map

EIRU base address: 0x4000B000

Table 17-1 EIRU memory map

Offset	Register	Width (In bits)	Reset value
0x0	EIRU Control Register (EIRU_CR)	32	0x80000000
0x4	EIRU ECC Error Injection Address (EIRU_EEIADDR)	32	0x00000000
0x10 + 0x20*n (n=0~1)	EIRU SRAM n ECC Error Status Register (EIRU_SRAMn_EESR)	32	0x00000000
0x14 + 0x20*n (n=0~1)	EIRU SRAM n ECC Fault Address Register (EIRU_SRAMn_FADDR)	32	0x00000000
0x18 + 0x20*n (n=0~1)	EIRU SRAM n ECC Fault Data Register (EIRU_SRAMn_FDATA)	32	0x00000000
0x1C + 0x20*n (n=0~1)	EIRU SRAM n ECC Check Bit Mask (EIRU_SRAMn_ECCMASK)	32	0x00000000
0x20 + 0x20*n (n=0~1)	EIRU SRAM n Data Bit Mask (EIRU_SRAMn_DATAMASK)	32	0x00000000

17.2.2 EIRU Control Register (EIRU_CR)

17.2.2.1 Offset

Register	Offset
EIRU_CR	0x00

17.2.2.2 Function

This register is used to set basic configuration.

17.2.2.3 Fields

Field	Access	Description	Reset value
0 SRAM0_ER_DIS	RW	This bit defines whether the ECC error will be reported when an ECC error occurs on SRAML read access. This bit can only be written when LOCK bit is unlocked. 0b: SRAML ECC read is enabled. 1b: SRAML ECC read is disabled.	0x0
1 SRAM0_EW_DIS	RW	This bit defines whether the ECC check bit will be calculated before writing data to SRAML. This bit can only be written when LOCK bit is unlocked. 0b: SRAML ECC write is enabled. 1b: SRAML ECC write is disabled.	0x0
2 SRAM0_ESB_DIS	RW	This bit defines whether ECC error will be reported when SRAML single-bit error occurred. This bit can only be written when LOCK bit is unlocked. 0b: SRAML single-error report is enabled. 1b: SRAML single-error report is disabled.	0x0
3 SRAM0_EMB_DIS	RW	This bit defines whether ECC error will be reported when SRAML multi-bit error occurred. This bit can only be written when LOCK bit is unlocked. 0b: SRAML multi-error report is enabled. 1b: SRAML multi-error report is disabled.	0x0
4 SRAM0_EEI_EN	RW	This bit controls the SRAML ECC error injection enable or disable. This bit can only be written when LOCK bit is unlocked. 0b: SRAML ECC error injection is disabled. 1b: SRAML ECC error injection is enabled.	0x0
7:5 RSVD	RO	Reserved	0x0
8 SRAM1_ER_DIS	RW	This bit defines whether the ECC error will be reported when an ECC error occurs on SRAMU read access. This bit can only be written when LOCK bit is unlocked. 0b: SRAMU ECC read is enabled. 1b: SRAMU ECC read is disabled.	0x0
9 SRAM1_EW_DIS	RW	This bit defines whether the ECC check bit will be calculated before writing data to SRAMU. This bit can only be written when LOCK bit is unlocked. 0b: SRAMU ECC write is enabled. 1b: SRAMU ECC write is disabled.	0x0

Field	Access	Description	Reset value
10 SRAM1_ESB_DIS	RW	This bit defines whether ECC error will be reported when SRAMU single-bit error occurred. This bit can only be written when LOCK bit is unlocked. 0b: SRAMU single-error report is enabled. 1b: SRAMU single-error report is disabled.	0x0
11 SRAM1_EMB_DIS	RW	This bit defines whether ECC error will be reported when SRAMU multi-bit error occurred. This bit can only be written when LOCK bit is unlocked. 0b: SRAMU multi-error report is enabled. 1b: SRAMU multi-error report is disabled.	0x0
12 SRAM1_EEI_EN	RW	This bit controls the SRAMU ECC error injection enable or disable. This bit can only be written when LOCK bit is unlocked. 0b: SRAMU ECC error injection is disabled. 1b: SRAMU ECC error injection is enabled.	0x0
30:13 RSVD	RO	Reserved	0x0
31 LOCK	RW	This lock bit defines whether EIRU_CR[12:0] field can be written. Only when this bit is unlocked can the EIRU_CR[12:0] field be written. Write 0x5C to field EIRU_CR[31:24] to unlock this bit. 0b: Unlocked. 1b: Locked.	0x1

17.2.3 EIRU ECC Error Injection Address (EIRU_EEIADDR)

17.2.3.1 Offset

Register	Offset
EIRU_EEIADDR	0x04

17.2.3.2 Function

This register configures the most significant bit [31:2] of ECC error injection address. When there is a read access made to SRAM, only when the read address is matched with this error injection address, ECC error can be injected on the data buses.

17.2.3.3 Fields

Field	Access	Description	Reset value
1:0 RSVD	RO	Reserved	0x0
31:2 EEIADDR	RW	This field defines the most significant bit [31:2] of ECC error injection address.	0x0

17.2.4 EIRU SRAM n ECC Error Status Register (EIRU_SRAMn_EESR)

17.2.4.1 Offset

Register	Offset
EIRU_SRAMn_EESR	0x10 + 0x20*n(n=0~1)

17.2.4.2 Function

This register indicates the error status on SRAM.

17.2.4.3 Fields

Field	Access	Description	Reset value
3:0 EFPROT	RO	Indicates protect status when error is detected. HPROT[0]: 0b: Opcode fetch, 1b: Data access HPROT[1]: 0b: User access, 1b: Privileged access Others: Reserved.	0x0
6:4 EFSIZE	RO	Indicates error caused by what size of access. 00b: 8 bits 01b: 16 bits 10b: 32 bits Others: Reserved	0x0
7 RSVD_7	RO	Reserved	0x0
15:8 EFMST	RO	Indicates error caused by which master. 00b: Master is core. 01b: Master is debug. 10b: Master is DMA. Others: Reserved.	0x0
28:16 RSVD_28_16	RO	Reserved	0x0
29 SBERR	W1C	0b: No error detected or multi-bit error detected. 1b: Single-bit error detected.	0x0
30 MBERR	W1C	0b: No error detected or single-bit error detected. 1b: Multi-bit error detected.	0x0
31 OVERRUN	W1C	If the first ECC error occurs and MBERR or SBERR bit is not cleared, the second ECC error will cause ECC error overrun. 0b: The overrun does not happen. 1b: The overrun happens.	0x0

17.2.5 EIRU SRAM n ECC Fault Address Register (EIRU_SRAMn_FADDR)

17.2.5.1 Offset

Register	Offset
EIRU_SRAMn_FADDR	0x14 + 0x20*n(n=0~1)

17.2.5.2 Function

This register is used to record SRAMn address where error is detected.

17.2.5.3 Fields

Field	Access	Description	Reset value
31:0 EFADDR	RO	Records the SRAMn read address where error is detected.	0x0

17.2.6 EIRU SRAM n ECC Fault Data Register (EIRU_SRAMn_FDATA)

17.2.6.1 Offset

Register	Offset
EIRU_SRAMn_FDATA	0x18 + 0x20*n(n=0~1)

17.2.6.2 Function

This register is used to record SRAMn data when error is detected.

17.2.6.3 Fields

Field	Access	Description	Reset value
31:0 EFDATA	RO	Records the SRAMn read data when error is detected.	0x0

17.2.7 EIRU SRAM n ECC Check Bit MASK (EIRU_SRAMn_ECCMASK)

17.2.7.1 Offset

Register	Offset
EIRU_SRAMn_ECCMASK	0x1C + 0x20*n(n=0~1)

17.2.7.2 Function

This register configures the error injection bit on ECC check bit. The inversion of the ECC check bit occurs only during a read operation and does not alter the actually stored value. Value 1 represents this related ECC check bit will invert, value 0 represents this related ECC check bit will not change.

17.2.7.3 Fields

Field	Access	Description	Reset value
6:0 ECCMASK	RW	This field defines the inverted bit of ECC check bit.	0x0
31:7 RSVD_31_7	RO	Reserved	0x0

17.2.8 EIRU SRAM n DATA Bit MASK (EIRU_SRAMn_DATAMASK)

17.2.8.1 Offset

Register	Offset
EIRU_SRAMn_DATAMASK	0x20 + 0x20*n(n=0~1)

17.2.8.2 Function

This register configures the error injection bit on SRAM read data bit. The inversion of the ECC check bit occurs only during a read operation and does not alter the actually stored value. Value 1 represents this related read data bit will invert, value 0 represents this related read data bit will not change.

17.2.8.3 Fields

Field	Access	Description	Reset value
31:0 DATAMASK	RW	This field defines the inverted bit of SRAM read data.	0x0

17.3 Function description

This section provides the functional information of the EIRU module.

17.3.1 ECC error injection

On a SRAM read access, EIRU can inject error on the read data and ECC check bits by inverting the single-bit or multi-bit output data. The EIRU_SRAMn_ECCMASK[ECCMASK] fields define which ECC check bits are inverted, and EIRU_SRAMn_DATAMASK[DATAMASK] will define which read data bits are inverted.

The error can only be injected on SRAM dedicated address. This address is configurable. The address is hit and error can be injected on the data bus of SRAM only when the most significant bit [31:2] of SRAM read address matches with the EIRU_EEIADDR[EEIADDR]. If the address is not hit, the error injection is invalid.

EIRU has global error injection enable control bit. The error injection is valid only when EIRU_CR[SRAMn_EEI_EN] is enabled and address is hit.

EIRU supports single-bit error injection and multi-bit error injection scenarios:

- If the total inverted bit defined by ECCMASK and DATAMASK is 1, it is single-bit error injection and generates single-bit error.
- If the total inverted bit defined by ECCMASK and DATAMASK is 2, it is multi-bit error injection and generates multi-bit error.

NOTE

As the ECC only supports single-bit error correction and multi-bit error report, if the total inverted bit defined by ECCMASK and DATAMASK is more than 2, it may result in unpredictable behavior.

17.3.2 ECC error report

When an ECC single-bit error or multi-bit error occurs, the EIRU will capture the ECC fault address and fault data, and record the error information in ECC error status register EIRU_SRAMn_EESR.

The EIRU provides report disable control on single-bit error and multi-bit error. Only when EIRU_CR[SRAMn_ESB_DIS] and EIRU_CR[SRAMn_EMB_DIS] are 0 will the ECC error be reported by EIRU when the related ECC error occurs.

The EIRU also provides ECC write disable and read disable ability.

- ECC write disable: Before writing data to SRAM, the SRAM controller will first calculate the ECC check bit based on the write data, and then write the calculated ECC check bit into SRAM array. If the ECC write is disabled, the calculated ECC check bit will be 0.
- ECC read disable: When reading data from SRAM, the SRAM controller will compare the read data and check bit with expected value. If they are not matched, EIRU will report error. If the ECC read is disabled, EIRU will not report error when an ECC error occurred.

17.3.3 Write protection

EIRU implements global lock bit EIRU_CR[LOCK] for write protection on some critical register fields. When the LOCK bit is 1, EIRU_CR[12:0] fields are write protected. The default value of LOCK bit is 1 after chip reset. It can only be unlocked by writing 0x5C to EIRU_CR[31:24] field. Writing a 1 to LOCK bit will lock it again.

Chapter 18

System Error Report Unit (SERU)

18.1 Introduction

System Error Report Unit (SERU) is used to consolidate all the MCU internal errors from other hardware modules and map these errors to one or several configurable error reactions. In case of a detected error in MCU, depending on the configuration, SERU sends interrupt to system interrupt controller or sends NMI interrupt to core directly, or sends normal reset request to system reset controller. Meanwhile, SERU is able to send 2 error outputs (EOUTs) to external monitor, which can shut down the external actuators or communication interfaces to make sure the whole system is in safe state. If ALARM/NMI interrupt is generated, SW shall try to locally recover the error in time. If the error fails to be recovered, SW shall trigger further reaction via dedicated SW error indication register to make sure system enters safe state (e.g. assert normal reset and EOUTs). If SERU module is to be used, the FIRC clock must be kept valid.

18.1.1 Features

The SERU module provides the following features:

- Up to 15 error channels to SERU checker, including 3 internal software channels
- Configure mode to protect SERU registers from being tampered
- IO force on mode for EOUT to monopolize corresponding pins
- False Fault Injection function on all error channels for SERU self-test
- 3 internal software channels to generate error response for user software
- Register parity interrupt for key registers
- Configurable normal reset or cold reset request for every channel response
- Configurable reset delay on error event of every channel
- Configurable normal interrupt or NMI interrupt for every channel response
- Configurable EOUT polarity for every channel response
- Implicit lock-step check for every channel response
- Error record function for every channel regardless of whether it is enabled or not
- Only 32-bit access is supported for SERU registers

18.1.1.1 Error channel mapping information

SERU error channel	Source module	Error description
0	SW	SWERR0
1	SW	SWERR1
2	SW	SWERR2
3	PMU	LVW flag
4	PMU/SRMC	Device enters STOP mode or WAIT mode
5	EWDT	EWDT interrupt flag
6	CPU	Lockup flag
8	SCC	OSC40M loss of clock
9	CMU0	Bus clock outside the threshold for reset
10	CMU2	LPO outside the threshold

SERU error channel	Source module	Error description
		for reset
11	Flash	Flash Multi-bit ECC error
12	Flash	Flash un-secure
13	SRAM	SRAMU Multi-bit ECC error
14	SRAM	SRAML Multi-bit ECC error
16	CPU	Debug & halt

NOTE

- When a low-power mode is detected using channel 4, interrupts should not be enabled if the user expects a normal reset or cold reset. Similarly, if the user needs to generate an interrupt request, the user also needs to disable reset.
- When OSC40M loss of clock is detected using channel 8, cold reset must be used if the user expects a reset.

18.1.2 Modes of operation

18.1.2.1 WAIT mode

When the MCU enters WAIT mode, the SERU module is maintained, and all SERU functions work as normal.

18.1.2.2 STOP mode

In STOP mode, if the FIRC64M clock is enabled, the SERU module is maintained, and SERU reset and interrupt can wake up STOP mode of MCU.

18.1.2.3 STANDBY mode

In STANDBY mode, any functions of SERU are not supported.

18.1.3 Block diagram

The block diagram of the SERU module is shown in the following figure.

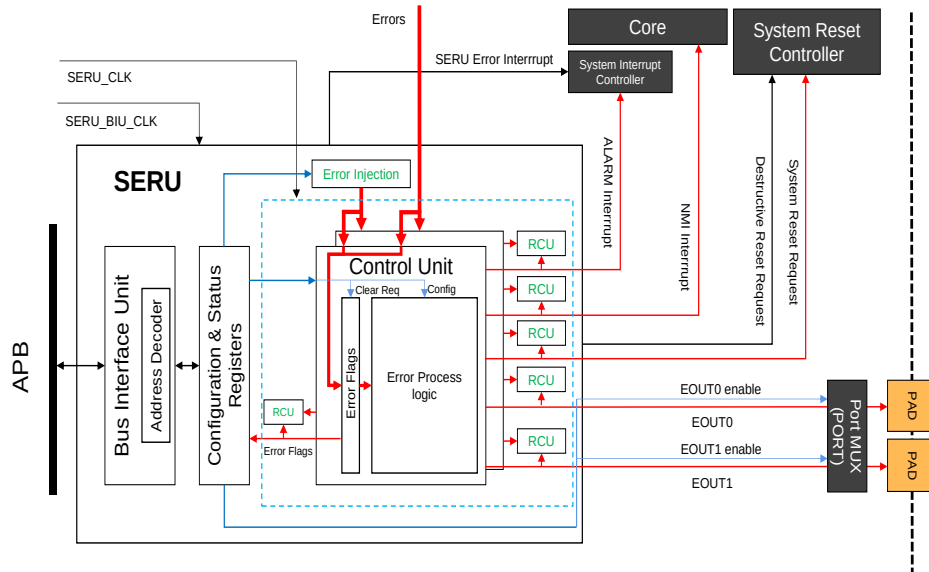


Figure 18-1 SERU block diagram

18.2 Memory map and register definition

This section provides the memory map and register definitions of the SERU module.

18.2.1 Memory map

SERU base address: 0x4007F000

Table 18-1 SERU memory map

Offset	Register	Width (In bits)	Reset Value
0x4	SERU Parameter Register (SERU_PARAM)	32	0x00000000
0x8	SERU Configure and Setting Register (SERU_CSR)	32	0x00000000
0xC	SERU INT Flag Register (SERU_INTF)	32	0x00000000
0x10	SERU Configure Key Register (SERU_CFG_KEY)	32	0x00000000
0x14	SERU Channel Fake Fault Injection Register (SERU_CH_FFI)	32	0x00000000
0x18	SERU Software Error Generator Register (SERU_SWERR_GNT)	32	0x00000000
0x1C	SERU Channel Flag Register (SERU_CH_FLAG)	32	0x00000000
0x20 + n*0x4 (n=0 ~ 6, 8 ~ 14, 16)	SERU Channel Configure Register (SERU_CFG_CHn)	32	0x00000000

18.2.2 SERU Parameter Register (SERU_PARAM)

18.2.2.1 Offset

Register	Offset
SERU_PARAM	0x4

18.2.2.2 Function

This is IP parameter register and it is read-only.

18.2.2.3 Fields

Field	Access	Description	Reset Value
31:0 PARAM	RO	This is IP parameter register.	0x0

18.2.3 SERU Configure and Setting Register (SERU_CSR)

18.2.3.1 Offset

Register	Offset
SERU_CSR	0x8

18.2.3.2 Fields

Field	Access	Description	Reset Value
0 CRPAR_INT_EN	RW	SERU internal error flag will be set if the mismatch of configuration register parity is detected. 0b: SERU internal error interrupt is disabled. 1b: SERU internal error interrupt is enabled.	0x0
3:1 RSVD_3_1	RW	Reserved	0x0
4 EOUT_PS	RW	Default value selection for EOUT (in the case of no fault generated). 0b: EOUT1 is low, and EOUT0 is high as default. 1b: EOUT1 is high, and EOUT0 is low as default.	0x0
5 EOUT0_IO_EN	RW	EOUT0 bypasses PORT control or not. 0b: EOUT0 is controlled by PORT ALT selection to output. 1b: EOUT0 bypasses PORT control to output external pin directly.	0x0
6 EOUT1_IO_EN	RW	EOUT1 bypasses PORT control or not. 0b: EOUT1 is controlled by PORT ALT selection to output. 1b: EOUT1 bypasses PORT control to output external pin directly.	0x0
7 FFI_EN	RW	Fake fault injection enable bit. 0b: Disable Fake fault injection. 1b: Enable Fake fault injection.	0x0
8 CFG_MODE	RO	0b: SERU is in normal mode (SERU_CSR[7:0] and SERU_CFG_CHx[7:0] are not writable.)	0x0

Field	Access	Description	Reset Value
		1b: SERU is in config mode (SERU_CSR[7:0] and SERU_CFG_CHx[7:0] are writable.)	
31:9 RSVD_31_9	RO	Reserved	0x0

18.2.4 SERU INT Flag Register (SERU_INTF)

18.2.4.1 Offset

Register	Offset
SERU_INTF	0xC

18.2.4.2 Fields

Field	Access	Description	Reset Value
0 S_CRPAR_ERR	RW1C	0b: No configuration register parity mismatch event occurs 1b: Configuration register parity mismatch event occurs	0x0
31:1 RSVD_31_1	RW	Reserved	0x0

18.2.5 SERU Configure Key Register (SERU_CFG_KEY)

18.2.5.1 Offset

Register	Offset
SERU_CFG_KEY	0x10

18.2.5.2 Function

Specific key to move SERU to configure mode or return to normal mode.

18.2.5.3 Fields

Field	Access	Description	Reset Value
31:0 KEY	RW	Return 0 when register is read. Writing 0x7f1a2dfe is to set SERU_CSR[CFG_MODE] bit. Writing 0x6cc92dd9 is to clear SERU_CSR[CFG_MODE] bit.	0x0

18.2.6 SERU Channel Fake Fault Injection Register (SERU_CH_FFI)

18.2.6.1 Offset

Register	Offset
SERU_CH_FFI	0x14

18.2.6.2 Fields

Field	Access	Description	Reset Value
6:0 CH_FFI_6_0	RW	0b: No action 1b: Fake fault injection at corresponding fault channel	0x0
7 RSVD	RO	Reserved	0x0
14:8 CH_FFI_14_8	RW	0b: No action 1b: Fake fault injection at corresponding fault channel	0x0
15 RSVD	RO	Reserved	0x0
16 CH_FFI_16	RW	0b: No action 1b: Fake fault injection at corresponding fault channel	0x0
31:17 RSVD	RO	Reserved	0x0

18.2.7 SERU Software Error Generator Register (SERU_SWERR_GNT)

18.2.7.1 Offset

Register	Offset
SERU_SWERR_GNT	0x18

18.2.7.2 Fields

Field	Access	Description	Reset Value
0 SWERR_CH0	RW	0b: No action or software error cancelled. 1b: Generate software error.	0x0
1 SWERR_CH1	RW	0b: No action or software error cancelled. 1b: Generate software error.	0x0
2 SWERR_CH2	RW	0b: No action or software error cancelled. 1b: Generate software error.	0x0
31:3 RSVD_31_3	RW	Reserved	0x0

18.2.8 SERU Channel Flag Register (SERU_CH_FLAG)

18.2.8.1 Offset

Register	Offset
SERU_CH_FLAG	0x1C

18.2.8.2 Fields

Field	Access	Description	Reset Value
6:0 CH_FLAG_6_0	R/W1C	0b: Error on this channel is not detected. 1b: Error on this channel is detected. Return to 0 if writing 1 to clear this bit and the	0x0

Field	Access	Description	Reset Value
		corresponding error is not present.	
7 RSVD	RO	Reserved	0x0
14:8 CH_FLAG_14_8	RW1C	0b: Error on this channel is not detected. 1b: Error on this channel is detected. Return to 0 if writing 1 to clear this bit and the corresponding error is not present.	0x0
15 RSVD	RO	Reserved	0x0
16 CH_FLAG_16	RW1C	0b: Error on this channel is not detected. 1b: Error on this channel is detected. Return to 0 if writing 1 to clear this bit and the corresponding error is not present.	0x0
31:17 RSVD	RO	Reserved	0x0

18.2.9 SERU Channel Configure Register (SERU_CFG_CHn)

18.2.9.1 Offset

Register	Offset
SERU_CFG_CHn	0x20 + n*0x4 (n=0 ~ 6, 8 ~ 14, 16)

18.2.9.2 Fields

Field	Access	Description	Reset Value
0 CH_EN	RW	0b: Channel is disabled, no respond for error generation. 1b: Channel is enabled.	0x0
1 CH_INT_CFG	RW	0b: Error input on this channel does not generate interrupt. 1b: Error input on this channel generates interrupt.	0x0
2 CH_INT_TYP	RW	0b: Normal interrupt is selected. 1b: NMI is selected.	0x0
3 CH_EOUT_CFG	RW	0b: Error input on this channel does not generate EOUT fault state. 1b: Error input on this channel generates EOUT fault state.	0x0
4 CH_RESET_CFG	RW	0b: Error input on this channel does not generate reset. 1b: Error input on this channel generates reset.	0x0
5 CH_RESET_TYP	RW	0b: Normal reset is selected. 1b: Cold reset is selected.	0x0
7:6 CH_RESET_DLY	RW	00b: No delay for reset generation. 01b: 2048 seru_clk cycle for reset delay (32us). 10b: 4096 seru_clk cycle for reset delay (64us). 11b: 8192 seru_clk cycle for reset delay (128us).	0x0
8 CH_FLAG	RW1C	This bit is same with the corresponding bit in SERU_CH_FLAG.	0x0

Field	Access	Description	Reset Value
		0b: Error on this channel is not detected. 1b: Error on this channel is detected. Return to 0 if writing 1 to clear this bit and the corresponding error is not present.	
31:9 RSVD_31_9	RW	Reserved	0x0

18.3 Functional description

The following sections describe functional details of the SERU module.

18.3.1 Interrupts

The SERU module can generate interrupt requests to the MCU upon the occurrence of various SERU module events. These interrupt events and their associated status and control bits are shown in the following table.

Table 18-2 SERU Module Interrupt Sources

SERU Event	Readable Status Bit	Interrupt Enable Bit
SERU parity error	SERU_INTF[S_CRPAR_ERR]	SERU_CSR[CRPAR_INT_EN]
SERU NMI interrupt response	SERU_CFG_CHn[CH_FLAG]	SERU_CFG_CHn[CH_INT_CFG] SERU_CFG_CHn[CH_INT_TYP]
SERU normal interrupt response	SERU_CFG_CHn[CH_FLAG]	SERU_CFG_CHn[CH_INT_CFG]

The SERU module can generate bus error (hard fault) interrupt for illegal address access as well.

18.3.2 Reset requests

The SERU module can generate two reset requests: normal reset request and cold reset request. Reset control module of MCU will generate normal reset based on SERU normal reset request while all SERU registers will maintain their states on normal reset process. Reset control module of MCU will generate cold reset for SERU cold reset request, and SERU and MCU system will be reset on cold reset process.

SERU normal reset request is generated when SERU channel receives error input with below configuration on corresponding channel:

- SERU channel is enabled: SERU_CFG_CHn[CH_EN] = 1
- Reset request is enabled to generate: SERU_CFG_CHn[CH_RESET_CFG] = 1
- Reset type is set to 0: SERU_CFG_CHn[CH_RESET_TYP] = 0
- Reset delay is optional to configure: SERU_CFG_CHn[CH_RESET_DLY] = 0, 1, 2, or 3 to configure reset delay as no delay, 32us, 64us, and 128us

SERU cold reset request is generated when SERU channel receives error input with below configuration on corresponding channel:

- SERU channel is enabled: SERU_CFG_CHn[CH_EN] = 1
- Reset request is enabled to generate: SERU_CFG_CHn[CH_RESET_CFG] = 1
- Reset type is set to 1: SERU_CFG_CHn[CH_RESET_TYP] = 1
- Reset delay is optional to configure: SERU_CFG_CHn[CH_RESET_DLY] = 0, 1, 2, or 3 to configure reset delay as no delay, 32us, 64us, and 128us

18.3.3 EOUT outputs

The SERU module can provide 2-bit EOUT output function for error event on any channel to alarm external monitor of the chip. The value of EOUT safe state or error state is configurable with SERU_CSR[EOUT_PS] setting. And the type of EOUT output is configurable with SERU_CSR[EOUT1_IO_EN] and SERU_CSR[EOUT0_IO_EN]. When the 2-bit region is set to high, EOUT will bypass PORT control to output external pin directly, otherwise, EOUT is controlled by PORT ALT selection to output as normal.

The EOUT output of SERU is generated when SERU channel receives error input with below configuration on corresponding channel:

- EOUT force-on output and parity are optional to configure: SERU_CSR[EOUT_PS] = 0 or 1, SERU_CSR[EOUT1_IO_EN] = 0 or 1 and SERU_CSR[EOUT0_IO_EN] = 0 or 1.
- EOUT function is enabled on SERU channel: SERU_CFG_CHn[CH_EOUT_CFG] = 1.

18.3.4 SERU configure mode

The SERU module provides configure mode to protect SERU_CSR and SERU_CFG_CHn registers. After POR reset, the configure mode of SERU is disabled. The below sequence is recommended in the user's application case:

- Write 32-bit unlock key 0x7f1a2dfe to register SERU_CFG_KEY to set SERU_CSR[CFG_MODE] and SERU enters configure mode
- Configure SERU_CSR and SERU_CFG_CHn registers
- Write 32-bit lock key 0x6cc92dd9 to register SERU_CFG_KEY to clear SERU_CSR[CFG_MODE] and SERU exits configure mode

18.3.5 SERU software error channels

The SERU module provides 3 software error channels, which are assigned to SERU channel 0-2, to test SERU circuit or be intentional to generate channel error for some special application cases. The below sequence is recommended to use software error:

- Configure SERU_CSR and SERU_CFG_CH0-2 to enable interrupt, reset or EOUT response function
- Write 1-7 to SERU_SWERR_GNT[2:0] to generate error input on SERU channel 0-2
- Check error response and write 0 to SERU_SWERR_GNT[2:0] to disable error generation

Additionally, on any reset process, the register SERU_SWERR_GNT[2:0] will be cleared automatically, and the error input will be disabled. On cold reset process, all of SERU registers are reset, including error flag. And on normal reset process, the error flag for software error may be recorded or not if reset delay is not enabled.

18.3.6 SERU false fault injection

The SERU module provides up to 32-bit SERU false fault injection on error channels to test SERU circuit. The below sequence is recommended to use software error:

- Enable false fault injection function: SERU_CSR[FFI_EN] = 1
- Configure SERU_CSR and SERU_CFG_CHn to enable interrupt, reset or EOUT response function
- Write 1 to SERU_CH_FFI[31:0] corresponding bits to generate error input on SERU channels
- Check error response and write 0 to SERU_CH_FFI[31:0] to disable false fault generation

Additionally, on any reset process, the register SERU_CH_FFI[31:0] will be cleared automatically, and the error input will be disabled. On cold reset process, all of SERU registers are reset, including error flag. And on normal reset process, the error flag for false

fault injection may be recorded or not if reset delay is not enabled.

18.3.7 SERU lock-step function

The SERU module provides lock-step function on every channel, which is a safety function to guarantee that the check and response logic of SERU channels works well on different conditions. Any lock-step error will generate cold reset request immediately.

18.3.8 SERU register parity check

The SERU module provides register parity check function for registers SERU_CSR and SERU_CFG_CHn, which is a safety function to guarantee that the key configuration of SERU registers SERU_CSR and SERU_CFG_CHn keeps right. Any register parity error will generate cold reset request immediately.

18.4 Initialization

SERU module is only reset by POR reset and cold reset. The initializing sequence is as below:

1. Clear SERU error flag if any error flag is present: writing 1 to SERU_CH_FLAG[31:0] or SERU_CFG_CHn[CH_FLAG].
2. Check SERU check and response function by software error channels or false fault injection function.
3. Configure SERU_CSR and SERU_CFG_CHn to enable interrupt, reset or EOUT response function to monitor MCU system.
4. Use software to handle any error response.

Chapter 19

True Random Number Generator (TRNG)

19.1 Introduction

The TRNG is a true random number generator that continuously provides 16-bit entropy samples, based on two analog noise sources. It can be used by the application as a live entropy source to build a Counter Deterministic Random Bit Generator (CTR-DRBG) to produce random bits as required.

19.1.1 Features

The TRNG supports the following key features:

- The TRNG delivers 16-bit true random numbers as a seed, produced by an Entropy source.
- The CTR-DRBG makes use of these seeds to generate random bits in normal mode or stream mode.
- Optional test mode to assess the randomness of the seeds.
- Configurable reseeds cycle and health tests cycle for the CTR-DRBG.
- Configurable sequence size of the test mode.

19.2 Memory map and register definition

This section provides the memory map and register definitions of TRNG module.

19.2.1 Memory map

TRNG base address: 0x40034000

Table 19-1 TRNG memory map

Offset	Register	Width (In bits)	Reset Value
0x0008	TRNG Status Register (TRNG_STATUS)	32	0x00000009
0x000C	TRNG Configuration Register (TRNG_CONFIG)	32	0x00000000
0x0010	TRNG CTR_DRBG Control Register (TRNG_CTR_DRBG_CONTROL)	32	0x00000000
0x0014	TRNG CTR_DRBG Request Size Register (TRNG_CTR_DRBG_REQUEST_SIZE)	32	0x00000000
0x0018	TRNG CTR_DRBG Configuration Register 1 (TRNG_CTR_DRBG_CONFIG_1)	32	0x00000000
0x001C	TRNG CTR_DRBG Configuration Register 2 (TRNG_CTR_DRBG_CONFIG_2)	32	0x00000000
0x0020	TRNG Alarm Register (TRNG_ALARM)	32	0x00000000
0x0024	TRNG CTR_DRBG Alarm Register (TRNG_CTR_DRBG_ALARM)	32	0x00000008
0x0028	TRNG Raw Random Control Register (TRNG_RAW_RANDOM_CONTROL)	32	0x00000000
0x002C	TRNG Raw Random Data Register (TRNG_RAW_RANDOM_DATA)	32	0x00000000

Offset	Register	Width (In bits)	Reset Value
0x0030	TRNG CTR_DRBG Random Data Register (TRNG_CTR_DRBG_RANDOM_DATA)	32	0x00000000
0x0034	TRNG Additional Configuration Register 0 (TRNG_ADDITIONAL_CONFIG_0)	32	0x00000000
0x0038	TRNG Additional Configuration Register 1 (TRNG_ADDITIONAL_CONFIG_1)	32	0x00000000
0x003C	TRNG Additional Configuration Register 2 (TRNG_ADDITIONAL_CONFIG_2)	32	0x00000000
0x0040	TRNG Additional Measurement Control Register (TRNG_ADDITIONAL_MEAS_CONTROL)	32	0x00000000
0x0044	TRNG Additional Measurement Configuration Register (TRNG_ADDITIONAL_MEAS_CONFIG)	32	0x00000000
0x0048	TRNG Additional Measurement Status Register (TRNG_ADDITIONAL_MEAS_STATUS)	32	0x00000001
0x004C	TRNG Additional Measurement Data Register 0 (TRNG_ADDITIONAL_MEAS_DATA_0)	32	0x00000000
0x0050	TRNG Additional Measurement Data Register 1 (TRNG_ADDITIONAL_MEAS_DATA_1)	32	0x00000000
0x0054	TRNG Interrupt Request Status Register (TRNG_IRQ_STATUS)	32	0x00000000
0x0058	TRNG Interrupt Request Clear Register (TRNG_IRQ_CLEAR)	32	0x00000000
0x005C	TRNG Interrupt Request Configuration Register (TRNG_IRQ_CONFIG)	32	0x00000000
0x0060	TRNG Error Configuration Register (TRNG_ERROR_CONFIG)	32	0x00000000
0x0064	TRNG Error Control Register (TRNG_ERROR_CONTROL)	32	0x00000000
0x0068	TRNG Error Status Register (TRNG_ERROR_STATUS)	32	0x00000005
0x006C	TRNG Monitoring Feature Configuration Register (TRNG_STATS_CONFIG)	32	0x00000000
0x0070	TRNG Monitoring Feature Control Register (TRNG_STATS_CONTROL)	32	0x00000000
0x0074	TRNG Monitoring Feature Status Register (TRNG_STATS_DATA)	32	0x00000000

19.2.2 TRNG Status Register (TRNG_STATUS)

19.2.2.1 Offset

Register	Offset
TRNG_STATUS	0x0008

19.2.2.2 Function

This register gives TRNG and CTR-DRBG status bits.

19.2.2.3 Fields

Field	Access	Description	Reset Value
0 RD	RO	CTR-DRBG ready. 0b: The TRNG IP is not ready to accept a new operation. 1b: The TRNG IP is ready to accept a new operation.	0x1
1 IN	RO	CTR-DRBG instantiated bit. 0b: CTR-DRBG is not instantiated. 1b: CTR-DRBG is instantiated.	0x0
2 TA	RO	TRNG alarms valid bit. 0b: Alarms are not available. 1b: Alarms are available on register TRNG_ALARM.	0x0
3 CA	RO	CTR-DRBG alarms valid bit. 0b: Alarms are not available. 1b: Alarms are available on register CTR_DRBG_ALARM.	0x1
31:4 RSVD	RO	Reserved	0x0

19.2.3 TRNG Configuration Register (TRNG_CONFIG)

19.2.3.1 Offset

Register	Offset
TRNG_CONFIG	0x000C

19.2.3.2 Function

This is the TRNG configuration register.

19.2.3.3 Fields

Field	Access	Description	Reset Value
0 EN	RW	CTR-DRBG enable bit. 0b: CTR-DRBG is disabled. 1b: CTR-DRBG is enabled, true random data will be generated.	0x0
1 SM	RW	Stream mode enable bit. 0b: CTR-DRBG stream mode is disabled. 1b: CTR-DRBG stream mode is enabled.	0x0
3:2 RSVD	RO	Reserved	0x0
13:4 DEC	RW	Decimation parameter. 'd0 and 'd1 mean there is no decimation. 'dx means 1 bit from the entropy source is picked up every x cycles. Recommended value for this input is 'd20.	0x0
18:14	RW	Latency parameter.	0x0

Field	Access	Description	Reset Value
LP		'd0 means the tested sequence size is 20 000 (i.e. maximum size). 'dx means the tested sequence size is x*1000. Recommended value for this input is 'd20.	
31:19 RSVD	RO	Reserved	0x0

19.2.4 TRNG CTR-DRBG Control Register (TRNG_CTR_DRBG_CONTROL)

19.2.4.1 Offset

Register	Offset
TRNG_CTR_DRBG_CONTROL	0x0010

19.2.4.2 Function

This is the CTR-DRBG control register. Reading this register will cause bus error.

19.2.4.3 Fields

Field	Access	Description	Reset Value
0 GE	WO	CTR_DRBG generate bit. 0b: No event. 1b: Launches a 'generate' operation (it is taken into account only when the DRBG is ready).	0x0
1 IN	WO	CTR_DRBG instantiate bit. 0b: No event. 1b: Launches an 'instantiate' operation (it is taken into account only when the DRBG is ready).	0x0
2 HT	WO	CTR_DRBG health test bit. 0b: No event. 1b: Launches a health test (it is taken into account only when the DRBG is ready).	0x0
3 UN	WO	CTR_DRBG uninstantiate bit. 0b: No event. 1b: Launches an 'uninstantiate' operation (it is taken into account only when the DRBG is ready).	0x0
4 RS	WO	CTR_DRBG reseed bit. 0b: No event. 1b: Launches a 'reseed' operation (it is taken into account only when the DRBG is ready).	0x0
5 AK	WO	CTR_DRBG alarm acknowledgement bit. 0b: No event. 1b: Acknowledge CTR_DRBG alarm.	0x0
31:6 RSVD	RO	Reserved	0x0

19.2.5 TRNG CTR-DRBG Request Size Register (TRNG_CTR_DRBG_REQUEST_SIZE)

19.2.5.1 Offset

Register	Offset
TRNG_CTR_DRBG_REQUEST_SIZE	0x0014

19.2.5.2 Function

This is the CTR_DRBG request size register (in 128-bit words).

19.2.5.3 Fields

Field	Access	Description	Reset Value
11:0 RS	RW	Number of 128-bit words that are requested. 'd0 means the number of requested 128-bit words is 1. 'dn means the number of requested 128-bit words is n+1.	0x0
31:12 RP	RW	Reserved	0x0

19.2.6 TRNG CTR-DRBG Configuration Register 1 (TRNG_CTR_DRBG_CONFIG_1)

19.2.6.1 Offset

Register	Offset
TRNG_CTR_DRBG_CONFIG_1	0x0018

19.2.6.2 Function

This is the CTR_DRBG configuration register 1.

19.2.6.3 Fields

Field	Access	Description	Reset Value
31:0 RP	RW	Number of generate requests between two reseeds. 'd0 means the number of generate requests between two reseeds is 'd1. 'dn means the number of generate requests between two reseeds is n+1.	0x0

19.2.7 TRNG CTR-DRBG Configuration Register 2 (TRNG_CTR_DRBG_CONFIG_2)

19.2.7.1 Offset

Register	Offset
TRNG_CTR_DRBG_CONFIG_2	0x001C

19.2.7.2 Function

This is the CTR_DRBG configuration register 2.

19.2.7.3 Fields

Field	Access	Description	Reset Value
31:0 HTP	RW	Number of random generations between two periodic health tests. 'd0 means the number of random generations between two periodic health tests is 'd1. 'dn means the number of random generations between two periodic health tests is n+1.	0x0

19.2.8 TRNG Alarm Register (TRNG_ALARM)

19.2.8.1 Offset

Register	Offset
TRNG_ALARM	0x0020

19.2.8.2 Function

The register provides TRNG alarms.

19.2.8.3 Fields

Field	Access	Description	Reset Value
0 AL0	RO	TRNG alarm bit 0: 0b: Embedded test 0 has successfully passed. 1b: Embedded test 0 (Monobit test) failed.	0x0
1 AL1	RO	TRNG alarm bit 1: 0b: Embedded test 1 has successfully passed. 1b: Embedded test 1 (poker test) failed.	0x0
2 AL2	RO	TRNG alarm bit 2: 0b: Embedded test 2 has successfully passed. 1b: Embedded test 2 (run test) failed.	0x0
3 AL3	RO	TRNG alarm bit 3: 0b: Embedded test 3 has successfully passed. 1b: Embedded test 3 (Long run test) failed.	0x0
4 VA	RO	TRNG Alarm valid bit. 0b: Alarm data is not valid and should not be considered. 1b: Alarm data (bits AL0 to AL3) is valid.	0x0
31:5 RSVD	RO	Reserved	0x0

19.2.9 TRNG CTR_DRBG Alarm Register (TRNG_CTR_DRBG_ALARM)

19.2.9.1 Offset

Register	Offset
TRNG_CTR_DRBG_ALARM	0x0024

19.2.9.2 Function

The register provides CTR-DRBG alarms.

19.2.9.3 Fields

Field	Access	Description	Reset Value
0 AL0	RO	CTR_DRBG alarm bit 0: 0b: Test of 'instantiate' function has been successful. 1b: Test of 'instantiate' function failed.	0x0
1 AL1	RO	CTR_DRBG alarm bit 1: 0b: Test of 'reseed' function has been successful. 1b: Test of 'reseed' function failed.	0x0
2 AL2	RO	CTR_DRBG alarm bit 2: 0b: Test of 'generate' function has been successful. 1b: Test of 'generate' function failed.	0x0
3 VA	RO	CTR_DRBG alarm valid bit. 0b: Alarm data is not valid and should not be considered. 1b: Alarm data (bits AL0 to AL2) is valid. When this bit goes to 1, the DRBG alarm must be acknowledged.	0x1
31:4 RSVD	RO	Reserved	0x0

19.2.10 TRNG Raw Random Control Register (TRNG_RAW_RANDOM_CONTROL)

19.2.10.1 Offset

Register	Offset
TRNG_RAW_RANDOM_CONTROL	0x0028

19.2.10.2 Function

This is the raw random control register. Reading this register will cause bus error.

19.2.10.3 Fields

Field	Access	Description	Reset Value
0 FL	WO	Raw data FIFO flush bit. 0b: No event. 1b: Flush the raw data FIFO.	0x0
31:1 RSVD	RO	Reserved	0x0

19.2.11 TRNG Raw Random Data Register (TRNG_RAW_RANDOM_DATA)

19.2.11.1 Offset

Register	Offset
TRNG_RAW_RANDOM_DATA	0x002C

19.2.11.2 Function

The register provides the raw data.

19.2.11.3 Fields

Field	Access	Description	Reset Value
15:0 DATA	RO	Raw Random data	0x0
16 VA	RO	TRNG raw random data valid bit. 0b: Data is not valid and should not be considered. 1b: Random data is valid.	0x0
31:17 RSVD	RO	Reserved	0x0

19.2.12 TRNG CTR_DRBG Random Data Register (TRNG_CTR_DRBG_RANDOM_DATA)

19.2.12.1 Offset

Register	Offset
TRNG_CTR_DRBG_RANDOM_DATA	0x0030

19.2.12.2 Function

The register provides the raw data.

19.2.12.3 Fields

Field	Access	Description	Reset Value
15:0 DATA	RO	CTR_DRBG random data	0x0
16 VA	RO	CTR_DRBG random data valid bit. 0b: Data is not valid and should not be considered. 1b: Random data is valid.	0x0
31:17 RSVD	RO	Reserved	0x0

19.2.13 TRNG Additional Configuration Register 0 (TRNG_ADDITIONAL_CONFIG_0)

19.2.13.1 Offset

Register	Offset
TRNG_ADDITIONAL_CONFIG_0	0x0034

19.2.13.2 Function

The register provides additional control bits.

19.2.13.3 Fields

Field	Access	Description	Reset Value
0 I0	RW	Monobit test (TRNG embedded tests) ignore bit. 0b: Monobit test result is not ignored. 1b: Monobit test result is ignored.	0x0
1 I1	RW	Poker test (TRNG embedded tests) ignore bit. 0b: Poker test result is not ignored. 1b: Poker test result is ignored.	0x0
2 I2	RW	Run test (TRNG embedded tests) ignore bit. 0b: Run test result is not ignored. 1b: Run test result is ignored.	0x0
3 I3	RW	Long run test (TRNG embedded tests) ignore bit. 0b: Long run test result is not ignored. 1b: Long run test result is ignored.	0x0
4 MC	RW	Manual Control enable bit. 0b: Manual control is disabled. 1b: Manual control is enabled.	0x0
5 RE	RW	Raw data recording enable bit. 0b: Recording of raw data is not forced. 1b: Recording of raw data is forced.	0x0
6 E0	RW	Monobit test (TRNG embedded tests) enable bit. 0b: Monobit test result is disabled. 1b: Monobit test result is enabled.	0x0
7 E1	RW	Poker test (TRNG embedded tests) enable bit. 0b: Poker test result is disabled. 1b: Poker test result is enabled.	0x0
8 E2	RW	Run test (TRNG embedded tests) enable bit. 0b: Run test result is disabled. 1b: Run test result is enabled.	0x0
9 E3	RW	Long run test (TRNG embedded tests) enable bit. 0b: Long run test result is disabled. 1b: Long run test result is enabled.	0x0
10 SE	RW	Noise module enable bit (Manual control) 0b: The noise module is disabled. 1b: The noise module is enabled (when manual control is enabled).	0x0
31:11 RSVD	RO	Reserved	0x0

19.2.14 TRNG Additional Configuration Register 1 (TRNG_ADDITIONAL_CONFIG_1)

19.2.14.1 Offset

Register	Offset
TRNG_ADDITIONAL_CONFIG_1	0x0038

19.2.14.2 Function

The register provides additional control bits.

19.2.14.3 Fields

Field	Access	Description	Reset Value
10:0 AC1	RW	Additional configuration word 1 Recommended value for this parameter is 0x400.	0x0
31:11 RSVD	RO	Reserved	0x0

19.2.15 TRNG Additional Configuration Register 2 (TRNG_ADDITIONAL_CONFIG_2)

19.2.15.1 Offset

Register	Offset
TRNG_ADDITIONAL_CONFIG_2	0x003C

19.2.15.2 Function

The register provides additional control bits.

19.2.15.3 Fields

Field	Access	Description	Reset Value
31:0 AC2	RW	Additional configuration word 2 Recommended value for this parameter is 0xffffffff.	0x0

19.2.16 TRNG Additional Measurement Control Register (TRNG_ADDITIONAL_MEAS_CONTROL)

19.2.16.1 Offset

Register	Offset
TRNG_ADDITIONAL_MEAS_CONTROL	0x0040

19.2.16.2 Function

The register provides configuration for noise measurement. Reading this register will cause bus error.

19.2.16.3 Fields

Field	Access	Description	Reset Value
0 NS	WO	Noise measurement start signal 0b: No event. 1b: The noise measurement starts.	0x0
31:1 RSVD	RO	Reserved	0x0

19.2.17 TRNG Additional Measurement Configuration Register (TRNG_ADDITIONAL_MEAS_CONFIG)

19.2.17.1 Offset

Register	Offset
TRNG_ADDITIONAL_MEAS_CONFIG	0x0044

19.2.17.2 Function

The register provides configuration for noise measurement.

19.2.17.3 Fields

Field	Access	Description	Reset Value
29:0 TIME	RW	Noise measurement time 'dx = time in cycles of the noise measurement.	0x0
31:30 SEL	RW	Noise measurement selection 00b: No source is enabled. 01b: Noise source 1 is enabled. 10b: Noise source 2 is enabled. 11b: Both noise sources are enabled.	0x0

19.2.18 TRNG Additional Measurement Status Register (TRNG_ADDITIONAL_MEAS_STATUS)

19.2.18.1 Offset

Register	Offset
TRNG_ADDITIONAL_MEAS_STATUS	0x0048

19.2.18.2 Function

The register provides status bits for noise measurement.

19.2.18.3 Fields

Field	Access	Description	Reset Value
0 RD	RO	Returns 1 when the noise measurement operation is ready to start.	0x1
1 OV	RO	Returns 1 when the noise measurement operation is over.	0x0

Field	Access	Description	Reset Value
31:2 RSVD	RO	Reserved	0x0

19.2.19 TRNG Additional Measurement Data Register 0 (TRNG_ADDITIONAL_MEAS_DATA_0)

19.2.19.1 Offset

Register	Offset
TRNG_ADDITIONAL_MEAS_DATA_0	0x004C

19.2.19.2 Function

The register provides noise data.

19.2.19.3 Fields

Field	Access	Description	Reset Value
31:0 D0	RO	Return noise measurement data 0.	0x0

19.2.20 TRNG Additional Measurement Data Register 1 (TRNG_ADDITIONAL_MEAS_DATA_1)

19.2.20.1 Offset

Register	Offset
TRNG_ADDITIONAL_MEAS_DATA_1	0x0050

19.2.20.2 Function

The register provides noise data.

19.2.20.3 Fields

Field	Access	Description	Reset Value
31:0 D1	RO	Return noise measurement data 1.	0x0

19.2.21 TRNG Interrupt Request Status Register (TRNG_IRQ_STATUS)

19.2.21.1 Offset

Register	Offset
TRNG_IRQ_STATUS	0x0054

19.2.21.2 Function

The IRQ status register indicates what event is the source of the IRQ.

19.2.21.3 Fields

Field	Access	Description	Reset Value
0 IS0	RO	When this bit is 1, then the “timeout” event has been detected.	0x0
1 IS1	RO	When this bit is 1, then the “random data ready” event has been detected.	0x0
2 IS2	RO	When this bit is 1, then the “TRNG alarm” event has been detected.	0x0
3 IS3	RO	When this bit is 1, then the “DRBG alarm” event has been detected.	0x0
31:4 RSVD	RO	Reserved	0x0

19.2.22 TRNG Interrupt Request Clear Register (TRNG_IRQ_CLEAR)

19.2.22.1 Offset

Register	Offset
TRNG_IRQ_CLEAR	0x0058

19.2.22.2 Function

The register is used for clearing IRQ. Reading this register will cause bus error.

19.2.22.3 Fields

Field	Access	Description	Reset Value
0 CL0	WO	Write 0x1 (automatically goes back to 0) on bit CL0. This bit clears the “Timeout” IRQ.	0x0
1 CL1	WO	Write 0x1 (automatically goes back to 0) on bit CL1. This bit clears the “random data ready” IRQ.	0x0
2 CL2	WO	Write 0x1 (automatically goes back to 0) on bit CL2. This bit clears the “TRNG alarm” IRQ.	0x0
3 CL3	WO	Write 0x1 (automatically goes back to 0) on bit CL3. This bit clears the “DRBG alarm” IRQ.	0x0
31:4 RSVD	RO	Reserved	0x0

19.2.23 TRNG Interrupt Request Configuration Register (TRNG_IRQ_CONFIG)

19.2.23.1 Offset

Register	Offset
TRNG_IRQ_CONFIG	0x005C

19.2.23.2 Function

The register is used for specifying which event produces IRQ.

19.2.23.3 Fields

Field	Access	Description	Reset Value
0 E0	RW	IRQ Enable 0 0b: No event. 1b: The “timeout” event is enabled. (=1 recommended).	0x0
1 E1	RW	IRQ Enable 1 0b: No event. 1b: The “random data ready” event is enabled. (=1 recommended).	0x0
2 E2	RW	IRQ Enable 2 0b: No event. 1b: The “TRNG alarm” event is enabled. (=0 recommended).	0x0
3 E3	RW	IRQ Enable 3 0b: No event. 1b: The “DRBG alarm” event is enabled. (=1 recommended).	0x0
31:4 RSVD	RO	Reserved	0x0

19.2.24 TRNG Error Configuration Register (TRNG_ERROR_CONFIG)

19.2.24.1 Offset

Register	Offset
TRNG_ERROR_CONFIG	0x0060

19.2.24.2 Function

The register is used for TRNG error detection configuration.

19.2.24.3 Fields

Field	Access	Description	Reset Value
31:0 TIMEOUT	RW	Configuration of the timeout value.	0x0

19.2.25 TRNG Error Control Register (TRNG_ERROR_CONTROL)

19.2.25.1 Offset

Register	Offset
TRNG_ERROR_CONTROL	0x0064

19.2.25.2 Function

The register is used for TRNG error detection control. Reading this register will cause bus error.

19.2.25.3 Fields

Field	Access	Description	Reset Value
0 ST	WO	Timeout start bit. 0b: No event. 1b: Starts counting.	0x0
31:1 RSVD	RO	Reserved	0x0

19.2.26 TRNG Error Status Register (TRNG_ERROR_STATUS)

19.2.26.1 Offset

Register	Offset
TRNG_ERROR_STATUS	0x0068

19.2.26.2 Function

The register is used for specifying which event produces IRQ.

19.2.26.3 Fields

Field	Access	Description	Reset Value
0 RD	RO	Timeout module ready. 0b: Timeout module has already counted. 1b: Timeout module is not operating and is ready to start a new operation.	0x1
1 OV	RO	Timeout module over. 0b: No event. 1b: Timeout module has finished its last operation.	0x0
2 TR	RO	Timeout error. 0b: No event. 1b: Timeout is reached, and no random data has been delivered by the TRNG IP.	0x1
31:3 RSVD	RO	Reserved	0x0

19.2.27 TRNG Monitoring Feature Configuration Register (TRNG_STATS_CONFIG)

19.2.27.1 Offset

Register	Offset
TRNG_STATS_CONFIG	0x006C

19.2.27.2 Function

The register is used for configuring the TRNG monitoring feature.

19.2.27.3 Fields

Field	Access	Description	Reset Value
0 MO	RW	0b: No event. 1b: Enables the monitoring feature. Default value is 0.	0x0
31:1 RSVD	RO	Reserved	0x0

19.2.28 TRNG Monitoring Feature Control Register (TRNG_STATS_CONTROL)

19.2.28.1 Offset

Register	Offset
TRNG_STATS_CONTROL	0x0070

19.2.28.2 Function

The register is used for controlling the TRNG monitoring feature. Reading this register will cause bus error.

19.2.28.3 Fields

Field	Access	Description	Reset Value
0 CL	WO	0b: No event. 1b: Reset of the monitoring feature counters is enabled.	0x0
31:1 RSVD	RO	Reserved	0x0

19.2.29 TRNG Monitoring Feature Status Register (TRNG_STATS_DATA)

19.2.29.1 Offset

Register	Offset
TRNG_STATS_DATA	0x0074

19.2.29.2 Function

The register gives the TRNG monitoring feature status.

19.2.29.3 Fields

Field	Access	Description	Reset Value

Field	Access	Description	Reset Value
15:0 EV	RO	The number of sequences that have been evaluated by the embedded test.	0x0
31:16 PA	RO	The number of sequences that have passed the embedded tests.	0x0

19.3 Function description

This section provides the functional information of the TRNG module.

19.3.1 NORMAL mode

In normal mode, the TRNG and its internal CTR-DRBG are controlled by software. The CTR-DRBG generates programmable numbers of random bits of 128 bits in this mode.

19.3.1.1 ENABLE operation

In order to enable the TRNG IP in normal mode, the software must perform the following operations:

1. Write value 0xFFFF in register TRNG_CTR_DRBG_CONFIG_1. This value configures the DRBG to be seeded every 65536 generate requests;
2. Write value 0x3C0 in register TRNG_ADDITIONAL_CONFIG_0;
3. Write value 0x400 in register TRNG_ADDITIONAL_CONFIG_1;
4. Write value 0xFFFFFFFF in register TRNG_ADDITIONAL_CONFIG_2;
5. Write value 0xFFFF in register TRNG_CTR_DRBG_CONFIG_2. This value configures the DRBG to internally perform health tests every 65536 random generations;
6. Write value 0x50141 in register TRNG_CONFIG. This value configures the decimation parameter at 20, the latency parameter at 20, and also enables the TRNG IP in normal mode;
7. If the bit IN of register TRNG_STATUS is equal to 0, then the CTR-DRBG must be instantiated (go to step 8). Otherwise go to the step 10;
8. Wait until the bit RD of register TRNG_STATUS is equal to 1;
9. Set the bit IN of register TRNG_CTR_DRBG_CONTROL;
10. Wait until the bit RD of register TRNG_STATUS is equal to 1;
11. Set the bit GE of register TRNG_CTR_DRBG_CONTROL to trigger CTR-DRBG to generate random data.

19.3.1.2 DISABLE operation

In order to disable the TRNG IP, the software must perform the following operation:

1. Write value 0x0 in register TRNG_CONFIG. This value completely disables the DRBG.

19.3.1.3 GET RANDOM operation

In order to read random data, the software must perform the following operations:

1. The CPU can read the register TRNG_STATUS in order to know if the TRNG IP is ready to accept a new operation. If the bit RD is equal to 1, then the TRNG IP is ready to

accept a new operation;

2. The user must define the size of the request (number of 128-bit random data) by writing in register TRNG_CTR_DRBG_REQUEST_SIZE;
3. The user must start the generation of random data by writing 1 in bit GE of register TRNG_CTR_DRBG_CONTROL. The bit RD of register TRNG_STATUS also goes to 0;
4. The user iteratively reads the register TRNG_CTR_DRBG_RANDOM_DATA. If the bit VA is high, then the user can use DATA as random data in its application;
5. The TRNG IP delivers an IRQ when all the random data has been generated. The bit RD of register TRNG_STATUS also goes back to 1.

19.3.2 Stream mode

The CTR-DRBG continuously generates stream of random bits in this mode. If the stream mode is enabled, the instantiation and generation of DRBG functions are automatically launched and much less control is required. Users are advised to use the stream mode.

19.3.2.1 ENABLE operation

In order to enable the TRNG IP in stream mode, the software must perform the following operations:

1. Write value 0xFFFF in register TRNG_CTR_DRBG_CONFIG_1. This value configures the DRBG to be seeded every 65536 generate requests;
2. Write value 0x3C0 in register TRNG_ADDITIONAL_CONFIG_0;
3. Write value 0x400 in register TRNG_ADDITIONAL_CONFIG_1;
4. Write value 0xFFFFFFFF in register TRNG_ADDITIONAL_CONFIG_2;
5. Write value 0xFFFF in register TRNG_CTR_DRBG_CONFIG_2. This value configures the DRBG to internally perform health tests every 65536 random generations;
6. Write value 0xFF in register TRNG_CTR_DRBG_REQUEST_SIZE;
7. Write value 0x50143 in register TRNG_CONFIG. This value configures the decimation parameter at 20, the latency parameter at 20, and also enables the TRNG IP in stream mode.

19.3.2.2 DISABLE operation

In order to disable the TRNG IP, the software must perform the following operation:

1. Write value 0x0 in register TRNG_CONFIG. This value completely disables the DRBG.

19.3.2.3 GET RANDOM operation

In order to read random data, the software must perform the following operation:

1. The user iteratively reads the register TRNG_CTR_DRBG_RANDOM_DATA. If the bit VA is high, then the data can be used, otherwise the data must be ignored.

19.3.3 IRQ management

An interrupt is generated if one of the following events is detected:

1. The timeout has not detected any random data during the time specified on input signal;
2. All the random data has been generated;
3. TRNG raises an alarm;
4. DRBG raises an alarm.

These events can be inhibited with the bits E0 (timeout), E1 (random data ready), E2 (TRNG alarm) and E3 (DRBG alarm) of register `IRQ_CONFIG`. If one of these bits is set to 1, the corresponding event will be detected and the interrupt can be generated.

When an event is detected, the corresponding bit in register `IRQ_STATUS` rises.

The interrupt remains active until it has been cleared by the CPU.

In order to clear the “timeout” IRQ event, the CPU must write 1 in bit CL0 of register `IRQ_CLEAR`, this bit is automatically reset.

In order to clear the “random data ready” IRQ event, the CPU must write 1 in bit CL1 of register `IRQ_CLEAR`, this bit is automatically reset.

In order to clear the “TRNG alarm” IRQ event, the CPU must write 1 in bit CL2 of register `IRQ_CLEAR`, this bit is automatically reset.

In order to clear the “DRBG alarm” IRQ event, the CPU must write 1 in bit CL3 of register `IRQ_CLEAR`, this bit is automatically reset.

19.3.4 Timeout management

After power-up, the CPU can launch the timeout feature, by performing the following operations:

1. Write the timeout time (number of cycles) into register `TRNG_ERROR_CONFIG`;
2. Start the timeout counting by writing 0x1 in register `TRNG_ERROR_CONTROL`;
3. Enable the TRNG, which refers to the ENABLE Operation;
4. Wait for interrupt (bit IS0 of register `IRQ_STATUS` = 1) or pull on register `TRNG_ERROR_STATUS`. When timeout is over, the bit OV is equal to 1. Bit TR = 1 indicates that no random data has been generated during the specified time.

Chapter 20 Cyclic Redundancy Check (CRC)

20.1 Introduction

The chapter defines a Cyclic Redundancy Check (CRC).

To detect errors, the Cyclic Redundancy Check (CRC) module generates a 16/32-bit CRC code, which is calculated for 32 bits of data at a time.

To implement a 16-bit or 32-bit CRC code, the CRC module provides a programmable polynomial and other parameters.

20.1.1 Features

The CRC supports the following key features:

- Hardware CRC generator circuit with a programmable shift register of a 16/32-bit CRC code
- Programmable initial seed value and polynomial
- Transpose the input data or output data (the CRC result) bit-wise or byte-wise. This option is required for certain CRC standards. However, the user cannot transpose a byte-wise transpose operation when accessing the CRC data register via 8-bit accesses. In this case, reading 32 bits is preferred to perform the byte-wise transpose function.
- Inversion of the final CRC results
- A programming interface for the 32-bit CPU register

20.1.2 Block diagram

The following figure is a block diagram of the Cyclic Redundancy Check (CRC) module.

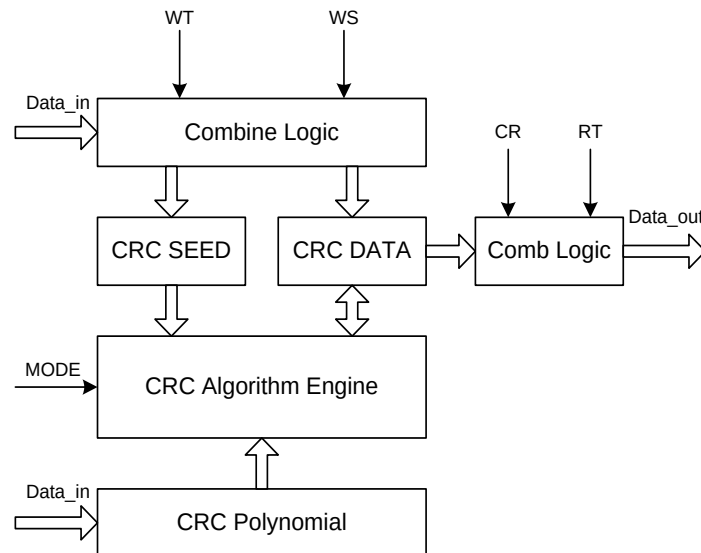


Figure 20-1 CRC block diagram

20.2 Memory map and register definition

This section provides the memory map and register definitions of the CRC module.

20.2.1 CRC memory map

CRC base address: 0x40032000

Table 20-1 CRC memory map

Offset	Register	Width (In bits)	Reset Value
0x8	CRC Data Register (CRC_DATA)	32	0xFFFFFFFF
0xC	CRC Polynomial Register (CRC_POLY)	32	0x00001021
0x10	CRC Control Register (CRC_CTRL)	32	0x00000000

20.2.2 CRC Data Register (CRC_DATA)

20.2.2.1 Offset

Register	Offset
CRC_DATA	0x8

20.2.2.2 Function

The CRC Data register supports 8/16/32-bit access.

The CRC Data register contains the value of the seed, data, and checksum:

When CRC_CTRL[WS] = 1, any write to this register is regarded as the seed value;

When CRC_CTRL[WS] = 0, any write to this register is regarded as CRC general data;

After all data values are written, the user can read the CRC result from this data register.

20.2.2.3 Fields

Field	Access	Description	Reset Value
7:0 BYTE0	RW	Seed written or data written to this field is for CRC checksum generation in both 16-bit and 32-bit CRC modes.	0xFF
15:8 BYTE1	RW	Seed written or data written to this field is for CRC checksum generation in both 16-bit and 32-bit CRC modes.	0xFF
23:16 BYTE2	RW	Only used in the 32-bit CRC mode to write a seed value. Data written to this field is for the CRC checksum generation in both 16-bit and 32-bit CRC modes.	0xFF
31:24 BYTE3	RW	Only used in CRC 32-bit mode to write a seed value. Data written to this field is for the CRC checksum generation in both 16-bit and 32-bit CRC modes.	0xFF

20.2.3 CRC Polynomial Register (CRC_POLY)

20.2.3.1 Offset

Register	Offset
CRC_POLY	0xC

20.2.3.2 Function

This register contains the value of the polynomial for the CRC calculation.

20.2.3.3 Fields

Field	Access	Description	Reset Value
15:0 LPHW	RW	Used in both 32-bit CRC mode and 16-bit CRC mode	0x1021
31:16 HPHW	RW	Only used in 32-bit CRC mode	0x0

20.2.4 CRC Control Register (CRC_CTRL)

20.2.4.1 Offset

Register	Offset
CRC_CTRL	0x10

20.2.4.2 Function

This register controls the configuration and working of the CRC module.

20.2.4.3 Fields

Field	Access	Description	Reset Value
0 MODE	RW	0b: 16-bit CRC protocol mode. 1b: 32-bit CRC protocol mode.	0x0
1 WS	RW	0b: Writes to the CRC data register are data values. 1b: Writes to the CRC data register are seed values.	0x0
2 CR	RW	Some CRC protocols require the final checksum to be XORed with 0xFFFFFFFF or 0xFFFF. Asserting this bit enables on the fly complementing of the read data. 0b: No XOR on reading. 1b: Invert or complement the read value of the CRC Data register.	0x0
3 RSVD_3	RO	Reserved	0x0
5:4 RT	RW	Identifies the transpose configuration of the value read from the CRC Data register: 00b: No transposition. 01b: Bits in bytes are transposed. Bytes are not transposed. 10b: Both bits in bytes and bytes are transposed. 11b: Only bytes are transposed. No bits in a byte are transposed.	0x0
7:6 WT	RW	Defines the transpose configuration of the data written to the CRC data register: 00b: No transposition. 01b: Bits in bytes are transposed. Bytes are not transposed. 10b: Both bits in bytes and bytes are transposed. 11b: Only bytes are transposed. No bits in a byte are transposed.	0x0
31:8 RSVD_31_8	RO	Reserved	0x0

20.3 Function description

20.3.1 CRC initialization/reinitialization

To enable CRC calculation, the user needs to program the following parameters in the applicable registers: CRC_CTRL[WS], CRC_POLY, necessary parameters for transposition and CRC result inversion. By asserting CRC_CTRL[WS], the user can program the seed value into the CRC_DATA register.

Once a CRC calculation is completed, the module can be reinitialized for a new CRC computation by reasserting CRC_CTRL[WS] and programming a new or previously used seed value. It is important for the user to set all other parameters before programming the seed value and subsequent data values.

20.3.2 CRC calculations

In 16-bit and 32-bit CRC modes, with all the provided bytes contiguous, data values can be programmed as 8 bits, 16 bits, or 32 bits at a time. Non-contiguous bytes can lead to an incorrect CRC computation.

20.3.3 16-bit CRC

To compute a 16-bit CRC:

1. Clear CRC_CTRL[MODE] to enable the 16-bit CRC mode.
2. Program the transpose and complement options in the CRC_CTRL register as required for the CRC calculation.
3. Write a 16-bit polynomial to the CRC_POLY[15:0] field. The CRC_POLY[31:16] field is not usable in 16-bit CRC mode.
4. Set CRC_CTRL[WS] to program the seed value.
5. Write a 16-bit seed to CRC_DATA[15:0]. CRC_DATA[31:16] are not used.
6. Clear CRC_CTRL[WS] to start writing data values.
7. Write data values into CRC_DATA. A CRC is computed on every data value write, and the intermediate CRC result is stored back into CRC_DATA[15:0] or CRC_DATA[31:16].
8. When all values have been written, user should read the final CRC result from CRC_DATA[15:0] or CRC_DATA[31:16].

NOTE

If CRC_CTRL[RT] = 00b or 01b, the final CRC result is stored back into CRC_DATA[15:0]. If CRC_CTRL[RT] = 10b or 11b, the final CRC result is stored back into CRC_DATA[31:16].

20.3.4 32-bit CRC

To compute a 32-bit CRC:

1. Set CRC_CTRL[MODE] to enable the 32-bit CRC mode.
2. Program the transpose and complement options in the CRC_CTRL register as required for the CRC calculation.
3. Write a 32-bit polynomial to CRC_POLY[31:0].
4. Set CRC_CTRL[WS] to program the seed value.
5. Write a 32-bit seed to CRC_DATA.
6. Clear CRC_CTRL[WS] to start writing data values.
7. Write data values into CRC_DATA. A CRC is computed on every data value write, and

the intermediate CRC result is stored back into CRC_DATA[31:0].

8. When all values have been written, read the final CRC result from CRC_DATA[31:0]. The CRC is calculated byte-wise, and two clocks are required to complete one CRC calculation.

20.3.5 Transpose feature

By default, the transpose feature is not enabled. However, some CRC standards require the input data and/or the final checksum to be transposed. The user software has the option to configure each transpose operation separately, according to the CRC standard. The data is transposed on the fly while being read or written.

Some protocols use little endian format for the data stream to calculate a CRC. In this case, the transpose feature flips the bits usefully. This transpose option is one of the types supported by the CRC module.

The CRC module provides several types of transpose functions to flip the bits and/or bytes, for both writing input data and reading the CRC results. The CRC module separately uses the CRC_CTRL[WT] or CRC_CTRL[RT] fields, according to the CRC calculation being used.

The following types of transpose functions are available for writing to and reading from the CRC data register:

1. CRC_CTRL[WT] or CRC_CTRL[RT] is 00. No transposition occurs.
2. CRC_CTRL[WT] or CRC_CTRL[RT] is 01. Bits in a byte are transposed, while bytes are not transposed. Reg[31:0] becomes {reg[24:31], reg[16:23], reg[8:15], reg[0:7]}
3. CRC_CTRL[WT] or CRC_CTRL[RT] is 10. Both bits in bytes and bytes are transposed. Reg[31:0] becomes = {reg[0:7], reg[8:15], reg[16:23], reg[24:31]}
4. CRC_CTRL[WT] or CRC_CTRL[RT] is 11. Bytes are transposed. But bits are not transposed. Reg[31:0] becomes {reg[7:0], reg[15:8], reg[23:16], reg[31:24]}

20.3.6 CRC result complement

When CRC_CTRL[CR] is set, the checksum is in the complemented read mode. The CRC result complement function outputs the complement of the checksum value stored in the CRC data register every time the CRC data register is read. When CRC_CTRL[CR] is cleared, reading the CRC data register will access the raw checksum value.

20.4 Initialization application information

1. Write to CRC_CTRL[MODE] to select CRC calculation mode.
2. Write to CRC_CTRL[CR] to select complement read, include No XOR on reading and invert or complement the read value of the CRC_DATA register.
3. Write to CRC_POLY register to set the value of the polynomial.
4. Write to CRC_CTRL[WT] to select the write type of transpose
5. Write to CRC_CTRL[RT] to select the read type of transpose
6. Configure the seed value.

Chapter 21

Reset and Boot

21.1 Introduction

The Z20K14xMC Series supports the following reset sources:

- POR reset: Power-on reset (POR)
- LVD reset: Low-voltage detect (LVD)
- SERU cold reset
- System resets:
 - External pin reset (PIN)
 - Watchdog reset
 - Loss-of-clock (LOC) reset
 - Low power mode acknowledge error (SACKERR)
 - Software reset (SW)
 - Lockup reset (LOCKUP)
 - ADM-AP system reset
 - SERU normal reset
 - Wake up reset from STANDBY mode

21.2 Reset

This section introduces the basic reset mechanisms and sources. We can configure some peripheral modules to generate interrupts instead of causing a reset. See the related chapters for more details.

Every reset source has an associated field in the System Reset and Mode Control (SRMC) System Reset Status (SRMC_SRS) register. In addition to the immediate system reset, the SRMC also supports the system reset's optional delays while an interrupt is generated. This feature provides an option to perform a graceful shutdown.

The MCU exits reset in a RUN mode in which the CPU is running the code.

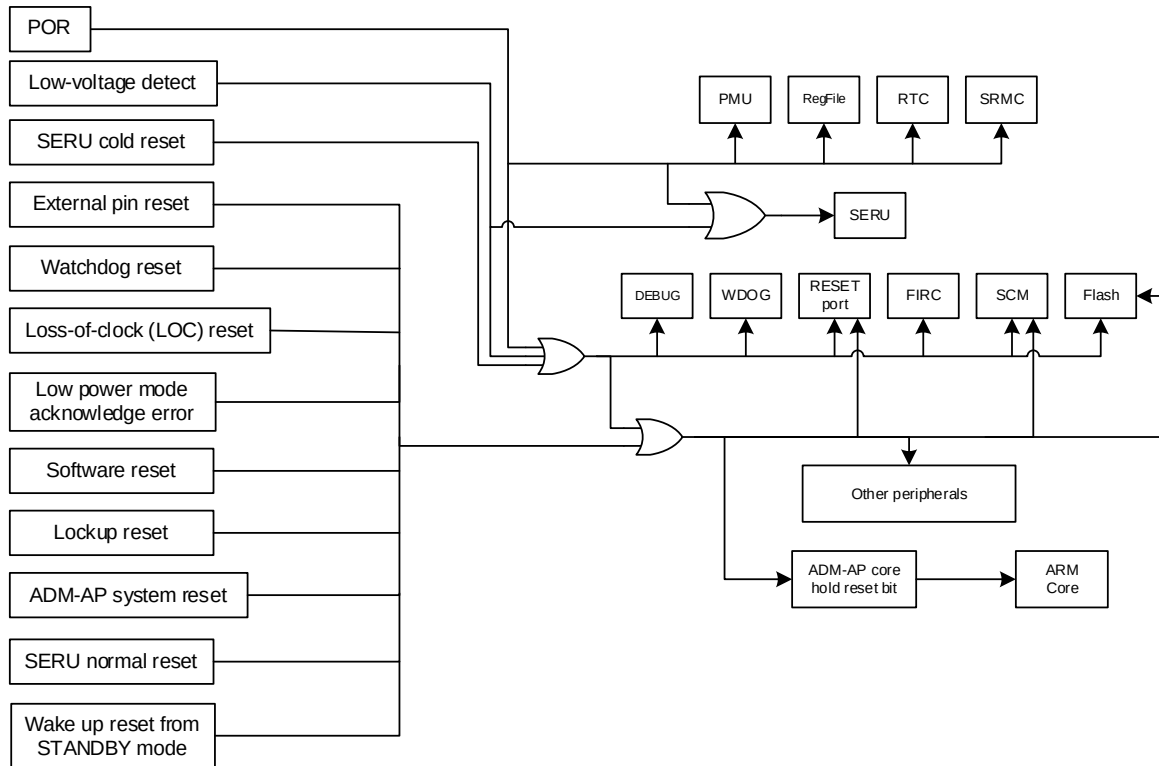


Figure 21-1 Reset system block diagram

21.2.1 Power-on reset (POR)

When the user applies power to the MCU initially or when the supply voltage drops below the POR detect voltage (V_{POR}), the POR circuit causes a POR reset condition.

As the supply voltage rises, the LVD circuit holds the MCU in reset until the supply has risen above the LVD low voltage detect threshold (V_{LVD}). The POR and LVD bits in SRMC_SRS register are set following a POR.

Note: SRMC_SSRS and SRMC_SRS will be cleared by POR or LVD event, so both POR and LVD bits in SRMC_SRS/SRMC_SSRS register will be set, or only LVD will be set following a POR.

21.2.2 Low voltage detect (LVD)

The chip integrates with a system for managing low voltage conditions to protect memory contents and control MCU system states during supply voltage variations. The system includes a POR circuit and an LVD circuit. The LVD system can be disabled when entering into the STOP mode to reduce power consumption.

The LVD system can be configured to generate a reset upon detection of a low voltage condition by writing 1 to PMU_CSR [VDD_LVD_RE]. After an LVD reset has occurred, the LVD system holds the MCU in reset until the supply voltage has risen above the low voltage detection threshold. The MCU writes 1 to SRMC_SRS[LVD] bit is set following either an LVD reset or POR.

During the POR, both SRMC_SRS[POR] and SRMC_SRS[LVD] are configured as 1 by MCU.

While performing a normal operation, MCU writes 1 to both SRMC_SRS[POR] and SRMC_SRS[LVD] if device supply goes below V_{POR} . If device supply goes below V_{LVD} (provided LVD is enabled to generate a reset) but above V_{POR} , MCU writes 1 to SRMC_SRS[LVD].

21.2.3 SERU cold reset

The SERU module collects system error report. If a system error occurs and SERU is allowed to request cold reset, the system reset is blocked because of its own causes. SERU issues cold reset request to SRMC to reset system and itself. The reset causes the MCU to write 1 to SRMC_SRS[SERU_COLD] bit.

21.2.4 System reset sources

The user can start processing from a known set of initial conditions via resetting the MCU. System reset begins with the on-chip regulator in full regulation and system clocking generation from an internal reference. When the processor exits reset, it:

- Reads the start stack pointer (SP) (SP_main) from vector-table offset 0
- Reads the start program counter (PC) from vector-table offset 4
- Writes 0xFFFF_FFFF to the Link Register (LR)

The on-chip peripheral modules are disabled, and the non-analog I/O pins are initially configured as disabled. The pins with analog functions assigned to them are configured for their analog functions after reset.

21.2.4.1 External pin reset

RESET_B is a shared pin (See Signal Multiplexing and Pin Assignment). The user must configure the corresponding PORT_PCR register before using this pin as RESET_B. On any reset event, the MCU configures the corresponding pad as RESET_B pad. Asserting RESET_B wakes the MCU from any mode.

- The system clock is required to be available for the proper functionality of Reset Pin. During a pin reset, the MCU writes 1 to SRMC_SRS[PIN].
- The RESET_B has IO passive filter, IO digital filter, and SRMC filter.
- The filters are asynchronously reset on-chip POR. The reset value for each filter assumes the RESET_B pin is negated.
- In a STOP mode in which the LPO clock is still active, the only filtering option is the LPO clock filter. The filtering logic either switches to bypass operation or has continued filtering operation depending upon the filtering mode selected by PORT_PCR[FILT].
- By default, the RESET_B pin and NMI_B pin also have a passive filter. The corresponding PORT_PCR[PFE] controls the RESET_B pin and NMI_B pin.

21.2.4.2 Watchdog reset

Watchdog timer (WDOG) monitors the operation of the system by expecting periodic communication from the software. This communication is generally known as servicing or refreshing the watchdog. If this periodic refreshing does not occur, the watchdog issues a system reset. The watchdog reset causes the MCU to write 1 to SRMC_SRS[WDOG].

21.2.4.3 Loss-of-Clock (LOC) reset

The SCC module supports clock monitor of OSC and FIRC.

If SCC_OSCCS[OSCCME] is set to 1 and SCC_OSCCS[OSCCMRE] is set to 1, SCC_OSCCS[OSCLOC] will be set to 1 and MCU will be reset when OSC clock is lost.

If SCC_FIRCCS[FIRCCME] is set to 1 and SCC_FIRCCS[FIRCCMRE] is set to 1, SCC_FIRCCS[FIRCLOC] will be set to 1 and MCU will be reset when FIRC clock is lost.

The MCU writes 1 to the SRMC_SRS[LOC] field to indicate this reset source. To prevent unexpected loss of clock reset events, all clock monitors should be disabled prior to entering into STOP modes.

21.2.4.4 Low power mode acknowledge error (SACKERR)

This reset is generated if the core attempts to enter the low power mode (STOP mode or STANDBY mode), but not all modules acknowledge low power mode request within configurable timeout (refer to SRMC_PMCTRL[DSMACKTMO]). The SACKERR reset causes the MCU to write 1 to SRMC_SRS[SACKERR] field.

21.2.4.5 Software reset (SW)

The SYSRESETREQ bit in the NVIC application interrupts and reset control register can be written to 1 to force a software reset. (See ARM's NVIC documentation for the full description of the register fields, especially the VECTKEY field requirements.) Writing 1 to SYSRESETREQ generates a software reset request. This reset forces a system reset of all major components except the debug module. A software reset causes the MCU to write 1 to SRMC_SRS[SW] field.

21.2.4.6 Lockup reset (LOCKUP)

The LOCKUP gives an immediate indication of seriously errant kernel software. This results from that the core is locked due to an unrecoverable exception following the activation of the processor's built-in system state protection hardware.

The LOCKUP condition causes a system reset and causes the MCU to write 1 to SRMC_SRS[LOCKUP].

21.2.4.7 ADM-AP system reset

Writing 1 to the system reset request field in the ADM-AP control register initiates a system reset. This is the primary method for resets via the SWD or JTAG interface. The system reset is held until this field is written to 0.

Writing 1 to the core hold reset bit in the ADM-AP control register holds the core in reset as the rest of the chip comes out of system reset.

21.2.4.8 SERU normal reset

The SERU module collects system error report. If a system error occurs and SERU is allowed to request normal reset, SERU issues a system reset request to SRMC to reset system. The reset causes the MCU to write 1 to SRMC_SRS[SERU_SYS] field.

21.2.4.9 Wake up reset from STANDBY mode

When MCU is waked up from STANDBY mode by available wakeup sources, the wakeup reset request will generate. In this MCU, wakeup sources are composed of RTC, Pin reset, LVW/LVD/POR reset and wakeup pins (PTA25, PTE13, PTE11, PTD0, PTD1, PTE15, PTE16, PTA8, PTA9, PTA4, PTC4, PTC5, PTE0, PTE1).

The Wakeup condition causes a system reset and also causes the MCU to write 1 to SRMC_SRS[WAKEUP] field.

21.2.5 Reset pin

The RESET_B pin for all reset sources is driven low by the MCU for at least 128 bus clock cycles and until flash memory initialization starts.

Keeping the RESET pin asserted externally delays the negation of the internal Chip Reset.

21.3 Boot

This section describes the boot sequence, including sources and options.

21.3.1 Boot sources

This MCU supports cold booting from internal flash memory.

When the MCU boots from internal flash memory, the reset vectors are located at address 0x0 (initial SP_main) and 0x4 (initial PC).

With software developing, the MCU can support serial code download via

- CAN
- LIN
- SPI etc.

21.3.2 Boot sequence

The on-chip regulator holds the system in a POR state at power-up until the input supply is above the POR threshold. The system continues to be held in this static state until the internally regulated supplies have reached a safe operating voltage as determined by the LVD. The mode controller reset logic then controls this sequence to exit reset.

1. A system reset is held on internal logic, the RESET_B pin is driven out low, and the SCC is enabled in its default clocking mode.
2. Required clocks are enabled (core clock, system clock, flash clock, and any bus clocks that do not have clock gate control reset to disabled).
3. After the RESET_B pin is released, the flash controller begins initialization operation while keeping flash core hold = 1.
4. When flash memory initialization completes, the flash core hold is released. The core clock is enabled, and the system is released from reset.
5. When the system exits reset, the processor sets up the stack, program counter (PC), and link register (LR). The processor reads the start SP (SP_main) from vector-table offset 0. The core reads the start PC from vector-table offset 4. LR is written to 0xFFFF_FFFF.

Chapter 22

Clock Distribution

22.1 Introduction

This chapter introduces the clock architecture overview of this device, the clock distribution, and module clocks, and provides a clock terminology section.

The System Clock Controller (SCC) module is used on this device for the main system clock generation. The SCC module controls which clock source (internal references, external crystals or external clock signals) is used to derive the system clocks. The SCC also divides the selected clock source into various clock domains, including the clocks for the system bus masters, system bus slaves: core_clk, bus_clk, and slow_clk, which are used by various parts of the SOC. The clock generation logic also implements module-specific clock gating to allow granular shutoff of modules.

The primary clocks for the system are generated from the SCCOUT clock. The clock generation circuitry provides several clock dividers that allow different portions of the device to be clocked at different frequencies. This allows for trade-offs between performance and power dissipation.

Various modules have module-specific clocks from the FIRC_CLK, OSC_CLK, OSC32K_CLK and power management unit (PMU) clock signal (LPO32K_CLK). Besides, various other module-specific clocks have other alternate sources. The PARCC module controls clock selection for most modules.

22.2 High level clocking diagram

The following diagram shows the high-level clocking architecture and various clock sources for this device.

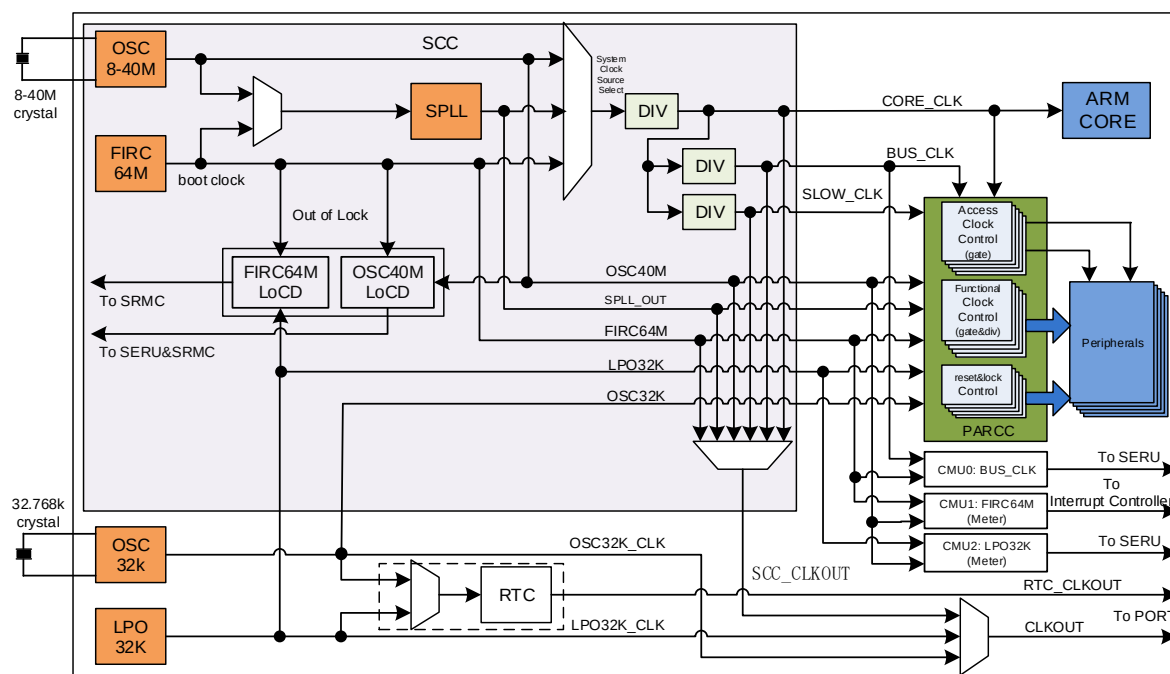


Figure 22-1 Clock Diagram

22.3 Clock definitions

The following table describes clocks shown in the diagram and other sections of this chapter.

Table 22-1 Clock descriptions

Clock name	Related clock selector	Related clock divider	Description
CORE_CLK	SCC_CFG[SCS]	SCC_CFG [DIVCORE] ($\div$ 1...16)	Clocks the ARM core, the Crossbar, NVIC, Flash, SRAM, TIM, GPIO, CAN, and so on, divided by DIVCORE bits inside SCC.
BUS_CLK	SCC_CFG[SCS]	SCC_CFG[DIVBUS] ($\div$ 1...16)	Clocks the chip peripherals.
SCC_SLOW_CLK in SCC	SCC_CFG[SCS]	SCC_CFG[DIVSLOW] ($\div$ 1...16)	This clock can be selected by the flash module to execute some Flash commands.
FIRC_CLK	—	—	Output clock of Fast IRC.
RTC_CLKOUT	—	—	RTC oscillator output driving external pin.
OSC32K_CLK	—	—	Source clock for RTC.
LPO32K_CLK	—	—	Always on low power oscillator clock generated by PMU.
SCC_CLKOUT	SCC_CLKOUTCFG [CLKOUTSEL]	—	SCC output clock that can be driven by OSC40M, FIRC64M, SPL, or CORE_CLK, BUS_CLK, SLOW_CLK.
CLKOUT	SCM_CHIPCTL[CLKOUTSEL]	SCM_CHIPCTL[CLKOUT DIV] ($\div$ 1,2,4,8)	Selected from one of the four internal clock sources.

22.4 Internal clocking requirements

The SCC module's clock divider registers program the clock dividers. Follow the requirements below when configuring the clocks for this chip:

- CORE_CLK and BUS_CLK must be less than or equal to 160 MHz and 80M respectively in RUN mode.
- Select slow clock as flash function clock, its frequency must be programmed to 8 MHz to execute most Flash commands, for example phrase program, sector erase, mass erase, etc.

OSC40M is used for OSC, SPL is configured to 160M.

The following is the common clock configuration for this chip in the normal clocking mode:

- SCC_CFG[SCS] = 011b
- SCC_CFG[DIVCORE] = 0000b
- SCC_CFG[DIVBUS] = 0001b
- SCC_CFG[DIVSLOW] = 0011b

- PARCC_FLASH[CLKMUX] = 110b
- PARCC_FLASH[CLKDIV] = 100b

Table 22-2 RUN mode example

Clock	Frequency
CORE_CLK	160 MHz
BUS_CLK	80 MHz
FLASH function clock	8 MHz

The default configuration out of reset has the CPU clocked by the Fast IRC (FIRC_CLK). The clocks (for example, CORE_CLK, FLASH function clock, and BUS_CLK) are configured in the SCC module for Flash initializing.

- SCC_CFG[SCS] = 001b
- SCC_CFG[DIVCORE] = 0001b
- SCC_CFG[DIVBUS] = 0001b
- SCC_CFG[DIVSLOW] = 0011b
- PARCC_FLASH[CLKMUX] = 011b
- PARCC_FLASH[CLKDIV] = 0111b

Table 22-3 Default clock setting

Clock	Frequency
CORE_CLK	32 MHz
BUS_CLK	16 MHz
FLASH function clock	8 MHz

22.5 Clock gating

The clock to each module can be individually gated on and off using the PARCC module. After any reset, PARCC disables the clock to the corresponding module to conserve power. Before initializing a module, set the corresponding clock gating control bits in the PARCC register to enable the clock. Before turning off the clock, make sure to disable the module. Any bus access to a peripheral that has its clock disabled generates an error termination.

22.6 Peripheral clock definition

The following table summarizes the clocks associated with each module.

Table 22-4 Peripheral clock summary

Modules	Interface Clock	Gating	Peripheral Functional Clock	Default Functional Clock	Functional Clock Divider	Comments
Memory						
SRAM_U	CORE_CLK	NO	CORE_CLK	CORE_CLK	NA	
SRAM_L	CORE_CLK	NO	CORE_CLK	CORE_CLK	NA	
Flash controller	BUS_CLK	YES	0: OFF 1: OFF 2: OFF 3: FIRC64M 4: OSC40M 5: SPILL 6: SLOW_CLK	3: FIRC64M	[3:0]	Function clock should be set to 8M. In STOP mode, as for the function clock, SLOW_CLK must be selected as clock source. After MCU

Modules	Interface Clock	Gating	Peripheral Functional Clock	Default Functional Clock	Functional Clock Divider	Comments
						reset, FIRC64M, OSC40M, SPLL and SLOW_CLK can be selected. Note: CORE_CLK is used for array access
System Module						
DMA	CORE_CLK	NO	CORE_CLK	CORE_CLK	NA	
DMAMUX	CORE_CLK	NO	CORE_CLK	CORE_CLK	NA	
SMPU	CORE_CLK	NO	CORE_CLK	CORE_CLK	NA	
GPIO	CORE_CLK	NO	CORE_CLK	CORE_CLK	NA	
PORT	BUS_CLK	YES	0: OFF 1: LPO32K 2: OSC32K 3: FIRC64M 4: OSC40M	0: OFF	[3:0]	Function Clock <= 64M
PMU	BUS_CLK	NO	LPO_CLK	LPO_CLK	NA	
SCM	BUS_CLK	NO	BUS_CLK	BUS_CLK	NA	
SRMC	BUS_CLK	NO	CORE_CLK	CORE_CLK	NA	
SERU	BUS_CLK	NO	FIRC64M	FIRC64M	NA	
CMU0	BUS_CLK	NO	BUS_CLK	BUS_CLK	NA	BUS_CLK is monitored
CMU1	BUS_CLK	NO	BUS_CLK	BUS_CLK	NA	FIRC64M is monitored
CMU2	BUS_CLK	NO	BUS_CLK	BUS_CLK	NA	LPO32K is monitored
EIRU	CORE_CLK	NO	CORE_CLK	CORE_CLK	NA	
Timers						
WDOG	BUS_CLK	YES	selected by WDOG register	LPO_CLK	NA	Function Clock < BUS_CLK
EWDT	BUS_CLK	YES	0: OFF 1: LPO32K 2: OSC32K 3: FIRC64M 4: OSC40M	0: OFF	[3:0]	Function Clock < BUS_CLK
RTC	BUS_CLK	NO	selected by RTC register	LPO_CLK	NA	
STIM	BUS_CLK	YES	0: OFF 1: LPO32K 2: OSC32K 3: FIRC64M 4: OSC40M 5: SPLL	0: OFF	[3:0]	Function Clock <= 160M
TIM	CORE_CLK	YES	0: OFF 1: LPO32K	0: OFF	[3:0]	

Modules	Interface Clock	Gating	Peripheral Functional Clock	Default Functional Clock	Functional Clock Divider	Comments
			2: OSC32K 3: FIRC64M 4: OSC40M 5: SPLL			
MCPWM	CORE_CLK	YES	0: OFF 1: LPO32K 2: OSC32K 3: FIRC64M 4: OSC40M 5: SPLL	0: OFF	[3:0]	
TDG	CORE_CLK	YES	0: OFF 1: LPO32K 2: OSC32K 3: FIRC64M 4: OSC40M 5: SPLL	0: OFF	[3:0]	Function Clock <= 160M
Communication IPs						
CAN FD	BUS_CLK	YES	0: OFF 1: OFF 2: OFF 3: FIRC64M 4: OSC40M 5: SPLL	0: OFF	[3:0]	Function Clock <= BUS_CLK
UART_LIN	BUS_CLK	YES	0: OFF 1: LPO32K 2: OSC32K 3: FIRC64M 4: OSC40M 5: SPLL	0: OFF	[3:0]	Function Clock <= BUS_CLK
SPI	BUS_CLK	YES	0: OFF 1: LPO32K 2: OSC32K 3: FIRC64M 4: OSC40M 5: SPLL	0: OFF	[3:0]	Function Clock <= BUS_CLK
I2C	BUS_CLK	YES	0: OFF 1: LPO32K 2: OSC32K 3: FIRC64M 4: OSC40M 5: SPLL	0: OFF	[3:0]	Function Clock <= BUS_CLK
I2S	BUS_CLK	YES	0: OFF 1: LPO32K 2: OSC32K 3: FIRC64M 4: OSC40M 5: SPLL	0: OFF	[3:0]	Function Clock <= 1/2 BUS_CLK
Analog IPs						
ADC	BUS_CLK	YES	0: OFF 1: LPO32K 2: OSC32K 3: FIRC64M 4: OSC40M	0: OFF	[3:0]	Function Clock <= BUS_CLK

Modules	Interface Clock	Gating	Peripheral Functional Clock	Default Functional Clock	Functional Clock Divider	Comments
CMP	BUS_CLK	YES	BUS_CLK	BUS_CLK	NA	
Security IPs						
CRC	BUS_CLK	YES	BUS_CLK	BUS_CLK	NA	
AES-128	BUS_CLK	YES	BUS_CLK	BUS_CLK	NA	
TRNG	BUS_CLK	YES	BUS_CLK	BUS_CLK	NA	

Chapter 23

System Clock Controller (SCC)

23.1 Introduction

This chapter defines a System Clock Controller (SCC). The SCC block provides the system clocks of the MCU, and it contains a fast internal RC clock (FIRC) and the oscillator clock (OSC), the system phase-locked loop (SPLL) clock. The SCC can select either the output clock of FIRC, OSC or SPLL as the source for the MCU system clock. The SCC also supports operation with crystal oscillators, which allows an external crystal, ceramic resonator, or another external clock source to produce the external reference clock.

23.2 Features

The SCC supports the following key features:

- Crystal oscillator (OSC):
 - Optional frequency mode for a high-speed mode and a low-speed mode.
 - Optional Loss of clock (LOC) detector to request system reset or interrupt.
 - Optional crystal bypass mode.
 - Optional enable or disable in WAIT, STOP, and DEBUG modes.
- Internal RC oscillator (FIRC):
 - Optional Loss of clock (LOC) detector to request system reset or interrupt.
 - Optional enable or disable in STOP mode.
- System Phase-Locked Loop (SPLL):
 - Voltage-controlled oscillator (VCO).
 - Programmable post divider for VCO clock output.
 - Programmable pre divider for VCO reference clock input.
 - OSC or FIRC can be selected as reference clock.
 - Optional enable or disable in STOP mode.
- Either SPLL, OSC or FIRC can be selected as system clock.
- Up to 16 frequency divisions for generating core/bus/slow clock from system clock. The division step is 1.

23.2.1 Block diagram

The following figure is a block diagram of the System Clock Controller (SCC) module.

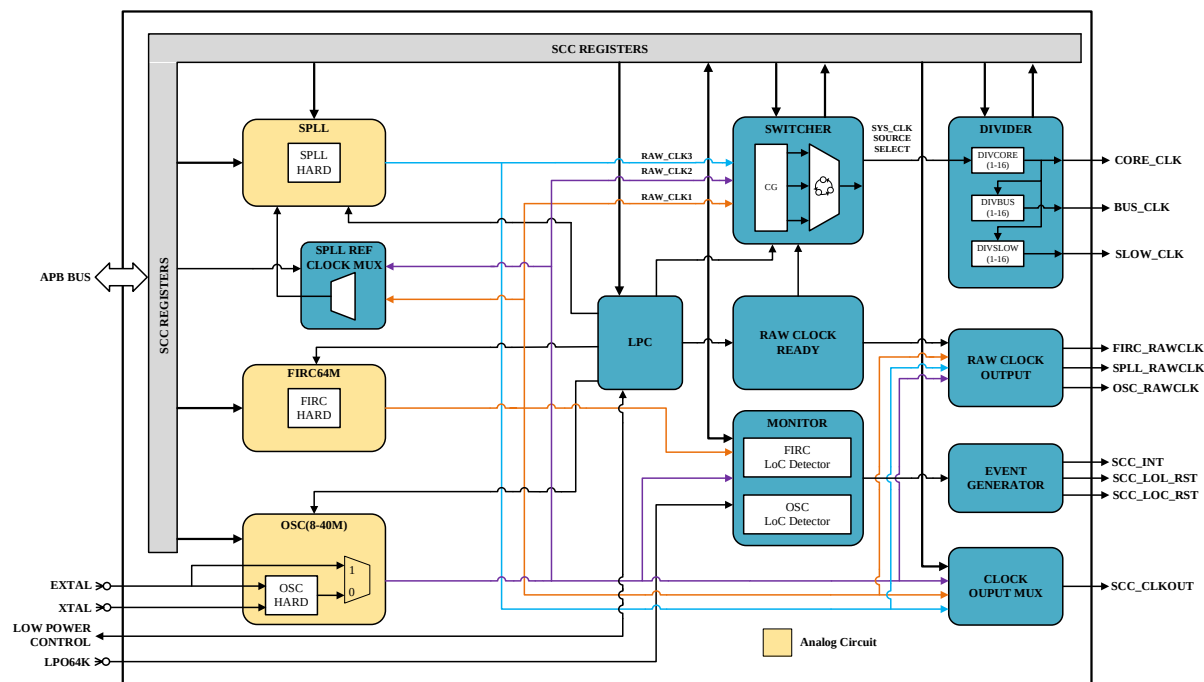


Figure 23-1 SCC block diagram

23.3 Memory map and register definition

This section provides the memory map and register definitions of the SCC module. The SCC registers can be written only in supervisor mode.

23.3.1 Memory map

SCC base address: 0x40064000

Table 23-1 SCC memory map

Offset	Register	Width (In bits)	Reset Value
0x4	System Clock Control Parameter Register (SCC_PARAM)	32	0x70000007
0x8	System Clock Control Clock Status Register (SCC_CST)	32	0x00010113
0xC	System Clock Control Clock Configuration Register (SCC_CFG)	32	0x80010113
0x10	System Clock Control Clock Out Configuration Register (SCC_CLKOUTCFG)	32	0x00000000
0x14	System Clock Control OSC Control Status Register (SCC_OSCCS)	32	0x80000000
0x18	System Clock Control OSC Configuration Register (SCC_OSCCFG)	32	0x80000105
0x1C	System Clock Control FIRC Control Status Register (SCC_FIRCCS)	32	0x80030001

Offset	Register	Width (In bits)	Reset Value
0x28	System Clock Control SPLLC Control Status Register (SCC_SPLLCSTATUS)	32	0x80000000
0x2C	System Clock Control SPLLC Configure1 Register (SCC_SPLLCFG1)	32	0x80030800
0x30	System Clock Control SPLLC Configure2 Register (SCC_SPLLCFG2)	32	0x80000305

23.3.2 System Clock Control Parameter (SCC_PARAM)

23.3.2.1 Offset

Register	Offset
SCC_PARAM	0x4

23.3.2.2 Function

This register is used to indicate design feature implementation.

23.3.2.3 Fields

Field	Access	Description	Reset Value
4:0 CLKSPRES	RO	Clock Source Present. CLKSPRES[4] reserved. CLKSPRES[3] reserved. CLKSPRES[2]=1 SPLLC is present. CLKSPRES[1]=1 OSC40M is present. CLKSPRES[0]=1 FIRC64M is present.	0x7
27:5 RSVD_27_5	RO	Reserved.	0x0
30:28 DIVPRES	RO	Divider Present. DIVPRES[30]=1 CORE divider is present. DIVPRES[29]=1 BUS divider is present. DIVPRES[28]=1 SLOW divider is present (slow clock is the same with flash clock).	0x7
31 RSVD_31	RO	Reserved.	0x0

23.3.3 System Clock Control Clock Status Register (SCC_CST)

23.3.3.1 Offset

Register	Offset
SCC_CST	0x8

23.3.3.2 Function

This register indicates the currently configured system clock source and the system clock dividers for the core (DIVCORE) and peripheral clock (DIVBUS, DIVSLOW). The

SCC_CST register reflects the configuration set.

23.3.3.3 Fields

Field	Access	Description	Reset Value
3:0 DIVSLOW	RO	Slow Clock Divide Ratio 0000b: Divide-by-1 0001b: Divide-by-2 0010b: Divide-by-3 0011b: Divide-by-4 0100b: Divide-by-5 0101b: Divide-by-6 0110b: Divide-by-7 0111b: Divide-by-8 1000b: Divide-by-9 1001b: Divide-by-10 1010b: Divide-by-11 1011b: Divide-by-12 1100b: Divide-by-13 1101b: Divide-by-14 1110b: Divide-by-15 1111b: Divide-by-16	0x3
7:4 DIVBUS	RO	Bus Clock Divide Ratio 0000b: Divide-by-1 0001b: Divide-by-2 0010b: Divide-by-3 0011b: Divide-by-4 0100b: Divide-by-5 0101b: Divide-by-6 0110b: Divide-by-7 0111b: Divide-by-8 1000b: Divide-by-9 1001b: Divide-by-10 1010b: Divide-by-11 1011b: Divide-by-12 1100b: Divide-by-13 1101b: Divide-by-14 1110b: Divide-by-15 1111b: Divide-by-16	0x1
11:8 DIVCORE	RO	Core Clock Divide Ratio 0000b: Divide-by-1 0001b: Divide-by-2 0010b: Divide-by-3 0011b: Divide-by-4 0100b: Divide-by-5 0101b: Divide-by-6 0110b: Divide-by-7 0111b: Divide-by-8 1000b: Divide-by-9 1001b: Divide-by-10 1010b: Divide-by-11 1011b: Divide-by-12 1100b: Divide-by-13 1101b: Divide-by-14 1110b: Divide-by-15 1111b: Divide-by-16	0x1
15:12	RO	Reserved	0x0

Field	Access	Description	Reset Value
RSVD_15_12			
18:16 SCS	RO	System Clock Source 000b: Reserved 001b: FIRC64M 010b: OSC40M 011b: SPLL 100b: Reserved 101b: Reserved 110b: Reserved 111b: Reserved	0x1
31:19 RSVD_31_19	RO	Reserved	0x0

23.3.4 System Clock Control Clock Configuration Register (SCC_CFG)

23.3.4.1 Offset

Register	Offset
SCC_CFG	0xC

23.3.4.2 Function

This register controls the system clock source and the system clock dividers for the core, platform, flash, external and bus clock domains. The user can only write this register with a 32-bit write. The clock source should be enabled first and be valid before system clocks switch to it. If the system clock divide ratios change when selecting a different clock mode, new divide ratios will not take effect until the new clock source is valid.

23.3.4.3 Fields

Field	Access	Description	Reset Value
3:0 DIVSLOW	RW	Slow Clock Divide Ratio. 0000b: Divide-by-1 0001b: Divide-by-2 0010b: Divide-by-3 0011b: Divide-by-4 0100b: Divide-by-5 0101b: Divide-by-6 0110b: Divide-by-7 0111b: Divide-by-8 1000b: Divide-by-9 1001b: Divide-by-10 1010b: Divide-by-11 1011b: Divide-by-12 1100b: Divide-by-13 1101b: Divide-by-14 1110b: Divide-by-15 1111b: Divide-by-16	0x3
7:4 DIVBUS	RW	Bus Clock Divide Ratio. 0000b: Divide-by-1 0001b: Divide-by-2 0010b: Divide-by-3	0x1

Field	Access	Description	Reset Value
		0011b: Divide-by-4 0100b: Divide-by-5 0101b: Divide-by-6 0110b: Divide-by-7 0111b: Divide-by-8 1000b: Divide-by-9 1001b: Divide-by-10 1010b: Divide-by-11 1011b: Divide-by-12 1100b: Divide-by-13 1101b: Divide-by-14 1110b: Divide-by-15 1111b: Divide-by-16	
11:8 DIVCORE	RW	Core Clock Divide Ratio. 0000b: Divide-by-1 0001b: Divide-by-2 0010b: Divide-by-3 0011b: Divide-by-4 0100b: Divide-by-5 0101b: Divide-by-6 0110b: Divide-by-7 0111b: Divide-by-8 1000b: Divide-by-9 1001b: Divide-by-10 1010b: Divide-by-11 1011b: Divide-by-12 1100b: Divide-by-13 1101b: Divide-by-14 1110b: Divide-by-15 1111b: Divide-by-16	0x1
15:12 RSVD_15_12	RO	Reserved	0x0
18:16 SCS	RW	System Clock Source. Default is FIRC. System clock should be set as OSC or SPLL clock source by software once reset is finished. Note: Before switching the system clock, it is necessary to modify the FLASH_FCTRL[RWSC] in advance according to the explanation of the FLASH_FCTRL[RWSC] in the flash module to ensure that the RWSC complies with the requirements of the AHB clock. 000b: Reserved 001b: FIRC64M 010b: OSC40M 011b: SPLL 100b: Reserved 101b: Reserved 110b: Reserved 111b: Reserved	0x1
30:19 RSVD_30_19	RO	Reserved	0x0
31 LOCK	RW	Lock register access. This bit can be set at any time. The rest of bits in this register would be read-only once LOCK bit is set. Write 0x5B to bit[31:24] to clear it.	0x1

23.3.5 System Clock Control Clock Out Configuration Register (SCC_CLKOUTCFG)

23.3.5.1 Offset

Register	Offset
SCC_CLKOUTCFG	0x10

23.3.5.2 Function

This register controls which SCC clock source is selected to be ported out to the CLKOUT pin.

23.3.5.3 Fields

Field	Access	Description	Reset Value
15:0 RSVD_15_0	RO	Reserved.	0x0
19:16 CLKOUTSEL	RW	SCC CLKOUT Configuration. 0000b: Disabled 0001b: FIRC64M 0010b: OSC40M 0011b: SPLL 0100b: Reserved 0101b: Reserved 0110b: Reserved 0111b: Reserved 1000b: SCC Slow Clock 1001b: SCC Bus Clock 1010b: SCC Core Clock	0x0
31:20 RSVD_31_20	RO	Reserved	0x0

23.3.6 System Clock Control OSC Control Status Register (SCC_OSCCS)

23.3.6.1 Offset

Register	Offset
SCC_OSCCS	0x14

23.3.6.2 Function

This register is used to control the OSC and indicate the status of OSC.

23.3.6.3 Fields

Field	Access	Description	Reset Value
0 OSCEN	RW	OSC Enable. 0b: Disable. 1b: Enable.	0x0

Field	Access	Description	Reset Value
1 OSCSTOPEN	RW	OSC STOP Mode Enable. 0b: OSC is disabled in STOP mode. 1b: OSC is enabled in STOP mode.	0x0
3:2 RSVD_3_2	RO	Reserved	0x0
4 OSCCLOCINTEN	RW	OSC Loss of Clock Interrupt Enable. 0b: Disable. 1b: Enable.	0x0
8:5 RSVD_8_5	RO	Reserved	0x0
9 OSCCME	RW	OSC Clock Monitor Enable. 0b: Disable. 1b: Enable. Note: If OSCCME is to be enabled, the FIRC clock must be kept valid.	0x0
10 OSCCMRE	RW	OSC Clock Monitor selects action when loss of clock is detected. 0b: Generated interrupt if loss of clock is detected. 1b: Generated reset if loss of clock is detected.	0x0
15:11 RSVD_15_11	RO	Reserved	0x0
16 OSCRDY	RO	OSC is ready for use. 0b: OSC is not enabled, or clock is not ready. 1b: OSC is enabled, and clock is ready.	0x0
17 OSCSELD	RO	OSC is selected as system clock source. 0b: OSC is not the system clock source. 1b: OSC is the system clock source.	0x0
18 OSCLOC	W1C	OSC Loss of Clock. This flag is reset on chip POR only. It can also be cleared by software writing a logic one. 0b: OSC clock monitor is disabled or has no loss of clock detection. 1b: OSC clock monitor is enabled and detected loss of clock.	0x0
30:19 RSVD_30_19	RO	Reserved	0x0
31 LOCK	RW	Lock register access. This bit can be set at any time. The rest of bits in this register would be read-only once LOCK bit is set. Write 0x5B to bit[31:24] to clear it.	0x1

23.3.7 System Clock Control OSC Configuration Register (SCC_OSCCFG)

23.3.7.1 Offset

Register	Offset
SCC_OSCCFG	0x18

23.3.7.2 Function

This register is used to configure OSC.

23.3.7.3 Fields

Field	Access	Description	Reset Value
0 EXCLKS	RW	OSC External Clock Select. 0b: Crystal: OSC clock comes from crystal oscillator. 1b: External source: OSC clock comes from an external square wave clock source.	0x1
1 OLMEN	RW	Open Loop Mode. It must be set to 1.	0x0
2 HFREQ	RW	High Frequency Mode 0b: Low Frequency Mode. 1b: High Frequency Mode (While using high frequency crystal). NOTE: It is recommended to set this bit when high frequency crystal (typically $\geq 24\text{MHz}$) is selected. Due to deviation of crystal, it is suggested to configure this bit after verification.	0x1
7:3 RSVD_7_3	RO	Reserved	0x0
9:8 ITRIM	RW	Driving Current Select The driver's drive strength can be fine-tuned by ITRIM while it can be coarse-tuned by HFREQ. It is necessary to combine them according to the following recommended configuration. SCC_OSCCFG[HFREQ]=0: 00b: Frequency=8MHz 01b: $8\text{MHz} < \text{Frequency} \leq 16\text{MHz}$ 10b: $16\text{M} < \text{Frequency} \leq 24\text{M}$ SCC_OSCCFG[HFREQ]=1: 01b: $24\text{M} < \text{Frequency} \leq 40\text{M}$ NOTE: Due to deviation of crystal, it is suggested to configure ITRIM after verification.	0x1
30:10 RSVD_30_10	RO	Reserved	0x0
31 LOCK	RW	Lock register access	0x1

23.3.8 System Clock Control FIRC Control Status Register (SCC_FIRCCS)

23.3.8.1 Offset

Register	Offset
SCC_FIRCCS	0x1C

23.3.8.2 Function

This register is used to control the FIRC and indicate the status of FIRC.

23.3.8.3 Fields

Field	Access	Description	Reset Value
0	RW	FIRC Enable.	0x1

Field	Access	Description	Reset Value
FIRCEN		0b: Disable. 1b: Enable.	
1 FIRCSTOPEN	RW	FIRC STOP Mode Enable. 0b: FIRC64M is disabled in STOP mode. 1b: FIRC64M is enabled in STOP mode.	0x0
3:2 RSVD_3_2	RO	Reserved	0x0
4 FIRCLOCINTEN	RW	FIRC Loss of Clock Interrupt Enable. 0b: Disable. 1b: Enable.	0x0
8:5 RSVD_8_5	RO	Reserved	0x0
9 FIRCCME	RW	FIRC Clock Monitor Enable. 0b: Disable. 1b: Enable.	0x0
10 FIRCCMRE	RW	FIRC Clock Monitor selects action when loss of clock is detected. 0b: Generated interrupt if loss of clock is detected. 1b: Generated reset if loss of clock is detected.	0x0
15:11 RSVD_15_11	RO	Reserved	0x0
16 FIRCRDY	RO	FIRC64M is ready for use. 0b: FIRC64M is not enabled or clock is not ready. 1b: FIRC64M is enabled and clock is ready.	0x1
17 FIRCSELD	RO	FIRC64M is selected as system clock source. 0b: FIRC64M is not the system clock source. 1b: FIRC64M is the system clock source.	0x1
18 FIRCLOC	W1C	FIRC Loss of Clock. This flag is reset on chip POR only. It can also be cleared by software writing a logic one. 0b: FIRC clock monitor is disabled or has no loss of clock detection. 1b: FIRC clock monitor is enabled and detected loss of clock.	0x0
30:19 RSVD_30_19	RO	Reserved	0x0
31 LOCK	RW	Lock register access. This bit can be set at any time. The rest of bits in this register would be read-only once LOCK bit is set. Write 0x5B to bit[31:24] to clear it.	0x1

23.3.9 System Clock Control SPLL Control Status Register (SCC_SPLLCS)

23.3.9.1 Offset

Register	Offset
SCC_SPLLCS	0x28

23.3.9.2 Function

This register is used to control or indicate the SPLL status. Users must enable the SPLL ready bit detector (RDYEN) before enabling the SPLL.

23.3.9.3 Fields

Field	Access	Description	Reset Value
0 SPLLEN	RW	SPLL Enable 0b: Disable 1b: Enable	0x0
1 SPLLSTOPEN	RW	SPLL STOP Mode Enable 0b: SPLL is disabled in STOP mode. 1b: SPLL is enabled in STOP mode.	0x0
8:2 RSVD_8_2	RO	Reserved	0x0
9 RDYEN	RW	SPLL ready bit detector enable. 0b: Disable 1b: Enable	0x0
11:10 RSVD_11_10	RO	Reserved	0x0
13:12 LOCKWIN	RW	SPLL phase lock window It is recommended to set to 0x3.	0x0
15:14 RSVD_15_14	RO	Reserved	0x0
16 SPLLRDY	RO	SPLL is ready for use. 0b: SPLL is not enabled or clock is not ready. 1b: SPLL is enabled and clock is ready.	0x0
17 SPLLSELD	RO	SPLL is selected as system clock source. 0b: SPLL is not the system clock source. 1b: SPLL is the system clock source.	0x0
24:18 RSVD_24_18	RO	Reserved	0x0
25 OUTEN	RW	SPLL output enable. 0b: Disable 1b: Enable	0x0
30:26 RSVD_30_26	RO	Reserved	0x0
31 LOCK	RW	Lock register access This bit can be set at any time. The rest of bits in this register would be read-only once LOCK bit is set. Write 0x5B to bit[31:24] to clear it.	0x1

23.3.10 System Clock Control SPLL Configuration Register 1 (SCC_SPLLCFG1)

23.3.10.1 Offset

Register	Offset
SCC_SPLLCFG1	0x2C

23.3.10.2 Function

This register is used to configure the SPLL.

23.3.10.3 Fields

Field	Access	Description	Reset Value

Field	Access	Description	Reset Value
1:0 PREDIV	RW	Pre-divider for reference clock 00b: Divide-by-1. 01b: Divide-by-2. 10b: Divide-by-4.	0x0
3:2 RSVD_3_2	RO	Reserved	0x0
4 FBPRESEN	RW	Enable prescaler for VCO feedback clock 0b: Prescaler is disabled. Actual prescaler is 1. 1b: Prescaler is enabled. Actual prescaler is 2.	0x0
5 REFCKS	RW	Reference clock select 0b: FIRC is selected as reference clock. 1b: OSC is selected as reference clock.	0x0
7:6 RSVD_7_6	RW	Reserved	0x0
15:8 MULT	RW	The MULT bits control the multiplication factor applied to the reference clock frequency. 8'd0 - 8'd7: reserved 8'd8: multiply factor = 8 8'd9: multiply factor = 9 ⋮ 8'd255: multiply factor = 255	0x8
18:16 POSTDIV	RW	Post divider 000b: Reserved 001b: Reserved 010b: Reserved 011b: Divide-by-6 100b: Divide-by-8 101b: Divide-by-10 110b: Divide-by-12 111b: Divide-by-14	0x3
30:19 RSVD_30_19	RO	Reserved	0x0
31 LOCK	RW	Lock register access. This bit can be set at any time. The rest of bits in this register would be read-only once LOCK bit is set. Write 0x5B to bit[31:24] to clear it.	0x1

23.3.11 System Clock Control SPLL Configuration Register 2 (SCC_SPLLCFG2)

23.3.11.1 Offset

Register	Offset
SCC_SPLLCFG2	0x30

23.3.11.2 Function

This register is used to configure the SPLL.

23.3.11.3 Fields

Field	Access	Description	Reset Value

Field	Access	Description	Reset Value
1:0 CPIR	RW	charge pump proportional path current setting	0x1
3:2 CPIC	RW	charge pump integral path current setting	0x1
5:4 CS	RW	LPF cap 2nd pole setting SPLL loop parameter cap setting	0x0
7:6 CAP	RW	LPF cap setting SPLL loop parameter cap setting	0x0
10:8 RES	RW	LPF resistor setting SPLL loop parameter res setting	0x3
11 KVCO	RW	KVCO gain setting 0b: Disable. 1b: Enable.	0x0
12 VCODIV	RW	SPLL VCO/2 option 0b: Disable. 1b: Enable.	0x0
13 PFDDIEN	RW	SPLL PFD dither enable: 0b: Disable. 1b: Enable.	0x0
14 CPISEL	RW	cp bias current selection: 0b: Bias from ICO current, self-bias. 1b: Bias from SPLL bias circuit.	0x0
23:15 RSVD_23_15	RO	Reserved	0x0
24 LDOODEN	RW	LDO over drive enable This bit enables LDO over drive if SPLL VCO needs output high frequency. This bit must be set to 1 to guarantee the driving capability.	0x0
30:25 RSVD_30_25	RO	Reserved	0x0
31 LOCK	RW	Lock register access This bit can be set at any time. The rest of bits in this register would be read-only once LOCK bit is set. Write 0x5B to bit[31:24] to clear it.	0x1

23.4 Function description

This section provides the functional information of the SCC module.

23.4.1 FIRC clock mode

System boots from this clock mode. FIRC clock mode is entered when all of below conditions occur:

- 001b is written to SCC_CFG[SCS].
- SCC_FIRCCS[FIRCEN] = 1b.
- SCC_FIRCCS[FIRCRDY]=1b.
- SCC_FIRCCS[FIRCSELD]=1b.

System clocks are derived from the FIRC in this mode.

23.4.2 OSC clock mode

OSC clock mode is entered when all of below conditions occur:

- 010b is written to SCC_CFG[SCS].

- SCC_OS CCS[OSCEN]=1b.
- SCC_OS CCS[OSCRDY]=1b.
- SCC_OS CCS[OSCSELD]=1b.

System clocks are derived from the OSC in this mode.

23.4.3 SPLL clock mode

SPLL clock mode is entered when all of below conditions occur:

- 011b is written to SCC_CFG[SCS].
- SCC_SPLLCS[SPLLEN]=1b.
- SCC_SPLLCS[RDYEN]=1b.
- SCC_SPLLCS[SPLL RDY]=1b.
- SCC_SPLLCS[SPLLSELD]=1b.
- SCC_SPLLCS[RDYEN]=0b.

System clocks are derived from the SPLL in this mode.

NOTE

Please set RDYEN bit when the user wants to use SPLL clock, and clear this bit when SPLL clock is ready as above.

23.4.4 SPLL frequency

The SPLL output frequency can be calculated as follows with the frequencies given in MHz:

$$F_{\text{spll}} = ((\text{Reference clock frequency} / \text{preDiv}) * \text{multi} * \text{preScaler}) / \text{postDiv}$$

The parameter selection must meet the following requirements. After divided by preDiv, the result must range from 8-16MHz. After multiplied by the multiplier circuits, the result must range from 0.8-1.2GHz.

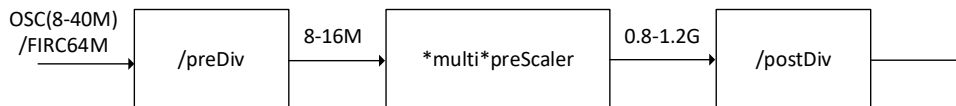


Figure 23-2 SPLL parameter constraint condition

23.4.5 System clock source

The default configuration out of reset has the CPU clocked by the Fast IRC (FIRC_CLK). The following two figures show the selection of system clock and the system clock switching mechanism respectively.

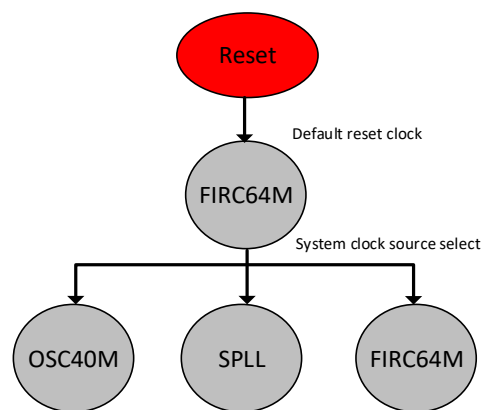


Figure 23-3 System clock selection diagram

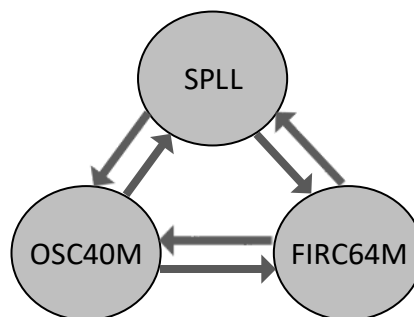


Figure 23-4 System clock switching mechanism diagram

Chapter 24

Peripheral Access & Reset & Clock Controller (PARCC)

24.1 Introduction

This chapter defines the Peripheral Access & Reset & Clock Controller (PARCC). The Peripheral Access & Reset & Clock Controller (PARCC) module provides supervisor mode access control, register write protection, module level reset control, and function clock source switch and divide, module level clock behavior control in STOP mode. Every peripheral is allowed to have an independent PARCC belonging to the peripheral itself.

24.1.1 Features

The PARCC supports the following key features:

- Peripheral register write protection.
- Optional peripheral register access protection in supervisor mode.
- Reset peripheral.
- Functional clock source switch.
- Functional clock source divide by specified ratio.
 - Divide ratio could be 1, 2, 3, ..., 15, 16.
- Three kinds of clock mode:
 - Always off mode.
 - Always on mode.
 - Stop off mode.

24.1.2 Block diagram

The following figure is a block diagram of the Peripheral Access & Reset & Clock Controller (PARCC) module.

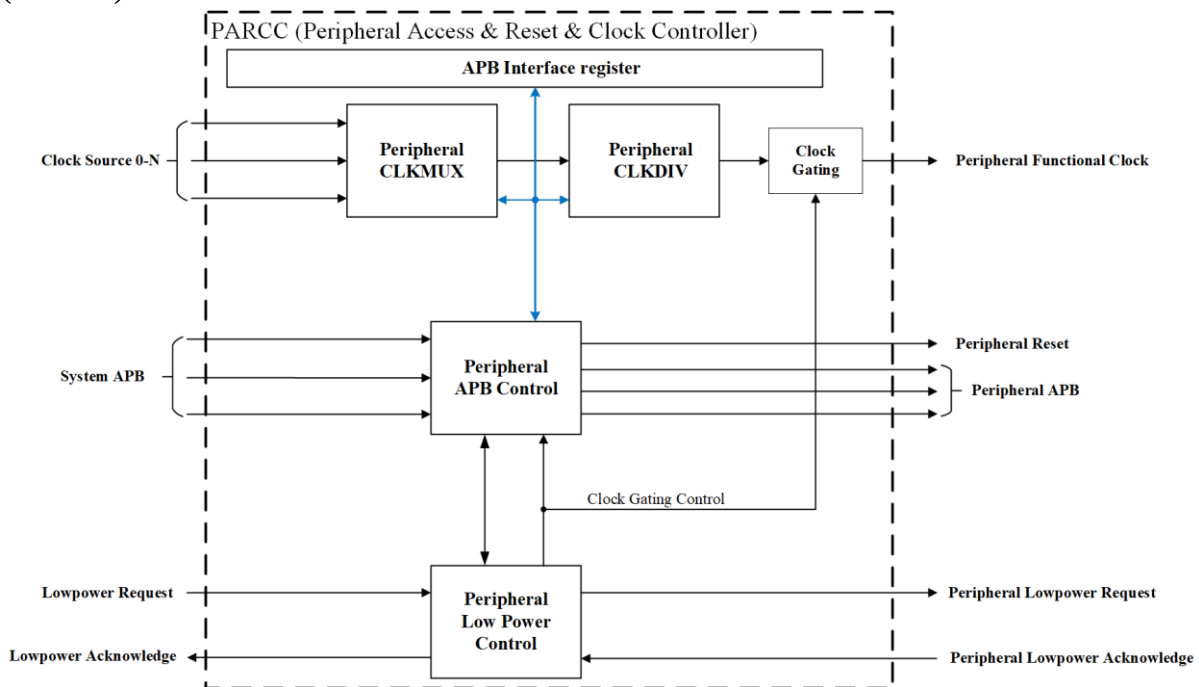


Figure 24-1 PARCC block diagram

24.2 Memory map and register definition

This section provides the memory map and register definitions of the PARCC module. Each peripheral has its own dedicated PARCC register, which controls the supervisor mode write access, software reset, clock mode for that specific module.

The PARCC registers can be written only in supervisor mode.

24.2.1 Memory map

PARCC base address: 0x40065000

Table 24-1 PARCC memory map

Offset	Register	Width (In bits)	Reset Value
0x0	PARCC for WDOG (PARCC_WDOG)	32	0x89200001
0x4	PARCC for EWDG (PARCC_EWDG)	32	0x8F000000
0x8	PARCC for STIM (PARCC_STIM)	32	0x8F000000
0xC	PARCC for TIM0 (PARCC_TIM0)	32	0x8F000000
0x10	PARCC for TIM1 (PARCC_TIM1)	32	0x8F000000
0x14	PARCC for TIM2 (PARCC_TIM2)	32	0x8F000000
0x18	PARCC for TIM3 (PARCC_TIM3)	32	0x8F000000
0x1C	PARCC for MCPWM0 (PARCC_MCPWM0)	32	0x8F000000
0x20	PARCC for MCPWM1 (PARCC_MCPWM1)	32	0x8F000000
0x24	PARCC for TDG0 (PARCC_TDG0)	32	0x8F000000
0x28	PARCC for TDG1 (PARCC_TDG1)	32	0x8F000000
0x2C	PARCC for CAN0 (PARCC_CAN0)	32	0x8F000000
0x30	PARCC for CAN1 (PARCC_CAN1)	32	0x8F000000
0x34	PARCC for CAN2 (PARCC_CAN2)	32	0x8F000000
0x38	PARCC for CAN3 (PARCC_CAN3)	32	0x8F000000
0x3C	PARCC for CAN4 (PARCC_CAN4)	32	0x8F000000
0x40	PARCC for CAN5 (PARCC_CAN5)	32	0x8F000000
0x44	PARCC for CAN6 (PARCC_CAN6)	32	0x8F000000
0x48	PARCC for CAN7 (PARCC_CAN7)	32	0x8F000000
0x4C	PARCC for UART0 (PARCC_UART0)	32	0x8F000000
0x50	PARCC for UART1 (PARCC_UART1)	32	0x8F000000

Offset	Register	Width (In bits)	Reset Value
0x54	PARCC for UART2 (PARCC_UART2)	32	0x8F000000
0x58	PARCC for UART3 (PARCC_UART3)	32	0x8F000000
0x5C	PARCC for UART4 (PARCC_UART4)	32	0x8F000000
0x60	PARCC for UART5 (PARCC_UART5)	32	0x8F000000
0x64	PARCC for SPI0 (PARCC_SPI0)	32	0x8F000000
0x68	PARCC for SPI1 (PARCC_SPI1)	32	0x8F000000
0x6C	PARCC for SPI2 (PARCC_SPI2)	32	0x8F000000
0x70	PARCC for SPI3 (PARCC_SPI3)	32	0x8F000000
0x74	PARCC for I2C0 (PARCC_I2C0)	32	0x8F000000
0x78	PARCC for I2C1 (PARCC_I2C1)	32	0x8F000000
0x7C	PARCC for I2S0 (PARCC_I2S0)	32	0x8F000000
0x80	PARCC for I2S1 (PARCC_I2S1)	32	0x8F000000
0x88	PARCC for ADC0 (PARCC_ADC0)	32	0x8F000000
0x8C	PARCC for ADC1 (PARCC_ADC1)	32	0x8F000000
0x90	PARCC for CMP0 (PARCC_CMP0)	32	0x89200000
0x94	PARCC for CRC (PARCC_CRC)	32	0x89200000
0x98	PARCC for AES (PARCC_AES)	32	0x89200000
0x9C	PARCC for TRNG (PARCC_TRNG)	32	0x89200000
0xA0	PARCC for FLASH (PARCC_FLASH)	32	0x8F200731
0xA8	PARCC for DMA (PARCC_DMA)	32	0x89200000
0xAC	PARCC for DMAMUX (PARCC_DMAMUX)	32	0x89200000
0xB0	PARCC for PORTA (PARCC_PORTA)	32	0x8F000000
0xB4	PARCC for PORTB (PARCC_PORTB)	32	0x8F000000
0xB8	PARCC for PORTC (PARCC_PORTC)	32	0x8F000000
0xBC	PARCC for PORTD (PARCC_PORTD)	32	0x8F000000
0xC0	PARCC for PORTE (PARCC_PORTE)	32	0x8F000000
0xC4	PARCC for TMU (PARCC_TMU)	32	0x89200000

Offset	Register	Width (In bits)	Reset Value
0xC8	PARCC for REGFILE (PARCC_REGFILE)	32	0x89200000
0xCC	PARCC for SMPU (PARCC_SMPU)	32	0x89200001
0xD0	PARCC for GPIO (PARCC_GPIO)	32	0x89200000

24.2.2 PARCC for WDOG (PARCC_WDOG)

24.2.2.1 Offset

Register	Offset
PARCC_WDOG	0x0

24.2.2.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x1
15:2 RSVD_15_2	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17 RSVD_19_17	RO	Reserved	0x0
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	When this bit is cleared, WDOG will not be reset, but it cannot be read and written, any reading and writing will generate hard fault. WDOG registers can only be read and written when this bit is set to 1.	0x1
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x0
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x0

Field	Access	Description	Reset Value
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.2.3 PARCC for EWDT (PARCC_EWDT)

24.2.3.1 Offset

Register	Offset
PARCC_EWDT	0x4

24.2.3.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
3:2 RSVD_3_2	RO	Reserved	0x0
6:4 CLKMUX	RW	Select clock source for peripheral functional clock. 000b: OFF 001b: LPO32K 010b: OSC32K 011b: FIRC64M 100b: OSC40M 101b: Reserved	0x0
7 RSVD_7	RO	Reserved	0x0
11:8 CLKDIV	RW	Select clock divide ratio for peripheral functional clock. 0000b: Clock divided by 1 0001b: Clock divided by 2 0010b: Clock divided by 3 0011b: Clock divided by 4 0100b: Clock divided by 5 0101b: Clock divided by 6 0110b: Clock divided by 7 0111b: Clock divided by 8 1000b: Clock divided by 9 1001b: Clock divided by 10 1010b: Clock divided by 11 1011b: Clock divided by 12	0x0

Field	Access	Description	Reset Value
		1100b: Clock divided by 13 1101b: Clock divided by 14 1110b: Clock divided by 15 1111b: Clock divided by 16	
15:12 RSVD_15_12	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17 RSVD_19_17	RO	Reserved	0x0
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	Reset peripheral when this bit is clear.	0x0
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x1
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x1
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.2.4 PARCC for STIM (PARCC_STIM)

24.2.4.1 Offset

Register	Offset
PARCC_STIM	0x8

24.2.4.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
3:2 RSVD_3_2	RO	Reserved	0x0
6:4 CLKMUX	RW	Select clock source for peripheral functional clock. 000b: OFF 001b: LPO32K 010b: OSC32K 011b: FIRC64M 100b: OSC40M 101b: SPILL	0x0
7 RSVD_7	RO	Reserved	0x0
11:8 CLKDIV	RW	Select clock divide ratio for peripheral functional clock. 0000b: Clock divided by 1 0001b: Clock divided by 2 0010b: Clock divided by 3 0011b: Clock divided by 4 0100b: Clock divided by 5 0101b: Clock divided by 6 0110b: Clock divided by 7 0111b: Clock divided by 8 1000b: Clock divided by 9 1001b: Clock divided by 10 1010b: Clock divided by 11 1011b: Clock divided by 12 1100b: Clock divided by 13 1101b: Clock divided by 14 1110b: Clock divided by 15 1111b: Clock divided by 16	0x0
15:12 RSVD_15_12	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17 RSVD_19_17	RO	Reserved	0x0
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	Reset peripheral when this bit is clear.	0x0
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1

Field	Access	Description	Reset Value
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x1
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x1
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.2.5 PARCC for TIM0 (PARCC_TIM0)

24.2.5.1 Offset

Register	Offset
PARCC_TIM0	0xC

24.2.5.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
3:2 RSVD_3_2	RO	Reserved	0x0
6:4 CLKMUX	RW	Select clock source for peripheral functional clock. 000b: OFF 001b: LPO32K 010b: OSC32K 011b: FIRC64M 100b: OSC40M 101b: SPLL	0x0
7 RSVD_7	RO	Reserved	0x0
11:8 CLKDIV	RW	Select clock divide ratio for peripheral functional clock. 0000b: Clock divided by 1 0001b: Clock divided by 2 0010b: Clock divided by 3 0011b: Clock divided by 4	0x0

Field	Access	Description	Reset Value
		0100b: Clock divided by 5 0101b: Clock divided by 6 0110b: Clock divided by 7 0111b: Clock divided by 8 1000b: Clock divided by 9 1001b: Clock divided by 10 1010b: Clock divided by 11 1011b: Clock divided by 12 1100b: Clock divided by 13 1101b: Clock divided by 14 1110b: Clock divided by 15 1111b: Clock divided by 16	
15:12 RSVD_15_12	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17 RSVD_19_17	RO	Reserved	0x0
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	Reset peripheral when this bit is clear.	0x0
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x1
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x1
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.2.6 PARCC for TIM1 (PARCC_TIM1)

24.2.6.1 Offset

Register	Offset
PARCC_TIM1	0x10

24.2.6.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
3:2 RSVD_3_2	RO	Reserved	0x0
6:4 CLKMUX	RW	Select clock source for peripheral functional clock. 000b: OFF 001b: LPO32K 010b: OSC32K 011b: FIRC64M 100b: OSC40M 101b: SPLL	0x0
7 RSVD_7	RO	Reserved	0x0
11:8 CLKDIV	RW	Select clock divide ratio for peripheral functional clock. 0000b: Clock divided by 1 0001b: Clock divided by 2 0010b: Clock divided by 3 0011b: Clock divided by 4 0100b: Clock divided by 5 0101b: Clock divided by 6 0110b: Clock divided by 7 0111b: Clock divided by 8 1000b: Clock divided by 9 1001b: Clock divided by 10 1010b: Clock divided by 11 1011b: Clock divided by 12 1100b: Clock divided by 13 1101b: Clock divided by 14 1110b: Clock divided by 15 1111b: Clock divided by 16	0x0
15:12 RSVD_15_12	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17 RSVD_19_17	RO	Reserved	0x0
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	Reset peripheral when this bit is clear.	0x0

Field	Access	Description	Reset Value
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x1
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x1
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.2.7 PARCC for TIM2 (PARCC_TIM2)

24.2.7.1 Offset

Register	Offset
PARCC_TIM2	0x14

24.2.7.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
3:2 RSVD_3_2	RO	Reserved	0x0
6:4 CLKMUX	RW	Select clock source for peripheral functional clock. 000b: OFF 001b: LPO32K 010b: OSC32K 011b: FIRC64M 100b: OSC40M 101b: SPL	0x0
7 RSVD_7	RO	Reserved	0x0

Field	Access	Description	Reset Value
11:8 CLKDIV	RW	Select clock divide ratio for peripheral functional clock. 0000b: Clock divided by 1 0001b: Clock divided by 2 0010b: Clock divided by 3 0011b: Clock divided by 4 0100b: Clock divided by 5 0101b: Clock divided by 6 0110b: Clock divided by 7 0111b: Clock divided by 8 1000b: Clock divided by 9 1001b: Clock divided by 10 1010b: Clock divided by 11 1011b: Clock divided by 12 1100b: Clock divided by 13 1101b: Clock divided by 14 1110b: Clock divided by 15 1111b: Clock divided by 16	0x0
15:12 RSVD_15_12	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17 RSVD_19_17	RO	Reserved	0x0
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	Reset peripheral when this bit is clear.	0x0
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x1
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x1
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write	0x1

Field	Access	Description	Reset Value
		lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	

24.2.8 PARCC for TIM3 (PARCC_TIM3)

24.2.8.1 Offset

Register	Offset
PARCC_TIM3	0x18

24.2.8.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
3:2 RSVD_3_2	RO	Reserved	0x0
6:4 CLKMUX	RW	Select clock source for peripheral functional clock. 000b: OFF 001b: LPO32K 010b: OSC32K 011b: FIRC64M 100b: OSC40M 101b: SPL	0x0
7 RSVD_7	RO	Reserved	0x0
11:8 CLKDIV	RW	Select clock divide ratio for peripheral functional clock. 0000b: Clock divided by 1 0001b: Clock divided by 2 0010b: Clock divided by 3 0011b: Clock divided by 4 0100b: Clock divided by 5 0101b: Clock divided by 6 0110b: Clock divided by 7 0111b: Clock divided by 8 1000b: Clock divided by 9 1001b: Clock divided by 10 1010b: Clock divided by 11 1011b: Clock divided by 12 1100b: Clock divided by 13 1101b: Clock divided by 14 1110b: Clock divided by 15 1111b: Clock divided by 16	0x0
15:12 RSVD_15_12	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17	RO	Reserved	0x0

Field	Access	Description	Reset Value
RSVD_19_17			
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	Reset peripheral when this bit is clear.	0x0
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x1
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x1
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.2.9 PARCC for MCPWM0 (PARCC_MCPWM0)

24.2.9.1 Offset

Register	Offset
PARCC_MCPWM0	0x1C

24.2.9.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
3:2 RSVD_3_2	RO	Reserved	0x0

Field	Access	Description	Reset Value
6:4 CLKMUX	RW	Select clock source for peripheral functional clock. 000b: OFF 001b: LPO32K 010b: OSC32K 011b: FIRC64M 100b: OSC40M 101b: SPLL	0x0
7 RSVD_7	RO	Reserved	0x0
11:8 CLKDIV	RW	Select clock divide ratio for peripheral functional clock. 0000b: Clock divided by 1 0001b: Clock divided by 2 0010b: Clock divided by 3 0011b: Clock divided by 4 0100b: Clock divided by 5 0101b: Clock divided by 6 0110b: Clock divided by 7 0111b: Clock divided by 8 1000b: Clock divided by 9 1001b: Clock divided by 10 1010b: Clock divided by 11 1011b: Clock divided by 12 1100b: Clock divided by 13 1101b: Clock divided by 14 1110b: Clock divided by 15 1111b: Clock divided by 16	0x0
15:12 RSVD_15_12	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17 RSVD_19_17	RO	Reserved	0x0
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	Reset peripheral when this bit is clear.	0x0
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x1
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x1
27	RO	Peripheral supervisor access control present	0x1

Field	Access	Description	Reset Value
PSUPVACPR		Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.2.10 PARCC for MCPWM1(PARCC_MCPWM1)

24.2.10.1 Offset

Register	Offset
PARCC_MCPWM1	0x20

24.2.10.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
3:2 RSVD_3_2	RO	Reserved	0x0
6:4 CLKMUX	RW	Select clock source for peripheral functional clock. 000b: OFF 001b: LPO32K 010b: OSC32K 011b: FIRC64M 100b: OSC40M 101b: SPLL	0x0
7 RSVD_7	RO	Reserved	0x0
11:8 CLKDIV	RW	Select clock divide ratio for peripheral functional clock. 0000b: Clock divided by 1 0001b: Clock divided by 2 0010b: Clock divided by 3 0011b: Clock divided by 4 0100b: Clock divided by 5 0101b: Clock divided by 6 0110b: Clock divided by 7 0111b: Clock divided by 8 1000b: Clock divided by 9 1001b: Clock divided by 10 1010b: Clock divided by 11 1011b: Clock divided by 12 1100b: Clock divided by 13	0x0

Field	Access	Description	Reset Value
		1101b: Clock divided by 14 1110b: Clock divided by 15 1111b: Clock divided by 16	
15:12 RSVD_15_12	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17 RSVD_19_17	RO	Reserved	0x0
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	Reset peripheral when this bit is clear.	0x0
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x1
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x1
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.2.11 PARCC for TDG0 (PARCC_TDG0)

24.2.11.1 Offset

Register	Offset
PARCC_TDG0	0x24

24.2.11.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
3:2 RSVD_3_2	RO	Reserved	0x0
6:4 CLKMUX	RW	Select clock source for peripheral functional clock. 000b: OFF 001b: LPO32K 010b: OSC32K 011b: FIRC64M 100b: OSC40M 101b: SPL	0x0
7 RSVD_7	RO	Reserved	0x0
11:8 CLKDIV	RW	Select clock divide ratio for peripheral functional clock. 0000b: Clock divided by 1 0001b: Clock divided by 2 0010b: Clock divided by 3 0011b: Clock divided by 4 0100b: Clock divided by 5 0101b: Clock divided by 6 0110b: Clock divided by 7 0111b: Clock divided by 8 1000b: Clock divided by 9 1001b: Clock divided by 10 1010b: Clock divided by 11 1011b: Clock divided by 12 1100b: Clock divided by 13 1101b: Clock divided by 14 1110b: Clock divided by 15 1111b: Clock divided by 16	0x0
15:12 RSVD_15_12	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17 RSVD_19_17	RO	Reserved	0x0
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	Reset peripheral when this bit is clear.	0x0
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1

Field	Access	Description	Reset Value
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x1
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x1
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.2.12 PARCC for TDG1 (PARCC_TDG1)

24.2.12.1 Offset

Register	Offset
PARCC_TDG1	0x28

24.2.12.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
3:2 RSVD_3_2	RO	Reserved	0x0
6:4 CLKMUX	RW	Select clock source for peripheral functional clock. 000b: OFF 001b: LPO32K 010b: OSC32K 011b: FIRC64M 100b: OSC40M 101b: SPL	0x0
7 RSVD_7	RO	Reserved	0x0
11:8 CLKDIV	RW	Select clock divide ratio for peripheral functional clock. 0000b: Clock divided by 1 0001b: Clock divided by 2 0010b: Clock divided by 3 0011b: Clock divided by 4	0x0

Field	Access	Description	Reset Value
		0100b: Clock divided by 5 0101b: Clock divided by 6 0110b: Clock divided by 7 0111b: Clock divided by 8 1000b: Clock divided by 9 1001b: Clock divided by 10 1010b: Clock divided by 11 1011b: Clock divided by 12 1100b: Clock divided by 13 1101b: Clock divided by 14 1110b: Clock divided by 15 1111b: Clock divided by 16	
15:12 RSVD_15_12	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17 RSVD_19_17	RO	Reserved	0x0
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	Reset peripheral when this bit is clear.	0x0
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x1
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x1
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.2.13 PARCC for CAN0 (PARCC_CAN0)

24.2.13.1 Offset

Register	Offset
PARCC_CAN0	0x2C

24.2.13.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
3:2 RSVD_3_2	RO	Reserved	0x0
6:4 CLKMUX	RW	Select clock source for peripheral functional clock. 000b: OFF 001b: Reserved 010b: Reserved 011b: FIRC64M 100b: OSC40M 101b: SPL	0x0
7 RSVD_7	RO	Reserved	0x0
11:8 CLKDIV	RW	Select clock divide ratio for peripheral functional clock. 0000b: Clock divided by 1 0001b: Clock divided by 2 0010b: Clock divided by 3 0011b: Clock divided by 4 0100b: Clock divided by 5 0101b: Clock divided by 6 0110b: Clock divided by 7 0111b: Clock divided by 8 1000b: Clock divided by 9 1001b: Clock divided by 10 1010b: Clock divided by 11 1011b: Clock divided by 12 1100b: Clock divided by 13 1101b: Clock divided by 14 1110b: Clock divided by 15 1111b: Clock divided by 16	0x0
15:12 RSVD_15_12	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17 RSVD_19_17	RO	Reserved	0x0
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	When CAN_MCR[MDIS] is set to 1, CAN0 will be reset when this bit is cleared.	0x0

Field	Access	Description	Reset Value
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x1
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x1
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.2.14 PARCC for CAN1 (PARCC_CAN1)

24.2.14.1 Offset

Register	Offset
PARCC_CAN1	0x30

24.2.14.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
3:2 RSVD_3_2	RO	Reserved	0x0
6:4 CLKMUX	RW	Select clock source for peripheral functional clock. 000b: OFF 001b: Reserved 010b: Reserved 011b: FIRC64M 100b: OSC40M 101b: SPL	0x0
7 RSVD_7	RO	Reserved	0x0

Field	Access	Description	Reset Value
11:8 CLKDIV	RW	Select clock divide ratio for peripheral functional clock. 0000b: Clock divided by 1 0001b: Clock divided by 2 0010b: Clock divided by 3 0011b: Clock divided by 4 0100b: Clock divided by 5 0101b: Clock divided by 6 0110b: Clock divided by 7 0111b: Clock divided by 8 1000b: Clock divided by 9 1001b: Clock divided by 10 1010b: Clock divided by 11 1011b: Clock divided by 12 1100b: Clock divided by 13 1101b: Clock divided by 14 1110b: Clock divided by 15 1111b: Clock divided by 16	0x0
15:12 RSVD_15_12	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17 RSVD_19_17	RO	Reserved	0x0
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	When CAN_MCR[MDIS] is set to 1, CAN1 will be reset when this bit is cleared.	0x0
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x1
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x1
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write	0x1

Field	Access	Description	Reset Value
		lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	

24.2.15 PARCC for CAN2 (PARCC_CAN2)

24.2.15.1 Offset

Register	Offset
PARCC_CAN2	0x34

24.2.15.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
3:2 RSVD_3_2	RO	Reserved	0x0
6:4 CLKMUX	RW	Select clock source for peripheral functional clock. 000b: OFF 001b: Reserved 010b: Reserved 011b: FIRC64M 100b: OSC40M 101b: SPL	0x0
7 RSVD_7	RO	Reserved	0x0
11:8 CLKDIV	RW	Select clock divide ratio for peripheral functional clock. 0000b: Clock divided by 1 0001b: Clock divided by 2 0010b: Clock divided by 3 0011b: Clock divided by 4 0100b: Clock divided by 5 0101b: Clock divided by 6 0110b: Clock divided by 7 0111b: Clock divided by 8 1000b: Clock divided by 9 1001b: Clock divided by 10 1010b: Clock divided by 11 1011b: Clock divided by 12 1100b: Clock divided by 13 1101b: Clock divided by 14 1110b: Clock divided by 15 1111b: Clock divided by 16	0x0
15:12 RSVD_15_12	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17	RO	Reserved	0x0

Field	Access	Description	Reset Value
RSVD_19_17			
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	When CAN_MCR[MDIS] is set to 1, CAN2 will be reset when this bit is cleared.	0x0
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x1
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x1
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.2.16 PARCC for CAN3 (PARCC_CAN3)

24.2.16.1 Offset

Register	Offset
PARCC_CAN3	0x38

24.2.16.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
3:2 RSVD_3_2	RO	Reserved	0x0

Field	Access	Description	Reset Value
6:4 CLKMUX	RW	Select clock source for peripheral functional clock. 000b: OFF 001b: Reserved 010b: Reserved 011b: FIRC64M 100b: OSC40M 101b: SPLL	0x0
7 RSVD_7	RO	Reserved	0x0
11:8 CLKDIV	RW	Select clock divide ratio for peripheral functional clock. 0000b: Clock divided by 1 0001b: Clock divided by 2 0010b: Clock divided by 3 0011b: Clock divided by 4 0100b: Clock divided by 5 0101b: Clock divided by 6 0110b: Clock divided by 7 0111b: Clock divided by 8 1000b: Clock divided by 9 1001b: Clock divided by 10 1010b: Clock divided by 11 1011b: Clock divided by 12 1100b: Clock divided by 13 1101b: Clock divided by 14 1110b: Clock divided by 15 1111b: Clock divided by 16	0x0
15:12 RSVD_15_12	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17 RSVD_19_17	RO	Reserved	0x0
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	When CAN_MCR[MDIS] is set to 1, CAN3 will be reset when this bit is cleared.	0x0
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x1
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x1
27	RO	Peripheral supervisor access control present	0x1

Field	Access	Description	Reset Value
PSUPVACPR		Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.2.17 PARCC for CAN4 (PARCC_CAN4)

24.2.17.1 Offset

Register	Offset
PARCC_CAN4	0x3C

24.2.17.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
3:2 RSVD_3_2	RO	Reserved	0x0
6:4 CLKMUX	RW	Select clock source for peripheral functional clock. 000b: OFF 001b: Reserved 010b: Reserved 011b: FIRC64M 100b: OSC40M 101b: SPLL	0x0
7 RSVD_7	RO	Reserved	0x0
11:8 CLKDIV	RW	Select clock divide ratio for peripheral functional clock. 0000b: Clock divided by 1 0001b: Clock divided by 2 0010b: Clock divided by 3 0011b: Clock divided by 4 0100b: Clock divided by 5 0101b: Clock divided by 6 0110b: Clock divided by 7 0111b: Clock divided by 8 1000b: Clock divided by 9 1001b: Clock divided by 10 1010b: Clock divided by 11 1011b: Clock divided by 12 1100b: Clock divided by 13	0x0

Field	Access	Description	Reset Value
		1101b: Clock divided by 14 1110b: Clock divided by 15 1111b: Clock divided by 16	
15:12 RSVD_15_12	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17 RSVD_19_17	RO	Reserved	0x0
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	When CAN_MCR[MDIS] is set to 1, CAN4 will be reset when this bit is cleared.	0x0
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x1
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x1
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.2.18 PARCC for CAN5 (PARCC_CAN5)

24.2.18.1 Offset

Register	Offset
PARCC_CAN5	0x40

24.2.18.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
3:2 RSVD_3_2	RO	Reserved	0x0
6:4 CLKMUX	RW	Select clock source for peripheral functional clock. 000b: OFF 001b: Reserved 010b: Reserved 011b: FIRC64M 100b: OSC40M 101b: SPLL	0x0
7 RSVD_7	RO	Reserved	0x0
11:8 CLKDIV	RW	Select clock divide ratio for peripheral functional clock. 0000b: Clock divided by 1 0001b: Clock divided by 2 0010b: Clock divided by 3 0011b: Clock divided by 4 0100b: Clock divided by 5 0101b: Clock divided by 6 0110b: Clock divided by 7 0111b: Clock divided by 8 1000b: Clock divided by 9 1001b: Clock divided by 10 1010b: Clock divided by 11 1011b: Clock divided by 12 1100b: Clock divided by 13 1101b: Clock divided by 14 1110b: Clock divided by 15 1111b: Clock divided by 16	0x0
15:12 RSVD_15_12	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17 RSVD_19_17	RO	Reserved	0x0
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	When CAN_MCR[MDIS] is set to 1, CAN5 will be reset when this bit is cleared.	0x0
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1

Field	Access	Description	Reset Value
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x1
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x1
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.2.19 PARCC for CAN6 (PARCC_CAN6)

24.2.19.1 Offset

Register	Offset
PARCC_CAN6	0x44

24.2.19.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
3:2 RSVD_3_2	RO	Reserved	0x0
6:4 CLKMUX	RW	Select clock source for peripheral functional clock. 000b: OFF 001b: Reserved 010b: Reserved 011b: FIRC64M 100b: OSC40M 101b: SPLL	0x0
7 RSVD_7	RO	Reserved	0x0
11:8 CLKDIV	RW	Select clock divide ratio for peripheral functional clock. 0000b: Clock divided by 1 0001b: Clock divided by 2 0010b: Clock divided by 3 0011b: Clock divided by 4	0x0

Field	Access	Description	Reset Value
		0100b: Clock divided by 5 0101b: Clock divided by 6 0110b: Clock divided by 7 0111b: Clock divided by 8 1000b: Clock divided by 9 1001b: Clock divided by 10 1010b: Clock divided by 11 1011b: Clock divided by 12 1100b: Clock divided by 13 1101b: Clock divided by 14 1110b: Clock divided by 15 1111b: Clock divided by 16	
15:12 RSVD_15_12	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17 RSVD_19_17	RO	Reserved	0x0
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	When CAN_MCR[MDIS] is set to 1, CAN6 will be reset when this bit is cleared.	0x0
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x1
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x1
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.2.20 PARCC for CAN7 (PARCC_CAN7)

24.2.20.1 Offset

Register	Offset
PARCC_CAN7	0x48

24.2.20.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
3:2 RSVD_3_2	RO	Reserved	0x0
6:4 CLKMUX	RW	Select clock source for peripheral functional clock. 000b: OFF 001b: Reserved 010b: Reserved 011b: FIRC64M 100b: OSC40M 101b: SPLL	0x0
7 RSVD_7	RO	Reserved	0x0
11:8 CLKDIV	RW	Select clock divide ratio for peripheral functional clock. 0000b: Clock divided by 1 0001b: Clock divided by 2 0010b: Clock divided by 3 0011b: Clock divided by 4 0100b: Clock divided by 5 0101b: Clock divided by 6 0110b: Clock divided by 7 0111b: Clock divided by 8 1000b: Clock divided by 9 1001b: Clock divided by 10 1010b: Clock divided by 11 1011b: Clock divided by 12 1100b: Clock divided by 13 1101b: Clock divided by 14 1110b: Clock divided by 15 1111b: Clock divided by 16	0x0
15:12 RSVD_15_12	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17 RSVD_19_17	RO	Reserved	0x0
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	When CAN_MCR[MDIS] is set to 1, CAN7 will be reset when this bit is cleared.	0x0

Field	Access	Description	Reset Value
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x1
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x1
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.2.21 PARCC for UART0 (PARCC_UART0)

24.2.21.1 Offset

Register	Offset
PARCC_UART0	0x4C

24.2.21.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
3:2 RSVD_3_2	RO	Reserved	0x0
6:4 CLKMUX	RW	Select clock source for peripheral functional clock. 000b: OFF 001b: LPO32K 010b: OSC32K 011b: FIRC64M 100b: OSC40M 101b: SPL	0x0
7 RSVD_7	RO	Reserved	0x0

Field	Access	Description	Reset Value
11:8 CLKDIV	RW	Select clock divide ratio for peripheral functional clock. 0000b: Clock divided by 1 0001b: Clock divided by 2 0010b: Clock divided by 3 0011b: Clock divided by 4 0100b: Clock divided by 5 0101b: Clock divided by 6 0110b: Clock divided by 7 0111b: Clock divided by 8 1000b: Clock divided by 9 1001b: Clock divided by 10 1010b: Clock divided by 11 1011b: Clock divided by 12 1100b: Clock divided by 13 1101b: Clock divided by 14 1110b: Clock divided by 15 1111b: Clock divided by 16	0x0
15:12 RSVD_15_12	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17 RSVD_19_17	RO	Reserved	0x0
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	Reset peripheral when this bit is clear.	0x0
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x1
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x1
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write	0x1

Field	Access	Description	Reset Value
		lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	

24.2.22 PARCC for UART1 (PARCC_UART1)

24.2.22.1 Offset

Register	Offset
PARCC_UART1	0x50

24.2.22.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
3:2 RSVD_3_2	RO	Reserved	0x0
6:4 CLKMUX	RW	Select clock source for peripheral functional clock. 000b: OFF 001b: LPO32K 010b: OSC32K 011b: FIRC64M 100b: OSC40M 101b: SPL	0x0
7 RSVD_7	RO	Reserved	0x0
11:8 CLKDIV	RW	Select clock divide ratio for peripheral functional clock. 0000b: Clock divided by 1 0001b: Clock divided by 2 0010b: Clock divided by 3 0011b: Clock divided by 4 0100b: Clock divided by 5 0101b: Clock divided by 6 0110b: Clock divided by 7 0111b: Clock divided by 8 1000b: Clock divided by 9 1001b: Clock divided by 10 1010b: Clock divided by 11 1011b: Clock divided by 12 1100b: Clock divided by 13 1101b: Clock divided by 14 1110b: Clock divided by 15 1111b: Clock divided by 16	0x0
15:12 RSVD_15_12	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17	RO	Reserved	0x0

Field	Access	Description	Reset Value
RSVD_19_17			
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	Reset peripheral when this bit is clear.	0x0
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x1
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x1
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.2.23 PARCC for UART2 (PARCC_UART2)

24.2.23.1 Offset

Register	Offset
PARCC_UART2	0x54

24.2.23.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
3:2 RSVD_3_2	RO	Reserved	0x0

Field	Access	Description	Reset Value
6:4 CLKMUX	RW	Select clock source for peripheral functional clock. 000b: OFF 001b: LPO32K 010b: OSC32K 011b: FIRC64M 100b: OSC40M 101b: SPLL	0x0
7 RSVD_7	RO	Reserved	0x0
11:8 CLKDIV	RW	Select clock divide ratio for peripheral functional clock. 0000b: Clock divided by 1 0001b: Clock divided by 2 0010b: Clock divided by 3 0011b: Clock divided by 4 0100b: Clock divided by 5 0101b: Clock divided by 6 0110b: Clock divided by 7 0111b: Clock divided by 8 1000b: Clock divided by 9 1001b: Clock divided by 10 1010b: Clock divided by 11 1011b: Clock divided by 12 1100b: Clock divided by 13 1101b: Clock divided by 14 1110b: Clock divided by 15 1111b: Clock divided by 16	0x0
15:12 RSVD_15_12	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17 RSVD_19_17	RO	Reserved	0x0
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	Reset peripheral when this bit is clear.	0x0
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x1
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x1
27	RO	Peripheral supervisor access control present	0x1

Field	Access	Description	Reset Value
PSUPVACPR		Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.2.24 PARCC for UART3 (PARCC_UART3)

24.2.24.1 Offset

Register	Offset
PARCC_UART3	0x58

24.2.24.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
3:2 RSVD_3_2	RO	Reserved	0x0
6:4 CLKMUX	RW	Select clock source for peripheral functional clock. 000b: OFF 001b: LPO32K 010b: OSC32K 011b: FIRC64M 100b: OSC40M 101b: SPL	0x0
7 RSVD_7	RO	Reserved	0x0
11:8 CLKDIV	RW	Select clock divide ratio for peripheral functional clock. 0000b: Clock divided by 1 0001b: Clock divided by 2 0010b: Clock divided by 3 0011b: Clock divided by 4 0100b: Clock divided by 5 0101b: Clock divided by 6 0110b: Clock divided by 7 0111b: Clock divided by 8 1000b: Clock divided by 9 1001b: Clock divided by 10 1010b: Clock divided by 11 1011b: Clock divided by 12 1100b: Clock divided by 13	0x0

Field	Access	Description	Reset Value
		1101b: Clock divided by 14 1110b: Clock divided by 15 1111b: Clock divided by 16	
15:12 RSVD_15_12	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17 RSVD_19_17	RO	Reserved	0x0
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	Reset peripheral when this bit is clear.	0x0
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x1
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x1
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.2.25 PARCC for UART4 (PARCC_UART4)

24.2.25.1 Offset

Register	Offset
PARCC_UART4	0x5C

24.2.25.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
3:2 RSVD_3_2	RO	Reserved	0x0
6:4 CLKMUX	RW	Select clock source for peripheral functional clock. 000b: OFF 001b: LPO32K 010b: OSC32K 011b: FIRC64M 100b: OSC40M 101b: SPILL	0x0
7 RSVD_7	RO	Reserved	0x0
11:8 CLKDIV	RW	Select clock divide ratio for peripheral functional clock. 0000b: Clock divided by 1 0001b: Clock divided by 2 0010b: Clock divided by 3 0011b: Clock divided by 4 0100b: Clock divided by 5 0101b: Clock divided by 6 0110b: Clock divided by 7 0111b: Clock divided by 8 1000b: Clock divided by 9 1001b: Clock divided by 10 1010b: Clock divided by 11 1011b: Clock divided by 12 1100b: Clock divided by 13 1101b: Clock divided by 14 1110b: Clock divided by 15 1111b: Clock divided by 16	0x0
15:12 RSVD_15_12	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17 RSVD_19_17	RO	Reserved	0x0
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	Reset peripheral when this bit is clear.	0x0
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1

Field	Access	Description	Reset Value
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x1
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x1
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.2.26 PARCC for UART5 (PARCC_UART5)

24.2.26.1 Offset

Register	Offset
PARCC_UART5	0x60

24.2.26.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
3:2 RSVD_3_2	RO	Reserved	0x0
6:4 CLKMUX	RW	Select clock source for peripheral functional clock. 000b: OFF 001b: LPO32K 010b: OSC32K 011b: FIRC64M 100b: OSC40M 101b: SPL	0x0
7 RSVD_7	RO	Reserved	0x0
11:8 CLKDIV	RW	Select clock divide ratio for peripheral functional clock. 0000b: Clock divided by 1 0001b: Clock divided by 2 0010b: Clock divided by 3 0011b: Clock divided by 4	0x0

Field	Access	Description	Reset Value
		0100b: Clock divided by 5 0101b: Clock divided by 6 0110b: Clock divided by 7 0111b: Clock divided by 8 1000b: Clock divided by 9 1001b: Clock divided by 10 1010b: Clock divided by 11 1011b: Clock divided by 12 1100b: Clock divided by 13 1101b: Clock divided by 14 1110b: Clock divided by 15 1111b: Clock divided by 16	
15:12 RSVD_15_12	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17 RSVD_19_17	RO	Reserved	0x0
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	Reset peripheral when this bit is clear.	0x0
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x1
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x1
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.2.27 PARCC for SPI0 (PARCC_SPI0)

24.2.27.1 Offset

Register	Offset
PARCC_SPI0	0x64

24.2.27.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
3:2 RSVD_3_2	RO	Reserved	0x0
6:4 CLKMUX	RW	Select clock source for peripheral functional clock. 000b: OFF 001b: LPO32K 010b: OSC32K 011b: FIRC64M 100b: OSC40M 101b: SPLL	0x0
7 RSVD_7	RO	Reserved	0x0
11:8 CLKDIV	RW	Select clock divide ratio for peripheral functional clock. 0000b: Clock divided by 1 0001b: Clock divided by 2 0010b: Clock divided by 3 0011b: Clock divided by 4 0100b: Clock divided by 5 0101b: Clock divided by 6 0110b: Clock divided by 7 0111b: Clock divided by 8 1000b: Clock divided by 9 1001b: Clock divided by 10 1010b: Clock divided by 11 1011b: Clock divided by 12 1100b: Clock divided by 13 1101b: Clock divided by 14 1110b: Clock divided by 15 1111b: Clock divided by 16	0x0
15:12 RSVD_15_12	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17 RSVD_19_17	RO	Reserved	0x0
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	Reset peripheral when this bit is clear.	0x0

Field	Access	Description	Reset Value
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x1
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x1
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.2.28 PARCC for SPI1 (PARCC_SPI1)

24.2.28.1 Offset

Register	Offset
PARCC_SPI1	0x68

24.2.28.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
3:2 RSVD_3_2	RO	Reserved	0x0
6:4 CLKMUX	RW	Select clock source for peripheral functional clock. 000b: OFF 001b: LPO32K 010b: OSC32K 011b: FIRC64M 100b: OSC40M 101b: SPLL	0x0
7 RSVD_7	RO	Reserved	0x0

Field	Access	Description	Reset Value
11:8 CLKDIV	RW	Select clock divide ratio for peripheral functional clock. 0000b: Clock divided by 1 0001b: Clock divided by 2 0010b: Clock divided by 3 0011b: Clock divided by 4 0100b: Clock divided by 5 0101b: Clock divided by 6 0110b: Clock divided by 7 0111b: Clock divided by 8 1000b: Clock divided by 9 1001b: Clock divided by 10 1010b: Clock divided by 11 1011b: Clock divided by 12 1100b: Clock divided by 13 1101b: Clock divided by 14 1110b: Clock divided by 15 1111b: Clock divided by 16	0x0
15:12 RSVD_15_12	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17 RSVD_19_17	RO	Reserved	0x0
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	Reset peripheral when this bit is clear.	0x0
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x1
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x1
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write	0x1

Field	Access	Description	Reset Value
		lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	

24.2.29 PARCC for SPI2 (PARCC_SPI2)

24.2.29.1 Offset

Register	Offset
PARCC_SPI2	0x6C

24.2.29.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
3:2 RSVD_3_2	RO	Reserved	0x0
6:4 CLKMUX	RW	Select clock source for peripheral functional clock. 000b: OFF 001b: LPO32K 010b: OSC32K 011b: FIRC64M 100b: OSC40M 101b: SPL	0x0
7 RSVD_7	RO	Reserved	0x0
11:8 CLKDIV	RW	Select clock divide ratio for peripheral functional clock. 0000b: Clock divided by 1 0001b: Clock divided by 2 0010b: Clock divided by 3 0011b: Clock divided by 4 0100b: Clock divided by 5 0101b: Clock divided by 6 0110b: Clock divided by 7 0111b: Clock divided by 8 1000b: Clock divided by 9 1001b: Clock divided by 10 1010b: Clock divided by 11 1011b: Clock divided by 12 1100b: Clock divided by 13 1101b: Clock divided by 14 1110b: Clock divided by 15 1111b: Clock divided by 16	0x0
15:12 RSVD_15_12	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17	RO	Reserved.	0x0

Field	Access	Description	Reset Value
RSVD_19_17			
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	Reset peripheral when this bit is clear.	0x0
23:22 RSVD_23_22	RO	Reserved.	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x1
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x1
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved.	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.2.30 PARCC for SPI3 (PARCC_SPI3)

24.2.30.1 Offset

Register	Offset
PARCC_SPI3	0x70

24.2.30.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
3:2 RSVD_3_2	RO	Reserved	0x0

Field	Access	Description	Reset Value
6:4 CLKMUX	RW	Select clock source for peripheral functional clock. 000b: OFF 001b: LPO32K 010b: OSC32K 011b: FIRC64M 100b: OSC40M 101b: SPLL	0x0
7 RSVD_7	RO	Reserved	0x0
11:8 CLKDIV	RW	Select clock divide ratio for peripheral functional clock. 0000b: Clock divided by 1 0001b: Clock divided by 2 0010b: Clock divided by 3 0011b: Clock divided by 4 0100b: Clock divided by 5 0101b: Clock divided by 6 0110b: Clock divided by 7 0111b: Clock divided by 8 1000b: Clock divided by 9 1001b: Clock divided by 10 1010b: Clock divided by 11 1011b: Clock divided by 12 1100b: Clock divided by 13 1101b: Clock divided by 14 1110b: Clock divided by 15 1111b: Clock divided by 16	0x0
15:12 RSVD_15_12	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17 RSVD_19_17	RO	Reserved	0x0
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	Reset peripheral when this bit is clear.	0x0
23:22 RSVD_23_22	RO	Reserved.	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x1
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x1
27	RO	Peripheral supervisor access control present	0x1

Field	Access	Description	Reset Value
PSUPVACPR		Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	
30:28 RSVD_30_28	RO	Reserved.	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.2.31 PARCC for I2C0 (PARCC_I2C0)

24.2.31.1 Offset

Register	Offset
PARCC_I2C0	0x74

24.2.31.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
3:2 RSVD_3_2	RO	Reserved	0x0
6:4 CLKMUX	RW	Select clock source for peripheral functional clock. 000b: OFF 001b: LPO32K 010b: OSC32K 011b: FIRC64M 100b: OSC40M 101b: SPLL	0x0
7 RSVD_7	RO	Reserved	0x0
11:8 CLKDIV	RW	Select clock divide ratio for peripheral functional clock. 0000b: Clock divided by 1 0001b: Clock divided by 2 0010b: Clock divided by 3 0011b: Clock divided by 4 0100b: Clock divided by 5 0101b: Clock divided by 6 0110b: Clock divided by 7 0111b: Clock divided by 8 1000b: Clock divided by 9 1001b: Clock divided by 10 1010b: Clock divided by 11 1011b: Clock divided by 12 1100b: Clock divided by 13	0x0

Field	Access	Description	Reset Value
		1101b: Clock divided by 14 1110b: Clock divided by 15 1111b: Clock divided by 16	
15:12 RSVD_15_12	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17 RSVD_19_17	RO	Reserved	0x0
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	Reset peripheral when this bit is clear.	0x0
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x1
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x1
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.2.32 PARCC for I2C1 (PARCC_I2C1)

24.2.32.1 Offset

Register	Offset
PARCC_I2C1	0x78

24.2.32.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
3:2 RSVD_3_2	RO	Reserved	0x0
6:4 CLKMUX	RW	Select clock source for peripheral functional clock. 000b: OFF 001b: LPO32K 010b: OSC32K 011b: FIRC64M 100b: OSC40M 101b: SPLL	0x0
7 RSVD_7	RO	Reserved	0x0
11:8 CLKDIV	RW	Select clock divide ratio for peripheral functional clock. 0000b: Clock divided by 1 0001b: Clock divided by 2 0010b: Clock divided by 3 0011b: Clock divided by 4 0100b: Clock divided by 5 0101b: Clock divided by 6 0110b: Clock divided by 7 0111b: Clock divided by 8 1000b: Clock divided by 9 1001b: Clock divided by 10 1010b: Clock divided by 11 1011b: Clock divided by 12 1100b: Clock divided by 13 1101b: Clock divided by 14 1110b: Clock divided by 15 1111b: Clock divided by 16	0x0
15:12 RSVD_15_12	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17 RSVD_19_17	RO	Reserved	0x0
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	Reset peripheral when this bit is clear.	0x0
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1

Field	Access	Description	Reset Value
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x1
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x1
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.2.33 PARCC for I2S0 (PARCC_I2S0)

24.2.33.1 Offset

Register	Offset
PARCC_I2S0	0x7C

24.2.33.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
3:2 RSVD_3_2	RO	Reserved	0x0
6:4 CLKMUX	RW	Select clock source for peripheral functional clock. 000b: OFF 001b: LPO32K 010b: OSC32K 011b: FIRC64M 100b: OSC40M 101b: SPLL	0x0
7 RSVD_7	RO	Reserved	0x0
11:8 CLKDIV	RW	Select clock divide ratio for peripheral functional clock. 0000b: Clock divided by 1 0001b: Clock divided by 2 0010b: Clock divided by 3 0011b: Clock divided by 4	0x0

Field	Access	Description	Reset Value
		0100b: Clock divided by 5 0101b: Clock divided by 6 0110b: Clock divided by 7 0111b: Clock divided by 8 1000b: Clock divided by 9 1001b: Clock divided by 10 1010b: Clock divided by 11 1011b: Clock divided by 12 1100b: Clock divided by 13 1101b: Clock divided by 14 1110b: Clock divided by 15 1111b: Clock divided by 16	
15:12 RSVD_15_12	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17 RSVD_19_17	RO	Reserved	0x0
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	Reset peripheral when this bit is clear.	0x0
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x1
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x1
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.2.34 PARCC for I2S1 (PARCC_I2S1)

24.2.34.1 Offset

Register	Offset
PARCC_I2S1	0x80

24.2.34.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
3:2 RSVD_3_2	RO	Reserved	0x0
6:4 CLKMUX	RW	Select clock source for peripheral functional clock. 000b: OFF 001b: LPO32K 010b: OSC32K 011b: FIRC64M 100b: OSC40M 101b: SPLL	0x0
7 RSVD_7	RO	Reserved	0x0
11:8 CLKDIV	RW	Select clock divide ratio for peripheral functional clock. 0000b: Clock divided by 1 0001b: Clock divided by 2 0010b: Clock divided by 3 0011b: Clock divided by 4 0100b: Clock divided by 5 0101b: Clock divided by 6 0110b: Clock divided by 7 0111b: Clock divided by 8 1000b: Clock divided by 9 1001b: Clock divided by 10 1010b: Clock divided by 11 1011b: Clock divided by 12 1100b: Clock divided by 13 1101b: Clock divided by 14 1110b: Clock divided by 15 1111b: Clock divided by 16	0x0
15:12 RSVD_15_12	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17 RSVD_19_17	RO	Reserved	0x0
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	Reset peripheral when this bit is clear.	0x0

Field	Access	Description	Reset Value
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x1
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x1
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.2.35 PARCC for ADC0 (PARCC_ADC0)

24.2.35.1 Offset

Register	Offset
PARCC_ADC0	0x88

24.2.35.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
3:2 RSVD_3_2	RO	Reserved	0x0
6:4 CLKMUX	RW	Select clock source for peripheral functional clock. 000b: OFF 001b: LPO32K 010b: OSC32K 011b: FIRC64M 100b: OSC40M 101b: Reserved	0x0
7 RSVD_7	RO	Reserved	0x0

Field	Access	Description	Reset Value
11:8 CLKDIV	RW	Select clock divide ratio for peripheral functional clock. 0000b: Clock divided by 1 0001b: Clock divided by 2 0010b: Clock divided by 3 0011b: Clock divided by 4 0100b: Clock divided by 5 0101b: Clock divided by 6 0110b: Clock divided by 7 0111b: Clock divided by 8 1000b: Clock divided by 9 1001b: Clock divided by 10 1010b: Clock divided by 11 1011b: Clock divided by 12 1100b: Clock divided by 13 1101b: Clock divided by 14 1110b: Clock divided by 15 1111b: Clock divided by 16	0x0
15:12 RSVD_15_12	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17 RSVD_19_17	RO	Reserved	0x0
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	Reset peripheral when this bit is clear.	0x0
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x1
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x1
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write	0x1

Field	Access	Description	Reset Value
		lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	

24.2.36 PARCC for ADC1 (PARCC_ADC1)

24.2.36.1 Offset

Register	Offset
PARCC_ADC1	0x8C

24.2.36.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
3:2 RSVD_3_2	RO	Reserved	0x0
6:4 CLKMUX	RW	Select clock source for peripheral functional clock. 000b: OFF 001b: LPO32K 010b: OSC32K 011b: FIRC64M 100b: OSC40M 101b: Reserved	0x0
7 RSVD_7	RO	Reserved	0x0
11:8 CLKDIV	RW	Select clock divide ratio for peripheral functional clock. 0000b: Clock divided by 1 0001b: Clock divided by 2 0010b: Clock divided by 3 0011b: Clock divided by 4 0100b: Clock divided by 5 0101b: Clock divided by 6 0110b: Clock divided by 7 0111b: Clock divided by 8 1000b: Clock divided by 9 1001b: Clock divided by 10 1010b: Clock divided by 11 1011b: Clock divided by 12 1100b: Clock divided by 13 1101b: Clock divided by 14 1110b: Clock divided by 15 1111b: Clock divided by 16	0x0
15:12 RSVD_15_12	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17	RO	Reserved	0x0

Field	Access	Description	Reset Value
RSVD_19_17			
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	Reset peripheral when this bit is clear.	0x0
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x1
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x1
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.2.37 PARCC for CMP0 (PARCC_CMP0)

24.2.37.1 Offset

Register	Offset
PARCC_CMP0	0x90

24.2.37.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
15:2 RSVD_15_2	RO	Reserved	0x0

Field	Access	Description	Reset Value
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17 RSVD_19_17	RO	Reserved	0x0
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	Reset peripheral when this bit is clear.	0x1
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x0
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x0
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.2.38 PARCC for CRC (PARCC_CRC)

24.2.38.1 Offset

Register	Offset
PARCC_CRC	0x94

24.2.38.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode.	0x0

Field	Access	Description	Reset Value
		11b: Reserved.	
15:2 RSVD_15_2	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17 RSVD_19_17	RO	Reserved	0x0
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	Reset peripheral when this bit is clear.	0x1
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x0
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x0
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.2.39 PARCC for AES (PARCC_AES)

24.2.39.1 Offset

Register	Offset
PARCC_AES	0x98

24.2.39.2 Fields

Field	Access	Description	Reset Value

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
15:2 RSVD_15_2	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17 RSVD_19_17	RO	Reserved	0x0
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	Reset peripheral when this bit is clear.	0x1
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x0
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x0
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.2.40 PARCC for TRNG (PARCC_TRNG)

24.2.40.1 Offset

Register	Offset
PARCC_TRNG	0x9C

24.2.40.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
15:2 RSVD_15_2	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17 RSVD_19_17	RO	Reserved	0x0
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	Reset peripheral when this bit is clear.	0x1
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x0
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x0
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.2.41 PARCC for FLASH (PARCC_FLASH)

24.2.41.1 Offset

Register	Offset

Register	Offset
PARCC_FLASH	0xA0

24.2.41.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x1
3:2 RSVD_3_2	RO	Reserved	0x0
6:4 CLKMUX	RW	Select clock source for peripheral functional clock. 000b: OFF 011b: FIRC64M 100b: OSC40M 101b: SPLL 110b: SLOW_CLK	0x3
7 RSVD_7	RO	Reserved	0x0
11:8 CLKDIV	RW	Select clock divide ratio for peripheral functional clock. 0000b: Clock divided by 1 0001b: Clock divided by 2 0010b: Clock divided by 3 0011b: Clock divided by 4 0100b: Clock divided by 5 0101b: Clock divided by 6 0110b: Clock divided by 7 0111b: Clock divided by 8 1000b: Clock divided by 9 1001b: Clock divided by 10 1010b: Clock divided by 11 1011b: Clock divided by 12 1100b: Clock divided by 13 1101b: Clock divided by 14 1110b: Clock divided by 15 1111b: Clock divided by 16	0x7
20:12 RSVD_20_12	RO	Reserved	0x0
21 PRSTB	RW	When this bit is cleared, FLASH will not be reset, but it cannot be read and written, any reading and writing will generate hard fault. FLASH registers can only be read and written when this bit is set to 1.	0x1
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x1
26	RO	Peripheral clock divider present	0x1

Field	Access	Description	Reset Value
PCLKDIVPR		Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.2.42 PARCC for DMA (PARCC_DMA)

24.2.42.1 Offset

Register	Offset
PARCC_DMA	0xA8

24.2.42.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
15:2 RSVD_15_2	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17 RSVD_19_17	RO	Reserved	0x0
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	When this bit is cleared, DMA will not be reset, but it cannot be read and written, any reading and writing will generate hard fault. DMA registers can only be read and written when this bit is set to 1.	0x1
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1

Field	Access	Description	Reset Value
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x0
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x0
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.2.43 PARCC for DMAMUX (PARCC_DMAMUX)

24.2.43.1 Offset

Register	Offset
PARCC_DMAMUX	0xAC

24.2.43.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
15:2 RSVD_15_2	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17 RSVD_19_17	RO	Reserved	0x0
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	Reset peripheral when this bit is clear.	0x1
23:22 RSVD_23_22	RO	Reserved	0x0

Field	Access	Description	Reset Value
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x0
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x0
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.2.44 PARCC for PORTA (PARCC_PORTA)

24.2.44.1 Offset

Register	Offset
PARCC_PORTA	0xB0

24.2.44.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
3:2 RSVD_3_2	RO	Reserved	0x0
6:4 CLKMUX	RW	Select clock source for peripheral functional clock. 000b: OFF 001b: LPO32K 010b: OSC32K 011b: FIRC64M 100b: OSC40M 101b: Reserved	0x0
7 RSVD_7	RO	Reserved	0x0
11:8 CLKDIV	RW	Select clock divide ratio for peripheral functional clock.	0x0

Field	Access	Description	Reset Value
		0000b: Clock divided by 1 0001b: Clock divided by 2 0010b: Clock divided by 3 0011b: Clock divided by 4 0100b: Clock divided by 5 0101b: Clock divided by 6 0110b: Clock divided by 7 0111b: Clock divided by 8 1000b: Clock divided by 9 1001b: Clock divided by 10 1010b: Clock divided by 11 1011b: Clock divided by 12 1100b: Clock divided by 13 1101b: Clock divided by 14 1110b: Clock divided by 15 1111b: Clock divided by 16	
15:12 RSVD_15_12	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17 RSVD_19_17	RO	Reserved	0x0
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	Reset peripheral when this bit is clear.	0x0
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x1
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x1
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable.	0x1

Field	Access	Description	Reset Value
		1b: Register is read-only.	

24.2.45 PARCC for PORTB (PARCC_PORTB)

24.2.45.1 Offset

Register	Offset
PARCC_PORTB	0xB4

24.2.45.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
3:2 RSVD_3_2	RO	Reserved	0x0
6:4 CLKMUX	RW	Select clock source for peripheral functional clock. 000b: OFF 001b: LPO32K 010b: OSC32K 011b: FIRC64M 100b: OSC40M 101b: Reserved	0x0
7 RSVD_7	RO	Reserved	0x0
11:8 CLKDIV	RW	Select clock divide ratio for peripheral functional clock. 0000b: Clock divided by 1 0001b: Clock divided by 2 0010b: Clock divided by 3 0011b: Clock divided by 4 0100b: Clock divided by 5 0101b: Clock divided by 6 0110b: Clock divided by 7 0111b: Clock divided by 8 1000b: Clock divided by 9 1001b: Clock divided by 10 1010b: Clock divided by 11 1011b: Clock divided by 12 1100b: Clock divided by 13 1101b: Clock divided by 14 1110b: Clock divided by 15 1111b: Clock divided by 16	0x0
15:12 RSVD_15_12	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17 RSVD_19_17	RO	Reserved	0x0

Field	Access	Description	Reset Value
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	Reset peripheral when this bit is clear.	0x0
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x1
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x1
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.2.46 PARCC for PORTC (PARCC_PORTC)

24.2.46.1 Offset

Register	Offset
PARCC_PORTC	0xB8

24.2.46.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
3:2 RSVD_3_2	RO	Reserved	0x0
6:4 CLKMUX	RW	Select clock source for peripheral functional clock. 000b: OFF	0x0

Field	Access	Description	Reset Value
		001b: LPO32K 010b: OSC32K 011b: FIRC64M 100b: OSC40M 101b: Reserved	
7 RSVD_7	RO	Reserved	0x0
11:8 CLKDIV	RW	Select clock divide ratio for peripheral functional clock. 0000b: Clock divided by 1 0001b: Clock divided by 2 0010b: Clock divided by 3 0011b: Clock divided by 4 0100b: Clock divided by 5 0101b: Clock divided by 6 0110b: Clock divided by 7 0111b: Clock divided by 8 1000b: Clock divided by 9 1001b: Clock divided by 10 1010b: Clock divided by 11 1011b: Clock divided by 12 1100b: Clock divided by 13 1101b: Clock divided by 14 1110b: Clock divided by 15 1111b: Clock divided by 16	0x0
15:12 RSVD_15_12	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17 RSVD_19_17	RO	Reserved	0x0
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	Reset peripheral when this bit is clear.	0x0
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x1
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x1
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not	0x1

Field	Access	Description	Reset Value
		present. 1b: Peripheral supervisor access control is present.	
30:28 RSVD_30_28	ROO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.2.47 PARCC for PORTD (PARCC_PORTD)

24.2.47.1 Offset

Register	Offset
PARCC_PORTD	0xBC

24.2.47.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
3:2 RSVD_3_2	RO	Reserved	0x0
6:4 CLKMUX	RW	Select clock source for peripheral functional clock. 000b: OFF 001b: LPO32K 010b: OSC32K 011b: FIRC64M 100b: OSC40M 101b: Reserved	0x0
7 RSVD_7	RO	Reserved	0x0
11:8 CLKDIV	RW	Select clock divide ratio for peripheral functional clock. 0000b: Clock divided by 1 0001b: Clock divided by 2 0010b: Clock divided by 3 0011b: Clock divided by 4 0100b: Clock divided by 5 0101b: Clock divided by 6 0110b: Clock divided by 7 0111b: Clock divided by 8 1000b: Clock divided by 9 1001b: Clock divided by 10 1010b: Clock divided by 11 1011b: Clock divided by 12 1100b: Clock divided by 13 1101b: Clock divided by 14 1110b: Clock divided by 15	0x0

Field	Access	Description	Reset Value
		1111b: Clock divided by 16	
15:12 RSVD_15_12	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17 RSVD_19_17	RO	Reserved	0x0
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	Reset peripheral when this bit is clear.	0x0
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x1
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x1
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.2.48 PARCC for PORTE (PARCC_PORTE)

24.2.48.1 Offset

Register	Offset
PARCC_PORTE	0xC0

24.2.48.2 Fields

Field	Access	Description	Reset Value

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
3:2 RSVD_3_2	RO	Reserved	0x0
6:4 CLKMUX	RW	Select clock source for peripheral functional clock. 000b: OFF 001b: LPO32K 010b: OSC32K 011b: FIRC64M 100b: OSC40M 101b: Reserved	0x0
7 RSVD_7	RO	Reserved	0x0
11:8 CLKDIV	RW	Select clock divide ratio for peripheral functional clock. 0000b: Clock divided by 1 0001b: Clock divided by 2 0010b: Clock divided by 3 0011b: Clock divided by 4 0100b: Clock divided by 5 0101b: Clock divided by 6 0110b: Clock divided by 7 0111b: Clock divided by 8 1000b: Clock divided by 9 1001b: Clock divided by 10 1010b: Clock divided by 11 1011b: Clock divided by 12 1100b: Clock divided by 13 1101b: Clock divided by 14 1110b: Clock divided by 15 1111b: Clock divided by 16	0x0
15:12 RSVD_15_12	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17 RSVD_19_17	RO	Reserved	0x0
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	Reset peripheral when this bit is clear.	0x0
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1
25 PCLKMUXPR	RO	Peripheral clock mux present Read only.	0x1

Field	Access	Description	Reset Value
		0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x1
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.2.49 PARCC for TMU (PARCC_TMU)

24.2.49.1 Offset

Register	Offset
PARCC_TMU	0xC4

24.2.49.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
15:2 RSVD_15_2	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17 RSVD_19_17	RO	Reserved	0x0
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	Reset peripheral when this bit is clear.	0x1
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present.	0x1

Field	Access	Description	Reset Value
		1b: Peripheral is present.	
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x0
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x0
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.2.50 PARCC for REGFILE (PARCC_REGFILE)

24.2.50.1 Offset

Register	Offset
PARCC_REGFILE	0xC8

24.2.50.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
15:2 RSVD_15_2	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17 RSVD_19_17	RO	Reserved	0x0
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	When this bit is cleared, REGFILE will not be reset, but it cannot be read and written, any reading and writing will generate hard fault. REGFILE registers can only be read and written when this bit is set to 1.	0x1

Field	Access	Description	Reset Value
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x0
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x0
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.2.51 PARCC for SMPU (PARCC_SMPU)

24.2.51.1 Offset

Register	Offset
PARCC_SMPU	0xCC

24.2.51.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x1
15:2 RSVD_15_2	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0
19:17 RSVD_19_17	RO	Reserved	0x0
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor	0x0

Field	Access	Description	Reset Value
		mode.	
21 PRSTB	RW	When this bit is cleared, SMPU will not be reset, but it cannot be read and written, any reading and writing will generate hard fault. SMPU registers can only be read and written when this bit is set to 1.	0x1
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x0
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x0
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.2.52 PARCC for GPIO (PARCC_GPIO)

24.2.52.1 Offset

Register	Offset
PARCC_GPIO	0xD0

24.2.52.2 Fields

Field	Access	Description	Reset Value
1:0 PCKMD	RW	Peripheral Clock Mode 00b: Clock is always off. 01b: Clock is always on. 10b: Clock is off in STOP mode. 11b: Reserved.	0x0
15:2 RSVD_15_2	RO	Reserved	0x0
16 PWLK	RW	Peripheral write lock 0b: Peripheral register write is allowed. 1b: Peripheral register write is denied.	0x0

Field	Access	Description	Reset Value
19:17 RSVD_19_17	RO	Reserved	0x0
20 PSUPVACEN	RW	Peripheral supervisor access control enable 0b: Bypass peripheral access control in supervisor mode. 1b: Allow peripheral access control in supervisor mode.	0x0
21 PRSTB	RW	Reset peripheral when this bit is clear.	0x1
23:22 RSVD_23_22	RO	Reserved	0x0
24 PPR	RO	Peripheral present Read only. 0b: Peripheral is not present. 1b: Peripheral is present.	0x1
25 PCLKMUXPR	RO	Peripheral clock mux present Read only. 0b: Peripheral function clock mux is not present. 1b: Peripheral function clock mux is present.	0x0
26 PCLKDIVPR	RO	Peripheral clock divider present Read only. 0b: Peripheral function clock divider is not present. 1b: Peripheral function clock divider is present.	0x0
27 PSUPVACPR	RO	Peripheral supervisor access control present Read only. 0b: Peripheral supervisor access control is not present. 1b: Peripheral supervisor access control is present.	0x1
30:28 RSVD_30_28	RO	Reserved	0x0
31 LOCK	RW	Write register lock Write 0x5B to PARCC[31:24] to disable lock when LOCK bit is set. Write 1 to LOCK bit to enable write lock when LOCK bit is clear. 0b: Register is writable. 1b: Register is read-only.	0x1

24.3 Function description

This section provides the functional information of the PARCC module. Each peripheral's PARCC register contains a write register lock bit (LOCK), which is for locking/unlocking the writing permission of the PARCC register itself. Clear this bit before changing any other fields of the PARCC register.

24.3.1 Peripheral register write protection

All of registers in peripheral are read-only after writing 1 to PARCC[PWLK], write operation is ignored. Write operation is acceptable after writing 0 to PARCC[PWLK].

24.3.2 Register access protection in supervisor mode

All of registers in peripheral are writable only in supervisor mode after writing 1 to PARCC[PSUPVACEN]. Write operation leads to request hard fault interrupt in user mode. Write operation is allowable in both supervisor and user modes after writing 0 to PARCC[PSUPVACEN].

24.3.3 Reset peripheral

Peripheral remains in reset status when writes 0 to PARCC[PRSTB], attempts to access register lead to request hard fault interrupt. Peripheral exit reset status after writes 1 to PARCC[PRSTB], registers are accessible.

24.3.4 Functional clock source switch

This feature allows the user to select one from a group of clock sources as a peripheral functional clock. Writes PARCC[CLKMUX] to select the corresponding clock. This register field is writable only when PARCC[PRSTB] or PARCC[PCKMD] is 0.

24.3.5 Functional clock source divide

This feature allows the user to divide the selected functional clock source by a specified divisor between 1 and 16. Write specified value to PARCC[CLKDIV] to determine the divisor. This register field is writable only when PARCC[PRSTB] or PARCC[PCKMD] is 0.

24.3.6 Clock mode

Always off mode:

Peripheral functional clock and bus clock are gated by PARCC. Attempts to access register lead to request hard fault interrupt in this mode.

Always on mode:

Peripheral functional clock remains active when system enters STOP mode.

Stop off mode:

Peripheral functional clock is gated when system enters STOP mode.

24.4 Initialization/Application information

This section provides the application information of the PARCC module.

24.4.1 Supervisor mode write access control

Run the following steps to configure PARCC:

1. If the MSB bit of a module's PARCC is set, the software first writes 0x5B to bit [31:24] to unlock the PARCC.
2. Select peripheral write clock by programming the PWLK field.
3. Select peripheral supervisor access control enable by programming the PSUPVACEN field.
4. Lock the PARCC by writing 1 to the LOCK bit.

See the PARCC register of each module for more details.

24.4.2 Reset control

Run the following steps to configure PARCC:

1. If the MSB bit of a module's PARCC is set, the software first writes 0x5B to bit [31:24] to unlock PARCC.
2. Write 0 to PRSTB bit.
3. Write 1 to PRSTB bit.
4. Lock the PARCC by writing 1 to the LOCK bit.

See the PARCC register of each module for more details.

24.4.3 Peripheral clock mode control

Run the following steps to configure PARCC:

1. If the MSB bit of a module's PARCC is set, software first writes 0x5B to bit [31:24] to unlock PARCC.
2. If PRSTB bit is cleared, software must write 1 to PRSTB bit to release the module from the reset state.
3. Select peripheral clock mode by programming the PCKMD field.
4. Lock the PARCC by writing 1 to the LOCK bit

See the PARCC register of each module for more details.

Chapter 25

Power Management Unit (PMU)

25.1 Introduction

This chapter defines a Power Management Unit (PMU) that describes the various chip power modes and functionality of the individual modules in these modes.

25.1.1 Features

The Power Management Unit (PMU) provides multiple power modes, allowing users to optimize power consumption to implement the required functionality. The primary modes of operation are:

- RUN
- WAIT
- STOP
- STANDBY

The Wait For Interrupt (WFI) instruction invokes WAIT, STOP, and STANDBY modes for the chip. For more information about WFI, see the details in ARM documentation. PMU mainly generates a stable and accurate voltage supply and current reference for on-chip module use. The PMU internal bandgap provides stable voltage and current reference for its regulators and LVD monitors.

The PMU consists of following components, as Figure 25-1 shows in its diagram.

- Power-on reset (POR) circuit.
- Low voltage warning detect circuit (LVW) is valid for vdd.
- Low voltage detect circuit (LVD) for vdd.
- LVD for LDO-CLOCK.
- LVD for LDO-CORE.
- LVD for LDO-FLASH.
- Band gap.
- Internal voltage regulators like LDO-CORE, LDO-FLASH, and LDO-CLOCK.
- PMU bandgap voltage can be measured via ADC0 and ADC1.
- PMU regulators voltage can be measured via ADC0 and ADC1.
- PMU register access lock protection.

25.1.2 Block diagram

The power structure for Z20K14xMC is shown in the diagram below.

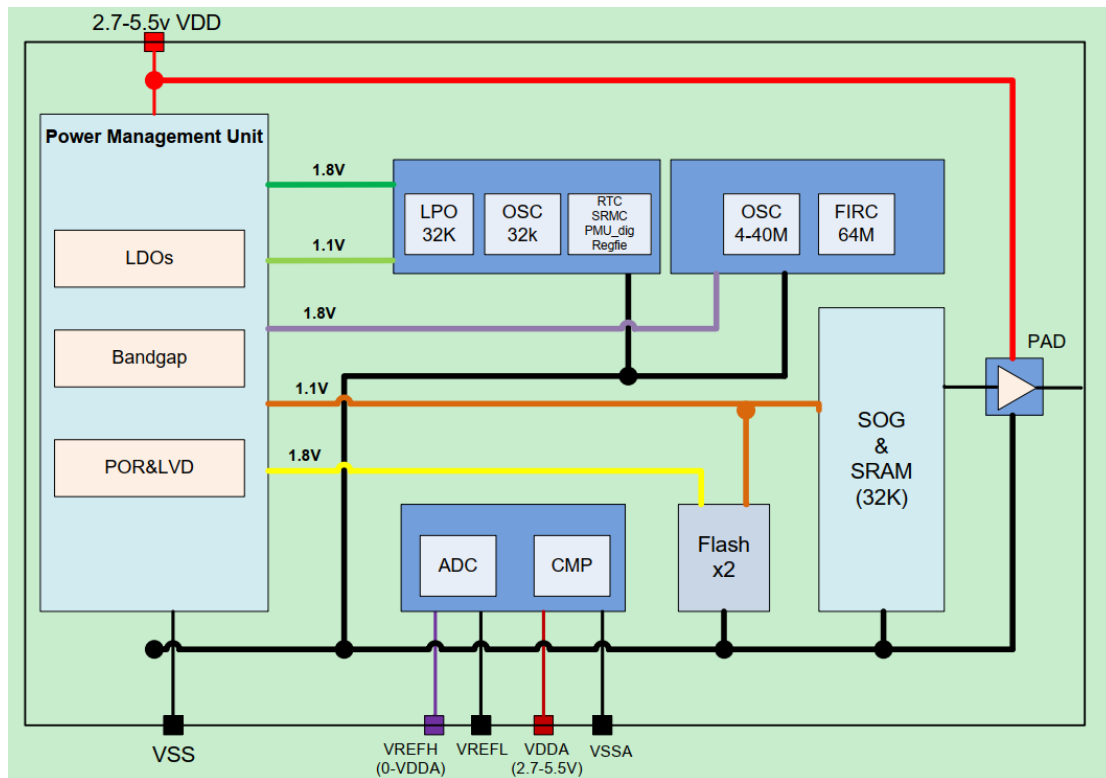


Figure 25-1 Power structure diagram

25.2 Power modes description

Depending on the power saving requirements of user application, various power modes that provide state retention, STANDBY mode or STOP mode of certain logic and/or memory are available. The following table compares the available power modes.

Table 25-1 Chip power modes

Chip mode	Description	Core mode	Normal recovery method
RUN	Default mode out of reset; the on-chip voltage regulator is on.	RUN	-
WAIT (via WFI instruction)	Places CPU in static state. In this power mode, all registers are retained and LVD protection is maintained. All power and clocks are maintained. NVIC is used to wake up from interrupt.	Sleep	All interrupts and system reset
STOP (via WFI instruction)	Places the chip in static state. In this power mode, all registers are retained and LVD protection is maintained. NVIC is disabled. WIC(IRQ logic) is used to wake up from interrupt. Some peripheral clocks are stopped.	Sleep Deep	Async interrupts and system reset

Chip mode	Description	Core mode	Normal recovery method
STANDBY (via WFI instruction)	It is recommended to enable all LVD functions. When chip enters the standby state, the hardware automatically disables LDO_CORE, LDO_CLOCK and LDO_FLASH. After it wakes up, the hardware automatically turns LDO_CORE, LDO_CLOCK and LDO_FLASH on. This is the lowest-power mode in which some pin interrupts are functional. Most peripheral clocks are stopped, only LPO32K, OSC32K and RTC can be used. 32 32-bit registers are used for user data retention. The RTC, VDD LVW/LVD interrupts and wakeup pin are used to wake with wakeup reset generated.	Sleep Deep	RTC, VDD LVW/LVD interrupts; Wakeup pin; POR/LVD reset.

25.3 Memory map and register definition

This section provides the memory map and register definitions of the PMU module. The PMU registers can be written only in supervisor mode.

25.3.1 Memory map

PMU base address: 0x4007D000

Table 25-2 PMU memory map

Offset	Register	Width (In bits)	Reset Value
0x00	PMU Lock Protection Register (PMU_LOCK)	32	0x80000000
0x04	PMU LPO32K Configure Register (PMU_LPO_32K_CFG)	32	0x00000004
0x0C	PMU Isolation Clear Register (PMU_ISO_CLR)	32	0x00000000
0x10	PMU Control and Status Register (PMU_CSR)	32	0x0000000A
0x14	PMU LDO Enable Configure Register (PMU_LDO_EN_CFG)	32	0x00000007
0x18	PMU LDO LVD Status Register (PMU_LDO_LVD_STATUS)	32	0x00000000
0x1C	PMU LDO LVD Configure Register (PMU_LDO_LVD_CFG)	32	0x00000277

25.3.2 PMU Lock Protection Register (PMU_LOCK)

25.3.2.1 Offset

Register	Offset
PMU_LOCK	0x00

25.3.2.2 Function

This register is reset via POR or LVD reset.

25.3.2.3 Fields

Field	Access	Description	Reset Value
30:0 RSVD_30_0	RO	Reserved	0x0
31 LOCK	RW	The default value LOCK is 1 after the POR reset. After the POR reset of the chip, LOCK will be 0 if you write the unlock key PMU_LOCK[31:16] = 0x9473, LOCK will be 0. After LOCK=0, the LOCK bit will be re-asserted to 1 when PMU_LOCK[31]=1 has been written to PMU_LOCK register. The LOCK bit can also be asserted to 1 via chip POR or LVD reset. 0b: Allow writing to PMU registers. 1b: Do not allow writing to PMU registers.	0x1

25.3.3 PMU LPO32K Configure Register (PMU_LPO_32K_CFG)

25.3.3.1 Offset

Register	Offset
PMU_LPO_32K_CFG	0x04

25.3.3.2 Function

This register is reset via POR reset. It can be written only when PMU_LOCK[LOCK] = 0.

25.3.3.3 Fields

Field	Access	Description	Reset Value
0 LPO_CLK_DIS	RW	LPO clock disable bit. 0b: Enable LPO raw clock. 1b: Disable LPO raw clock. Note: This field must be set to 0.	0x0
1 RSVD_1	RO	Reserved	0x0
2 LPO_CLK_READY	RO	LPO raw clock ready flag. 0b: LPO raw clock is not ready. 1b: LPO raw clock startup stable and ready for use.	0x1
31:3 RSVD_31_3	RO	Reserved	0x0

25.3.4 PMU Isolation Clear Register (PMU_ISO_CLR)

25.3.4.1 Offset

Register	Offset
PMU_ISO_CLR	0xC

25.3.4.2 Function

After exiting from STANDBY mode, PMU_ISO_CLR[ISO_CLR] will be asserted to 1'b1. It

will isolate the IO pads and ADC/CMP/SPLL modules. This bit need to W1C before any option after exiting from STANDBY mode.

25.3.4.3 Fields

Field	Access	Description	Reset Value
0 ISO_CLR	W1C	0b: Not isolated. 1b: Isolate the IO pads and ADC/CMP/SPLL modules after exiting from STANDBY mode.	0x0
31:1 RSVD_31_1	RO	Reserved	0x0

25.3.5 PMU Control and Status Register (PMU_CSR)

25.3.5.1 Offset

Register	Offset
PMU_CSR	0x10

25.3.5.2 Function

This register is reset via POR reset. It can be written only when PMU_LOCK[LOCK] =0.

25.3.5.3 Fields

Field	Access	Description	Reset Value
0 VDD_LVD_LP_EN	RW	VDD5V LVD detect control bit under low power mode 0b: VDD5V LVD detect disable under low power mode. 1b: VDD5V LVD LP detect enable under low power mode. Note: This field must be set to 1.	0x0
1 VDD_LVD_ACT_EN	RW	VDD5V LVD detect control bit under active mode 0b: VDD5V LVD detect disable under active mode. 1b: VDD5V LVD detect enable under active mode. Note: This field must be set to 1.	0x1
2 VDD_LVD_IE	RW	VDD5V LVD interrupt enable control 0b: VDD5V LVD interrupt disable. 1b: VDD5V LVD interrupt enable.	0x0
3 VDD_LVD_RE	RW	VDD5V LVD reset enable control bit 0b: VDD5V LVD reset disable. 1b: VDD5V LVD reset enable. Note: This field must be set to 1.	0x1
4 VDD_LVW_EN	RW	VDD5V LVW detect control bit. 0b: VDD5V low voltage detect circuit disable. 1b: VDD5V low voltage detect circuit enable.	0x0
5 VDD_LVW_IE	RW	VDD5V LVW interrupt enable 0b: VDD5V LVW interrupt disable. 1b: VDD5V LVW interrupt enable.	0x0
6	W1C	VDD5V LVW detect flag	0x0

Field	Access	Description	Reset Value
LVWF_VDD		0b: No VDD5V LVW flag occurs. 1b: VDD5V LVW flag occurs.	
7 LVDF_VDD	W1C	VDD5V LVD flag 0b: No VDD5v LVD flag occurs. 1b: VDD5V LVD flag occurs.	0x0
8 REF_BUF_1V_EN	RW	Reference 1V output enable with buffer control 0b: Disable REF1V buffer. 1b: Enable REF1V buffer. Note: This field must be set to 1 if bandgap was monitored under ADC.	0x0
31:9 RSVD_31_9	RO	Reserved	0x0

25.3.6 PMU LDO Enable Configure Register (PMU_LDO_EN_CFG)

25.3.6.1 Offset

Register	Offset
PMU_LDO_EN_CFG	0x14

25.3.6.2 Function

This register is reset via POR or LVD reset.

25.3.6.3 Fields

Field	Access	Description	Reset Value
0 LDO_CORE_EN	RW	LDO-CORE regulator enable control 0b: Means disable LDO-CORE. 1b: Means LDO-CORE regulator enable. Note: This field must be set to 1.	0x1
1 LDO_CLOCK_EN	RW	LDO-CLOCK regulator enable control 0b: Means disable LDO-CLOCK. 1b: Means LDO-CLOCK regulator enable. Note: This field must be set to 1.	0x1
2 LDO_FLASH_EN	RW	LDO-FLASH regulator enable control 0b: Means disable LDO-FLASH. 1b: Means LDO-FLASH regulator enable. Note: This field must be set to 1.	0x1
3 RSVD_3	RO	Reserved and read as 0	0x0
4 LDO_CORE_LPEN	RW	LDO-CORE low drive strength enable under STOP mode 0b: Means LDO-CORE high drive strength enable under STOP mode. 1b: Means LDO-CORE low drive strength enable under STOP mode.	0x0
5	RW	LDO-CLOCK low drive strength mode enable under	0x0

Field	Access	Description	Reset Value
LDO_CLOCK_LPEN		STOP mode 0b: Means LDO-CLOCK high drive strength enable under STOP mode. 1b: Means LDO-CLOCK low drive strength enable under STOP mode.	
6 LDO_FLASH_LPEN	RW	LDO-FLASH low drive strength mode enable under STOP mode 0b: Means LDO-FLASH high drive strength mode enable under STOP mode. 1b: Means LDO-FLASH low drive strength mode enable under STOP mode.	0x0
31:7 RSVD_31_7	RO	Reserved and read as 0	0x0

25.3.7 PMU LDO LVD Status Register (PMU_LDO_LVD_STATUS)

25.3.7.1 Offset

Register	Offset
PMU_LDO_LVD_STATUS	0x18

25.3.7.2 Function

This register is reset via POR only.

25.3.7.3 Fields

Field	Access	Description	Reset Value
0 LVD_CORE_FLAG	W1C	LDO CORE LVD flag 0b: No LDO CORE LVD occurs. 1b: LDO CORE LVD occurs.	0x0
1 LVD_CLOCK_FLAG	W1C	LDO CLOCK LVD flag 0b: No LDO CLOCK LVD occurs. 1b: LDO CLOCK LVD occurs.	0x0
2 LVD_FLASH_FLAG	W1C	LDO FLASH LVD flag 0b: No LDO FLASH LVD occurs. 1b: LDO FLASH LVD occurs.	0x0
3 RSVD_3	RO	Reserved	0x0
4 LVD_AOA_FLAG	W1C	Always-on analog LDO LVD flag 0b: No LDO_AOA LVD occurs. 1b: LDO_AOA LVD occurs.	0x0
31:5 RSVD_31_5	RO	Reserved	0x0

25.3.8 PMU LDO LVD Configure Register (PMU_LDO_LVD_CFG)

25.3.8.1 Offset

Register	Offset
PMU_LDO_LVD_CFG	0x1C

25.3.8.2 Function

This register is via POR reset only.

25.3.8.3 Fields

Field	Access	Description	Reset Value
0 LVD_CORE_RE	RW	0b: No LDO-CORE LVD reset occurs when LDO-CORE LVD flag is asserted to 1. 1b: LDO-CORE LVD reset occurs when LDO-CORE LVD flag is asserted to 1. Note: This field is recommended to be set as 1.	0x1
1 LVD_CLOCK_RE	RW	0b: No LDO-CLOCK LVD occurs when LDO-CLOCK LVD flag is asserted to 1. 1b: LDO-CLOCK LVD reset occurs when LDO-CLOCK LVD flag is asserted to 1. Note: This field is recommended to be set as 1.	0x1
2 LVD_FLASH_RE	RW	0b: No LDO-Flash LVD reset occurs when LDO-Flash LVD flag is asserted to 1. 1b: LDO-Flash LVD reset occurs when LDO-Flash LVD flag is asserted to 1. Note: This field is recommended to be set as 1.	0x1
3 RSVD_3	RO	Reserved	0x0
4 LVD_AOA_RE	RW	0b: No LDO-AOA LVD reset occurs when LDO-AOA LVD flag is asserted to 1. 1b: LDO-AOA LVD reset will occur when LDO-AOA LVD flag is asserted to 1. Note: This field is recommended to be set as 1.	0x1
5 ACT_CORE_LVDE	RW	0b: LDO-CORE LVD detect circuit disable under active mode. 1b: LDO-CORE LVD detect circuit enable under active mode. Note: This field is recommended to be set as 1.	0x1
6 ACT_CLOCK_LVDE	RW	0b: LDO-CLOCK LVD detect circuit disable under active mode. 1b: LDO-CLOCK LVD detect circuit enable under active mode. Note: This field is recommended to be set as 1.	0x1
8:7 RSVD_8_7	RO	Reserved	0x0
9 ACT_AOA_LVDE	RW	0b: LDO-AOA LVD detect circuit disable under active mode. 1b: LDO-AOA LVD detect circuit enable under active mode.	0x1

Field	Access	Description	Reset Value
		Note: This field is recommended to be set as 1.	
10 LP_CORE_LVDE	RW	Valid under STOP mode. 0b: LDO-CORE LVD detect circuit disable. 1b: LDO-CORE LVD detect circuit enable. Note: This field is recommended to be set as 1 if entering STOP mode.	0x0
11 LP_CLOCK_LVDE	RW	Valid under STOP mode. 0b: LDO-CLOCK LVD detect circuit disable. 1b: LDO-CLOCK LVD detect circuit enable. Note: This field is recommended to be set as 1 if entering STOP mode.	0x0
13:12 RSVD_13_12	RO	Reserved	0x0
14 LP_AOA_LVDE	RW	Valid under STOP and STANDBY mode. 0b: Disable LDO-AOA LVD detect circuit. 1b: Enable LDO-AOA LVD detect circuit. Note: This field is recommended to be set as 1 if entering STOP mode or STANDBY mode.	0x0
31:15 RSVD_31_15	RO	Reserved	0x0

25.4 Function description

This section provides the functional information of the PMU module.

25.4.1 Clocking gating

You can turn off most module clocks to conserve power by configuring the PRSTB and PCKMD fields of the module reset and clock control registers in the PARCC module. These fields are cleared after any reset, which disables the Protocol clock of the corresponding module. Before a module is initialized, set the corresponding field in the PARCC peripheral control register to enable the module clock. Make sure that you have disabled the module before turning off the clock.

25.4.2 Power mode transitions

The following figure shows the power mode transitions. Any reset always brings the chip back to the Run state. In the RUN, WAIT, STOP, and STANDBY modes, the active power regulation is enabled.

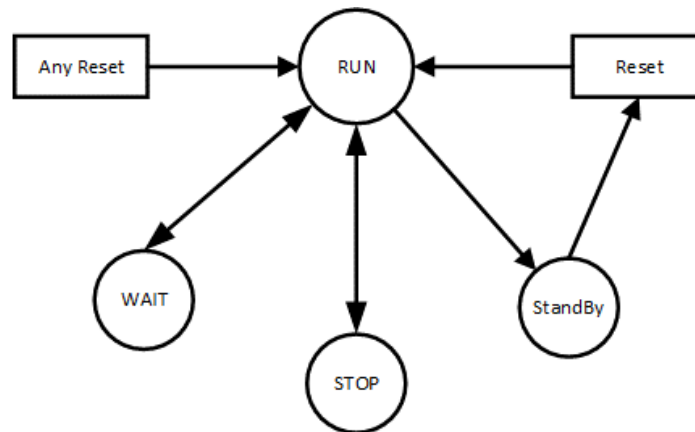


Figure 25-2 Power mode state transition diagram

25.4.3 Module operation in available power modes

The module functionality in each of the available modes is shown in the tables below.

Table 25-3 Terms and abbreviations

Term	Definition
Async operation	Fully functional with alternate clock source (if the selected clock source remains enabled).
FF	Full functionality. It means module functionality only depends on software configuration for this module.
ON	Module and clock sources should be enabled.
OFF	Module and clock sources are disabled.
Low power	The circuit is in low power mode.

Table 25-4 Module function in power modes

Modules	RUN	WAIT	STOP	STANDBY
PMU digital	On	On	On	On
Bandgap	On	On	On	On
POR	On	On	On	On
LVD/LVR	FF	FF	FF	VDD5V LVD FF LDOx LVD Off
OSC-40M	FF	FF	FF	Off
FIRC-64M	FF	FF	FF	Off
SPLL	FF	FF	FF	Off
LPO-32K	On	On	On	On
OSC-32K	FF	FF	FF	FF
CORE_CLK	FF	Off	Off	Off
BUS_CLK	FF	FF	Off	Off
Flash_CLK	FF	FF	Off	Off
ARM core	RUN	Sleep	Sleep deep	Sleep deep
SMPU	FF	FF	Off	Off
DMA	FF	FF	Off	Off

Modules	RUN	WAIT	STOP	STANDBY
Cache	FF	Off	Off	Off
Flash	FF	FF	Flash Sleep mode	Off
SRAM_U	FF	FF	No access	Off
SRAM_L	FF	FF	No access	Off
Regfile	FF	FF	No access	Retention
WDOG	FF	FF	Async operation	Off
EWDT	FF	FF	Off	Off
RTC	FF	FF	FF	FF
STIM	FF	FF	Async operation	Off
Timer	FF	FF	Off	Off
MCPWM	FF	FF	Off	Off
TDG	FF	FF	Off	Off
CAN FD	FF	FF	PNET supported	Off
UART/LIN	FF	FF	Async operation	Off
SPI	FF	FF	Off	Off
I2C	FF	FF	Off	Off
I2S	FF	FF	Off	Off
ADC	FF	FF	Async operation	Off
CMP	FF	FF	FF	Off
CRC	FF	FF	Off	Off
Crypto Engine	FF	FF	Off	Off
GPIO	FF	FF	Off	Off
PORT	FF	FF	Async operation	Async operation on AOD PORT

25.4.4 POR (power-on reset)

PMU will generate a POR reset during VDD5V ramp-up phase. It includes VDD5V POR and LDO-AOD LVD reset. Either VDD5V POR or LDO-AOD LVD reset will cause PMU internal LDO re-boot and the whole chip reset.

25.4.5 LVD (low voltage detect) circuit

PMU includes a low voltage detect circuit to monitor that chip supply drops to low level.

25.4.5.1 LVD of VDD5V

For VDD5V supply, it has an individual LVD detect circuit. In addition, it has a VDD_LVD_ACT_EN bit to enable this detect feature under active mode and has a VDD_LVD_LP_EN bit for low power mode. LVD detect is effective under low power mode only when both PMU_CSR[VDD_LVD_ACT_EN] and PMU_CSR[VDD_LVD_LP_EN] are set to 1.

When VDD5V drops lower than its trip point level LVW_VDD5V (see VLVD in Z20K14xMC Datasheet), it will generate a flag 'LVDF_VDD' on the PMU_CSR register. For example, if setting the interrupt enable bit $\text{PMU\_CSR[VDD\_LVD\_IE]} = 1$, PMU will

generate an LVD interrupt here. Similarly, if setting the reset enable bit `PMU_CSR[VDD_LVD_RE]` to 1, PMU will generate an LVD reset here. The flag can be cleared via the w1c option (write 1 to this bit `LVDF_VDD`).

25.4.5.2 LVW of VDD5V

For VDD5V supply, it also has an individual low voltage warning LVW detect circuit. When VDD5V drops lower than its trip point level `LVW_VDD5V` (see `VLVW` in Z20K14xMC Datasheet), it will generate a flag 'LVWF_VDD' on the `PMU_CSR` register. If setting the interrupt enable bit `PMU_CSR[VDD_LVW_IE] = 1`, PMU will generate a LVW interrupt here. The flag can be cleared via w1c option (write 1 to this bit `LVWF_VDD`).

Chapter 26

System Reset and Mode Control (SRMC)

26.1 Introduction

The chapter defines a System Reset and Mode Controller (SRMC).

The System Reset and Mode Control Module (SRMC) is responsible for system reset and sequencing the system into and out of all low-power and RUN modes.

The SRMC block provides the system reset and low power mode control of the MCU. The SRMC module monitors events to trigger the transitions between power modes while controlling the power, clocks, and memories of the system to achieve the power consumption and functionality of that mode.

26.1.1 Features

The SRMC supports the following key features:

- Power-On Reset (POR):
- System resets:
 - External pin reset (PIN)
 - Low-voltage detect (LVD)
 - Watchdog reset
 - Loss of clock reset
 - Low power mode acknowledge error (SACKERR)
 - Software reset (SW)
 - Core Lockup reset (LOCKUP)
 - ADM-AP system reset
 - SERU normal reset
 - SERU cold reset
- Chip mode of operation:
 - RUN mode: The MCU can run at full speed and the internal supply is fully regulated.
 - WAIT mode: CPU is under sleep mode, NVIC is sensitive to interrupt. Peripherals continued to be clocked.
 - STOP mode: In STOP mode, system clock, core clock, and the bus clocks are gated. CPU is under deep sleep mode, NVIC is disabled. Peripheral clocks are off. SOG and SRAM are powered on.
 - STANDBY mode: Most peripherals are off except AO domain like RTC, REGFILE, etc.

26.2 Memory map and register definition

This section provides the memory map and register definitions of the SRMC module.

26.2.1 Memory map

SRMC base address: 0x4007E000

Table 26-1 SRMC memory map

Offset	Register	Width (In bits)	Reset Value
0x04	Parameter Register (SRMC_PARAM)	32	0x00012EE7
0x08	System Reset Status Register	32	0x00000082

Offset	Register	Width (In bits)	Reset Value
	(SRMC_SRS)		
0x0C	Reset Control Register (SRMC_CTRL)	32	0x00000000
0x18	Sticky System Reset Status Register (SRMC_SSRS)	32	0x00000082
0x1C	System Reset Interrupt Enable Register (SRMC_SRIE)	32	0x00000000
0x20	Power Mode Protection Register (SRMC_PMPORT)	32	0x00000000
0x24	Power Mode Control Register (SRMC_PMCTRL)	32	0x00000002
0x28	Power Mode Status Register (SRMC_PMSTAT)	32	0x00000001
0x2C	DSM Wakeup Control Register (SRMC_DSMWUPC1)	32	0x80080000
0x30	DSM Wakeup Control Register (SRMC_DSMWUPC2)	32	0xFFFF0000
0x34	DSM Wakeup Status Register (SRMC_DSMWUPS)	32	0x00000000

26.2.2 Parameter Register (SRMC_PARAM)

26.2.2.1 Offset

Register	Offset
SRMC_PARAM	0x04

26.2.2.2 Function

This register is used to indicate design feature implementation.

26.2.2.3 Fields

Field	Access	Description	Reset Value
0 EWAKEUP	RO	Existence of SRS[WAKEUP] status indication feature This static bit states whether or not the feature is available on the device. 0b: The feature is not available. 1b: The feature is available.	0x1
1 ELVD	RO	Existence of SRS[LVD] status indication feature This static bit states whether or not the feature is available on the device. 0b: The feature is not available. 1b: The feature is available.	0x1
2 ELOC	RO	Existence of SRS[LOC] status indication feature This static bit states whether the feature is available on the device. 0b: The feature is not available. 1b: The feature is available.	0x1
4:3 RSVD_4_3	RO	Reserved	0x0
5 EWDG	RO	Existence of SRS[EWOG] status indication feature This static bit states whether the feature is available on	0x1

Field	Access	Description	Reset Value
		the device. 0b: The feature is not available. 1b: The feature is available.	
6 EPIN	RO	Existence of SRS[EPIN] status indication feature This static bit states whether the feature is available on the device. 0b: The feature is not available. 1b: The feature is available.	0x1
7 EPOR	RO	Existence of SRS[POR] status indication feature This static bit states whether the feature is available on the device. 0b: The feature is not available. 1b: The feature is available.	0x1
8 RSVD_8	RO	Reserved	0x0
9 ELOCKUP	RO	Existence of SRS[LOCKUP] status indication feature This static bit states whether the feature is available on the device. 0b: The feature is not available. 1b: The feature is available.	0x1
10 ESW	RO	Existence of SRS[SW] status indication feature This static bit states whether the feature is available on the device. 0b: The feature is not available. 1b: The feature is available.	0x1
11 EADM_AP	RO	Existence of SRS[ADM_AP] status indication feature This static bit states whether the feature is available on the device. 0b: The feature is not available. 1b: The feature is available.	0x1
12 RSVD_12	RO	Reserved	0x0
13 ESACKERR	RO	Existence of SRS[SACKERR] status indication feature This static bit states whether or not the feature is available on the device. 0b: The feature is not available. 1b: The feature is available.	0x1
15:14 RSVD_15_14	RO	Reserved	0x0
16 ESTB	RO	Existence of STANDBY feature This static bit states whether or not the feature is available on the device. 0b: The feature is not available. 1b: The feature is available.	0x1
31:17 RSVD_31_17	RO	Reserved	0x0

26.2.3 System Reset Status Register (SRMC_SRS)

26.2.3.1 Offset

Register	Offset
SRMC_SRS	0x08

26.2.3.2 Function

This register records the last occurred reset cause.

26.2.3.3 Fields

Field	Access	Description	Reset Value
0 WAKEUP	RO	Indicates a reset has been caused by wakeup from STANDBY mode. 0b: Reset was not caused by a standby wakeup. 1b: Reset was caused by a standby wakeup.	0x0
1 LVD	RO	Low Voltage Reset Indicates a reset has been caused by PMU low voltage detected. The PMU must be enabled for a low voltage to be detected. Refer to the detailed PMU description for information on enabling the voltage monitor. 0b: Reset was not caused by a low voltage detected. 1b: Reset was caused by a low voltage detected.	0x1
2 LOC	RO	Loss Of Clock Reset Indicates a reset has been caused by a loss of clock. The SCC clock monitor must be enabled for a loss of clock to be detected. Refer to the detailed SCC description for information on enabling the clock monitor. 0b: Reset was not caused by a loss of clock. 1b: Reset was caused by a loss of clock.	0x0
4:3 RSVD_4_3	RO	Reserved	0x0
5 WDOG	RO	WDOG Reset Indicates a reset has been caused by the watchdog timer timeout. This reset source can be blocked by disabling the watchdog. 0b: Reset was not caused by watchdog timeout. 1b: Reset was caused by watchdog timeout.	0x0
6 PIN	RO	External Pin Reset Indicates a reset has been caused by an active-low level on the external RESET pin. 0b: Reset was not caused by external reset pin. 1b: Reset was caused by external reset pin.	0x0
7 POR	RO	Power-On Reset Indicates a reset has been caused by the power-on detection logic. Because the internal supply voltage was ramping up at the time, the low-voltage reset (LVD) status bit is also set to indicate that the reset occurred while the internal supply was below the LVD threshold. 0b: Reset was not caused by POR. 1b: Reset was caused by POR.	0x1
8 RSVD_8	RO	Reserved	0x0
9 LOCKUP	RO	LOCKUP Reset Indicates a reset has been caused by the ARM core indication of a LOCKUP event. 0b: Reset was not caused by core LOCKUP event. 1b: Reset was caused by core LOCKUP event.	0x0
10 SW	RO	Software Reset Indicates a reset has been caused by software setting of SYSRESETREQ bit in Application Interrupt and	0x0

Field	Access	Description	Reset Value
		Reset Control Register in the ARM core. 0b: Reset was not caused by software setting of SYSRESETREQ bit. 1b: Reset was caused by software setting of SYSRESETREQ bit.	
11 ADM_AP	RO	ADM-AP System Reset Request Indicates a reset has been caused by the host debugger system setting of the System Reset Request bit in the ADM-AP Control Register. 0b: Reset was not caused by host debugger system setting of the System Reset Request bit. 1b: Reset was caused by host debugger system setting of the System Reset Request bit.	0x0
12 SERU_COLD	RO	SERU Cold Reset Request Indicates a reset has been caused by SERU cold reset request. 0b: Reset was not caused by SERU cold reset request. 1b: Reset was caused by SERU cold reset request.	0x0
13 SERU_SYS	RO	SERU System Reset Request Indicates a reset has been caused by SERU system reset request. 0b: Reset was not caused by SERU system reset request. 1b: Reset was caused by SERU system reset request.	0x0
14 RSVD_14	RO	Reserved	0x0
15 SACKERR	RO	Low Power Mode Acknowledge Error Indicates that after an attempt to enter low power mode (STOP mode or STANDBY mode), a reset has been caused by a failure of one or more peripherals to acknowledge within configurable timeout to enter low power mode. 0b: Reset was not caused by peripheral failure to acknowledge the attempt to enter the low power mode. 1b: Reset was caused by peripheral failure to acknowledge the attempt to enter the low power mode.	0x0
31:16 RSVD_31_16	RO	Reserved	0x0

26.2.4 Reset Control Register (SRMC_CTRL)

26.2.4.1 Offset

Register	Offset
SRMC_CTRL	0x0C

26.2.4.2 Function

This register is used for the reset configuration.
The SRMC_CTRL register can be written only in supervisor mode.

26.2.4.3 Fields

Field	Access	Description	Reset Value

Field	Access	Description	Reset Value
1:0 RSTFLTSRW	RW	Reset Pin Filter Select in RUN and WAIT Modes Selects how the reset pin filter is enabled in the RUN and WAIT modes. 00b: All filtering disabled. 01b: Bus clock filter enabled for normal operation. 10b: LPO clock filter enabled for normal operation. 11b: Reserved.	0x0
2 RSTFLTSS	RW	Reset Pin Filter Select in STOP Mode Selects how the reset pin filter is enabled in STOP mode. 0b: All filtering disabled. 1b: LPO clock filter enabled.	0x0
7:3 RSVD_7_3	RW	Reserved	0x0
12:8 RSTFLTSEL	RW	Reset Pin Filter Bus Clock Select Selects the reset pin bus clock filter width. Transitions for less than (RSTFLTSEL+1) bus clock cycles are always filtered, transitions equal to (RSTFLTSEL+1) bus clock cycles may be filtered.	0x0
30:13 RSVD_30_13	RO	Reserved	0x0
31 LOCKUP_RST_EN	RW	Core Lockup Reset enable 1b: Core Lockup Reset enable. 0b: Core Lockup Reset disable.	0x0

26.2.5 Sticky System Reset Status Register (SRMC_SSRS)

26.2.5.1 Offset

Register	Offset
SRMC_SSRS	0x18

26.2.5.2 Function

This register records all occurred reset causes until being cleared.
The SRMC_SSRS register can be written only in supervisor mode.

26.2.5.3 Fields

Field	Access	Description	Reset Value
0 SWAKEUP	W1C	Indicates a reset has been caused by wakeup from STANDBY mode. 0b: Reset was not caused by a standby wakeup. 1b: Reset was caused by a standby wakeup.	0x0
1 SLVD	W1C	Sticky low voltage reset Indicates a reset has been caused by PMU low voltage detected. The PMU must be enabled for a low voltage to be detected. Refer to the detailed PMU description for information on enabling the voltage monitor. 0b: Reset was not caused by a low voltage detected. 1b: Reset was caused by a low voltage detected.	0x1

Field	Access	Description	Reset Value
2 SLOC	W1C	Sticky loss of external clock. Indicates a reset has been caused by a loss of external clock. The SCC clock monitor must be enabled for a loss of clock to be detected. Refer to the detailed SCC description for information on enabling the clock monitor. 0b: Reset was not caused by a loss of external clock. 1b: Reset was caused by a loss of external clock.	0x0
4:3 RSVD_4_3	RO	Reserved	0x0
5 SWDOG	W1C	Sticky watchdog timer timeout Indicates a reset has been caused by the watchdog timer timeout. This reset source can be blocked by disabling the watchdog. 0b: Reset was not caused by watchdog timeout. 1b: Reset was caused by watchdog timeout.	0x0
6 SPIN	W1C	Sticky external RESET pin Reset Indicates a reset has been caused by an active-low level on the external RESET pin. 0b: Reset was not caused by external reset pin. 1b: Reset was caused by external reset pin.	0x0
7 SPOR	W1C	Sticky Power-On Reset Indicates a reset has been caused by the power-on detection logic. Because the internal supply voltage was ramping up at the time, the low-voltage reset (LVD) status bit is also set to indicate that the reset occurred while the internal supply was below the LVD threshold. 0b: Reset was not caused by POR. 1b: Reset was caused by POR.	0x1
8 RSVD_8	RO	Reserved	0x0
9 SLOCKUP	W1C	Sticky Core Lock Up Indicates a reset has been caused by the ARM core indication of a LOCKUP event. 0b: Reset was not caused by core LOCKUP event. 1b: Reset was caused by core LOCKUP event.	0x0
10 SSW	W1C	Sticky Software Reset Indicates a reset has been caused by software setting of SYSRESETREQ bit in Application Interrupt and Reset Control Register in the ARM core. 0b: Reset was not caused by software setting of SYSRESETREQ bit. 1b: Reset was caused by software setting of SYSRESETREQ bit.	0x0
11 SADM_AP	W1C	Sticky ADM-AP System Reset Request Indicates a reset has been caused by the host debugger system setting of the System Reset Request bit in the ADM-AP Control Register. 0b: Reset was not caused by host debugger system setting of the System Reset Request bit. 1b: Reset was caused by host debugger system setting of the System Reset Request bit.	0x0
12	W1C	Sticky SERU Cold Reset Request	0x0

Field	Access	Description	Reset Value
SSERU_COLD		Indicates a reset has been caused by SERU cold reset request. 0b: Reset was not caused by SERU cold reset request. 1b: Reset was caused by SERU cold reset request.	
13 SSERU_SYS	W1C	Sticky SERU System Reset Request Indicates a reset has been caused by SERU system reset request. 0b: Reset was not caused by SERU system reset request. 1b: Reset was caused by SERU system reset request.	0x0
14 RSVD_14	RO	Reserved	0x0
15 SSACKERR	W1C	Sticky Low Power Mode Acknowledge Error Indicates that after an attempt to enter low power mode (STOP mode or STANDBY mode), a reset has been caused by a failure of one or more peripherals to acknowledge within configurable timeout to enter low power mode. 0b: Reset was not caused by peripheral failure to acknowledge the attempt to enter the low power mode. 1b: Reset was caused by peripheral failure to acknowledge the attempt to enter the low power mode.	0x0
31:16 RSVD_31_16	RO	Reserved	0x0

26.2.6 System Reset Interrupt Enable Register (SRMC_SRIE)

26.2.6.1 Offset

Register	Offset
SRMC_SRIE	0x1C

26.2.6.2 Function

This register is used for the control interrupt.
The SRMC_SRIE register can be written only in supervisor mode.

26.2.6.3 Fields

Field	Access	Description	Reset Value
1:0 DELAY	RW	Maximum reset delay time Configures the maximum reset delay time from the interrupt assertion to the system reset occurrence. 00b: 8 LPO cycles 01b: 32 LPO cycles 10b: 128 LPO cycles 11b: 512 LPO cycles	0x0
2 LOC	RW	Loss-of-Clock Interrupt Enable 0b: Interrupt is disabled. 1b: Interrupt is enabled.	0x0

Field	Access	Description	Reset Value
4:3 RSVD_4_3	RO	Reserved	0x0
5 WDOG	RW	Watchdog Interrupt Enable 0b: Interrupt disabled. 1b: Interrupt enabled.	0x0
6 PIN	RW	External Reset Pin Interrupt Enable 0b: Reset not caused by external reset pin. 1b: Reset caused by external reset pin.	0x0
7 GIE	RW	Global Interrupt Enable 0b: All interrupt sources disabled. 1b: All interrupt sources enabled.	0x0
8 RSVD_8	RO	Reserved	0x0
9 LOCKUP	RW	Core Lockup Interrupt 0b: Interrupt disabled. 1b: Interrupt enabled.	0x0
10 SW	RW	Software Reset Interrupt Enable 0b: Interrupt disabled. 1b: Interrupt enabled.	0x0
11 RSVD_11	RO	Reserved	0x0
12 SERU_COLD	RW	SERU Cold Reset Interrupt 0b: Interrupt disabled. 1b: Interrupt enabled.	0x0
13 SERU_SYS	RW	SERU System Reset Interrupt 0b: Interrupt disabled. 1b: Interrupt enabled.	0x0
14 RSVD_14	RO	Reserved	0x0
15 SACKERR	RW	Stop Acknowledge Error Interrupt 0b: Interrupt disabled. 1b: Interrupt enabled.	0x0
31:16 RSVD_31_16	RO	Reserved	0x0

26.2.7 Power Mode Protection register (SRMC_PMPORT)

26.2.7.1 Offset

Register	Offset
SRMC_PMPORT	0x20

26.2.7.2 Function

This register is used for write protection for power mode configuration register. The SRMC_PMPORT register can be written only in supervisor mode.

26.2.7.3 Fields

Field	Access	Function	Reset Value
0 APD	RW	Provided the appropriate control bits are set up in PMCTRL, this write-once field allows the MCU to enter STANDBY mode (STANDBY).	0x0

Field	Access	Function	Reset Value
		0b: Standby is not allowed. 1b: Standby is allowed.	
31:1 RSVD_31_1	RO	Reserved	0x0

26.2.8 Power Mode Control Register (SRMC_PMCTRL)

26.2.8.1 Offset

Register	Offset
SRMC_PMCTRL	0x24

26.2.8.2 Function

This register is used to configure power mode.
The SRMC_PMCTRL register can be written only in supervisor mode.

26.2.8.3 Fields

Field	Access	Description	Reset Value
1:0 DSMC	RW	Low Power Mode Configuration When written, controls entry into the selected power mode. Writes to this field are blocked if the protection level has not been enabled using the PMPORT register. After any system reset, this field is cleared by hardware on any successful write to the PMPORT register. 10b: STOP 11b: STANDBY Others: Reserved	0x2
7:2 RSVD_7_2	RO	Reserved	0x0
15:8 DSMACKTMO	RW	Low Power Mode Acknowledge Time Out This field value is to determine how many LPO cycles reached and no acknowledge received will cause time out reset. 8'0b: No timeout function. Others: Represents DSMACKTMO * 128 LPO clocks. Note: It is recommended to configure DSMACKTMO as non-zero.	0x0
31:16 RSVD_31_16	RO	Reserved	0x0

26.2.9 Power Mode Status Register (SRMC_PMSTAT)

26.2.9.1 Offset

Register	Offset
SRMC_PMSTAT	0x28

26.2.9.2 Function

This register is used to indicate the power mode status.

26.2.9.3 Fields

Field	Access	Description	Reset Value
3:0 PMSTAT	RO	Power Mode Status 0001b: Current power mode is RUN. 0010b: Current power mode is WAIT. 0100b: Current power mode is STOP. 1000b: Current power mode is STANDBY. NOTE When debug is enabled, the PMSTAT will not update to STOP. When a STOP mode is enabled, the PMSTAT will not update to STOP.	0x1
7:4 RSVD_7_4	RO	Reserved	0x0
8 DSMABORT	RO	Low Power Mode Aborted When set, this read-only status bit indicates an interrupt occurred during the previous low power mode entry sequence, preventing the system from entering that mode. This field is cleared by reset or by hardware at the beginning of any low power mode entry sequence. This field is set if the sequence was aborted. 0b: The previous low power mode entry was successful. 1b: The previous low power mode entry was aborted.	0x0
31:9 RSVD_31_9	RO	Reserved	0x0

26.2.10 DSM Wakeup Control Register (SRMC_DSMWUPC1)

26.2.10.1 Offset

Register	Offset
SRMC_DSMWUPC1	0x2C

26.2.10.2 Function

This register is for the STANDBY mode wakeup source configuration.
The SRMC_DSMWUPC1 register can be written only in supervisor mode.

26.2.10.3 Fields

Field	Access	Description	Reset Value
2:0 WUPSRC_EN2_0	RW	Wakeup Source Enable When set, the corresponding STANDBY mode wake-up source can wakeup STANDBY mode. 1b: Wakeup is enabled. 0b: Wakeup is disabled. Each bit controls the relevant wake-up source: Bit0: PTD0 pin;	0x0

Field	Access	Description	Reset Value
		Bit1: PTD1 pin; Bit2: PTA4 pin	
3 RSVD_3	RO	Reserved	0x0
15:4 WUPSRC_EN15_4	RW	Wakeup Source Enable When set, the corresponding STANDBY mode wake-up source can wakeup STANDBY mode. 0b: Wakeup disable. 1b: Wakeup enable. Each bit controls relevant wake-up source: Bit4: PTC4 pin; Bit5: PTC5 pin; Bit6: PTE0 pin; Bit7: PTE1 pin; Bit8: PTE11 pin; Bit9: PTE13 pin; Bit10: PTE15 pin; Bit11: PTE16 pin; Bit12: PTA8 pin; Bit13: PTA9 pin; Bit14: PTA25 pin; Bit15: PMU	0x0
18:16 WUPSRC_PL2_0	RW	Wake-up Source Polarity When set, the corresponding wake-up source is active high. When cleared, the corresponding wake-up source is active low. 0b: Active low. 1b: Active high.	0x0
19 RSVD_19	RO	Reserved	0x1
30:20 WUPSRC_PL14_4	RW	Wake-up Source Polarity When set, the corresponding wake-up source is active high. When cleared, the corresponding wake-up source is active low. 0b: Active low. 1b: Active high.	0x0
31 RSVD_31	RO	Reserved	0x1

26.2.11 DSM Wakeup Control Register (SRMC_DSMWUPC2)

26.2.11.1 Offset

Register	Offset
SRMC_DSMWUPC2	0x30

26.2.11.2 Function

This register is for the STANDBY mode wakeup source configuration.
The SRMC_DSMWUPC2 register can be written only in supervisor mode.

26.2.11.3 Field

Field	Access	Description	Reset Value
0 WUPSRC_EN16	RW	Wakeup Source Enable When set, the corresponding wake-up source can wakeup STANDBY mode. 0b: Wakeup disable. 1b: Wakeup enable. Each bit controls the relevant wake-up source: Bit0: RTC	0x0
31:1 RSVD_31_1	RO	Reserved	0x7FFF8000

26.2.12 DSM Wakeup Status Register (SRMC_DSMWUPS)

26.2.12.1 Offset

Register	Offset
SRMC_DSMWUPS	0x34

26.2.12.2 Function

This register is for the STANDBY mode wakeup source status indication.
The SRMC_DSMWUPS register can be written only in supervisor mode.

26.2.12.3 Fields

Field	Access	Description	Reset Value
2:0 WUPS_2_0	W1C	Wake-up Status Indicates the corresponding wake-up source woke up system from STANDBY mode. 0b: System is not waked up by this source. 1b: System is waked up by this source. Each bit controls the relevant wake-up source: Bit0: PTD0 pin; Bit1: PTD1 pin; Bit2: PTA4 pin	0x0
3 RSVD_3	RO	Reserved	0x0
16:4 WUPS_16_4	W1C	Wake-up Status Indicates the corresponding wake-up source woke up system from STANDBY mode. 0b: System is not waked up by this source. 1b: System is waked up by this source. Each bit controls the relevant wake-up source: Bit4: PTC4 pin; Bit5: PTC5 pin; Bit6: PTE0 pin; Bit7: PTE1 pin; Bit8: PTE11 pin; Bit9: PTE13 pin; Bit10: PTE15 pin; Bit11: PTE16 pin; Bit12: PTA8 pin;	0x0

Field	Access	Description	Reset Value
		Bit13: PTA9 pin; Bit14: PTA25 pin; Bit15: PMU Bit16: RTC	
31:17 RSVD_31_17	RO	Reserved	0x0

26.3 Function description

This section provides the functional information of the SRMC module.

26.3.1 Reset generate

The SRMC implements many of the reset functions for the chip. See the chip's reset and boot chapter for more information.

26.3.2 Mode control

26.3.2.1 RUN mode

The RUN mode is the normal operating mode for the device.

This mode is selected after any reset. When the ARM processor exits reset, it sets up the stack, program counter (PC), and link register (LR):

- The processor reads the start SP (SP_main) from vector-table offset 0x000
- The processor reads the start PC from vector-table offset 0x004
- LR is set to 0xFFFF_FFFF.

To reduce power in this mode, disable the clocks to unused modules using their corresponding clock gating control bits in the SCM registers.

26.3.2.2 WAIT mode

The WAIT mode is entered when the ARM core enters the Sleep-Now or Sleep-On-Exit modes while SLEEPDEEP is cleared. The ARM CPU enters a low-power state in which it is not clocked, but peripherals continue to be clocked provided they are enabled. Clock gating to the peripheral is enabled via the PARCC module.

When an interrupt request occurs, the CPU exits the WAIT mode and resumes processing in RUN mode, beginning with the stacking operations leading to the interrupt service routine.

A system reset will cause an exit from WAIT mode, returning the device to normal RUN mode.

26.3.2.3 STOP mode

The STOP mode is entered via the sleep-now or sleep-on-exit with the SLEEPDEEP bit set in the System Control Register in the ARM core.

The SCC module can be configured to leave the reference clocks running.

A module that is able to provide an asynchronous interrupt to the device takes the device out of STOP mode and returns the device to normal RUN mode. Refer to the chapter of the device's Power Management for peripheral, I/O, and memory operation in STOP mode.

When an interrupt request occurs, the CPU exits the STOP mode and resumes processing, beginning with the stacking operations leading to the interrupt service routine.

A system reset will cause an exit from STOP mode, returning the device to normal RUN mode via an MCU reset.

The device contains the following wakeup source to the WIC module:

1. Available system resets
2. Pin interrupts
3. ADC
4. CMP
5. UART/CAN/RTC
6. NMI
7. PMU
8. SRMC
9. SERU
10. STIM

26.3.2.4 STANDBY mode

The standby mode is allowed via the PMPORT[APD] bit and enabled via the SRMC_PMCTRL[DSMC] bit and SLEEPDEEP bit of the System Control Register in the ARM core. The STANDBY mode wakeup source contains the following:

1. PTD0
2. PTD1
3. PTA4
4. PTA5(reset pin)
5. PTC4
6. PTC5
7. PTE0
8. PTE1
9. PTE11
10. PTE13
11. PTE15
12. PTE16
13. PTA8
14. PTA9
15. PTA25
16. PMU
17. RTC

Note: The wakeup source pins and PTA10, PTA11 get into the state of high impedance or the input after the device enters the STANDBY mode, the other pins can maintain the voltage level. The holding time of the valid wake-up level must be greater than 50ns.

The PTA5 is by default enabled as a low polarity wakeup source. No extra operation is needed when PTA5 is used to wake up the system from STANDBY mode. After wakeup in this way, the flags of PTA5 in SRMC_SRS[WAKEUP] and SRMC_SRS[PIN] will be set.

26.3.2.5 Power mode transitions

The following table shows the power mode state transitions available on the chip. Any reset always brings the MCU back to the normal RUN state.

The following table defines the triggers for the various state transitions shown in Figure 25-2.

Table 26-2 Power mode transition triggers

Transition	From	To	Trigger conditions
1	RUN	WAIT	Sleep-now or sleep-on-exit modes entered with SLEEPDEEP clear, which is controlled in System Control Register in ARM core.
2	WAIT	RUN	Interrupt or reset
3	RUN	STOP	Sleep-now or sleep-on-exit modes entered with SLEEPDEEP set, which is controlled in System Control Register in ARM core, and SRMC_PMCTRL[DSMC] is 2.

Transition	From	To	Trigger conditions
4	STOP	RUN	Interrupt or reset
5	RUN	STANDBY	Deep sleep request with setting SRMC_PMCTRL[DSMC] = 3 to enter the standby mode
6	STANDBY	RUN	Standby wakeup or reset

When entering the STOP mode, clocks are shut off in an orderly sequence to place the chip in the targeted low-power state safely. When entering the STANDBY mode, powers are shut off in an orderly sequence as well. When the core executes a WFI instruction, all low-power entry sequences are initiated. The core outputs, SLEEPDEEP and SLEEPING trigger an entry to the various low-power modes.

- WAIT mode equates to: SLEEPING
- STOP and STANDBY modes equate to: SLEEPDEEP

The chip performs the following actions in sequence to enter to and exit from the WAIT mode:

1. Configure ARM register SCR[SLEEPDEEP] = 0x0
2. Execute instruction WFI to enter low power mode
3. CPU will enter WAIT mode immediately, but all clocks and powers are maintained.
4. Any interrupts will wake-up WAIT mode to execute corresponding ISR function immediately. Any resets will wake-up WAIT mode to execute reset sequence immediately.

The following figure shows the sequence to enter to and exit from the STOP mode:

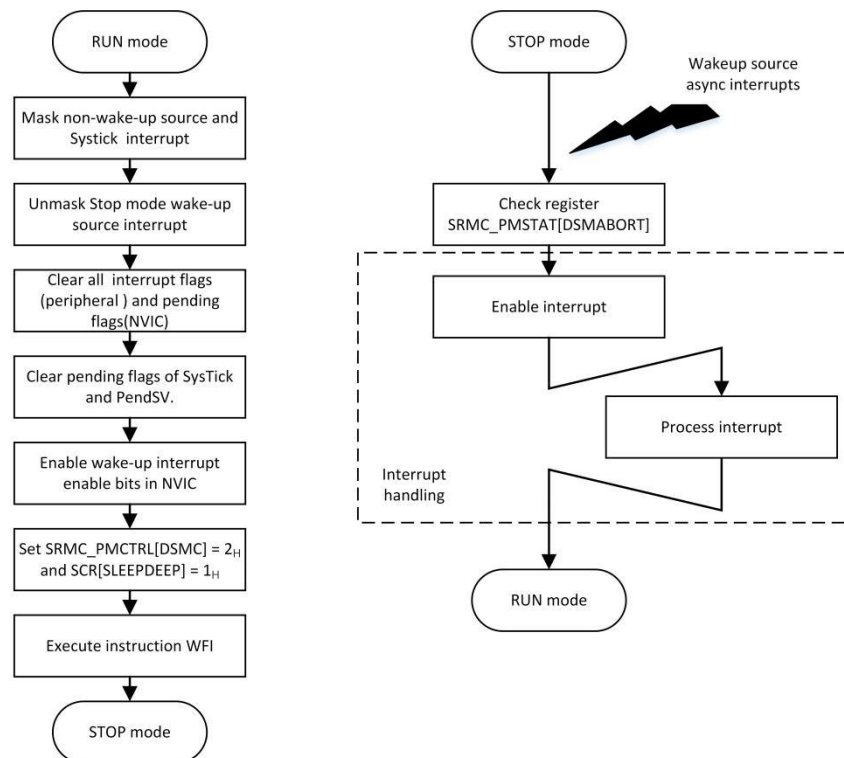


Figure 26-1 Sequence for STOP mode

1. Mask interrupts of all peripherals that are not used as wake-up sources.
2. Mask interrupt of SysTick.
3. Unmask interrupts of the peripherals that are used as wake-up sources and configure STOP mode wake-up interrupt (interrupt enable or polarity).
4. Clear all interrupt flags in the peripheral registers.
5. Clear all pending interrupt flags in the NVIC.
6. Clear pending interrupt flags of SysTick and PendSV.
7. Enable interrupt enable bits in NVIC corresponding to peripherals with unmasked

interrupts.

8. Configure SRMC register SRMC_PMCTRL[DSMC] = 0x2.
9. Configure ARM register SCR[SLEEPDEEP] = 0x1.
10. Execute instruction WFI to enter low power mode.
11. The chip immediately shuts off core and system clocks to the ARM core.
12. The chip polls the stop-acknowledge indications from supporting peripherals and the flash memory controller for indications that system clocks, bus clock, and/or flash memory clock need to be left enabled to complete a previously initiated operation, effectively stalling entry to the targeted low power mode. When all acknowledges are detected, system clock, bus clock and flash memory clock are turned off simultaneously.
13. The SCC and SRMC shut off clock sources based on SCC stop configuration and PMU changes corresponding supply and detect circuits to low power mode.
14. Any async interrupts will wakeup STOP mode, SRMC will request PMU to resume corresponding supply and detect circuits to normal mode, then SCC resume clock sources as SCC configuration after getting request from SRMC.
15. When system and peripheral clocks are resumed, CPU starts to execute corresponding ISR function.
16. Any resets will wake up STOP mode to execute reset sequence immediately.

The following table show different condition enter the STOP mode.

Table 26-3 Phenomenon of entering STOP mode

Condition	Phenomena
Correct operation to enter STOP.	Device enters STOP mode, power consumption reduction. After the wake-up interrupt signal, the program can continue to execute.
The NVIC interrupt flag is not cleared, execute STOP process.	CPU halt, device power consumption unchanged. After a interrupt signal, the program can continue to execute.
Device attempts to enter the STOP but not all module acknowledge STOP request.	For SRMC_PMCTRL[DSMACKTMO] is non-zero, the device will reset and set SRMC_SRS[SACKERR] when it is timeout. For SRMC_PMCTRL[DSMACKTMO] is zero, the device will keep waiting until the module acknowledge.

The following figure shows the sequence to enter to and exit from the STANDBY mode:

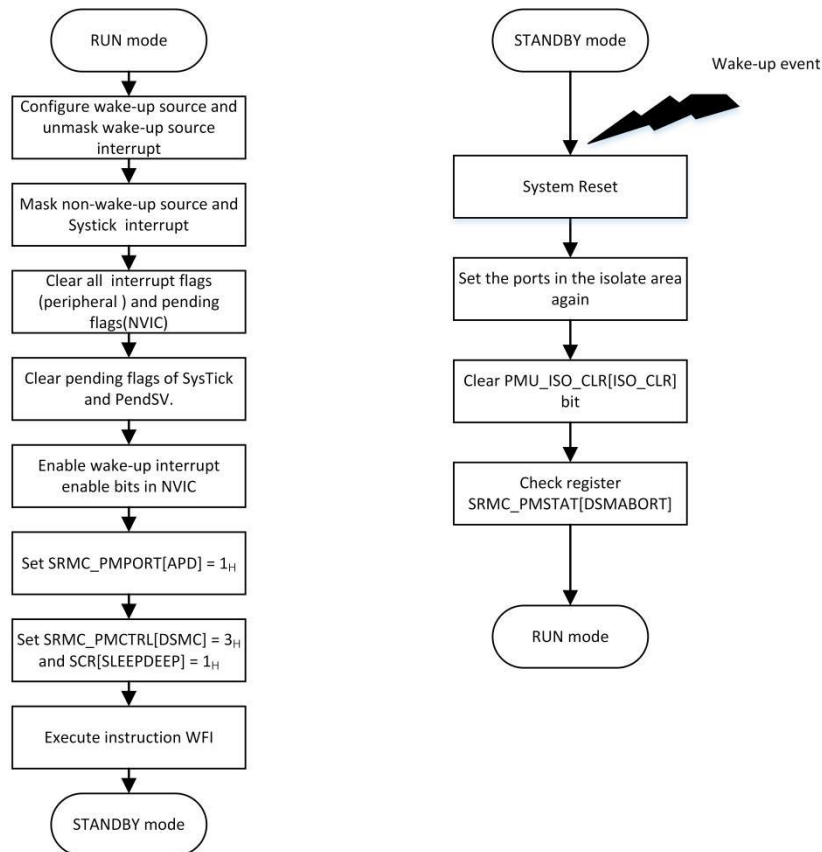


Figure 26-2 Sequence for STANDBY mode

1. Configure SRMC_DSMWUPC1/SRMC_DSMWUPC2 to set wake-up source enable and polarity. Before entering STANDBY mode, need to make the wake-up source non-wake-up state.
2. Unmask interrupts of the peripherals that are used as wake-up sources and configure them to ensure that an interrupt signal can be generated at the same time as the wake-up signal is generated.
3. Mask interrupts of all peripherals that are not used as wake-up sources.
4. Mask interrupt of SysTick.
5. Clear all interrupt flags in the peripheral registers.
6. Clear all pending interrupt flags in the NVIC.
7. Clear pending interrupt flags of SysTick and PendSV.
8. Enable interrupt enable bits in NVIC corresponding to peripherals with unmasked interrupts.
9. Configure SRMC register SRMC_PMPORT[APD] = 0x1.
10. Configure SRMC register SRMC_PMCTRL[DSMC] = 0x3.
11. Configure ARM register SCR[SLEEPDEEP] = 0x1.
12. Execute instruction WFI to enter low power mode.
13. The chip immediately shuts off core and system clocks to the Arm core.
14. The chip polls the stop-acknowledge indications from supporting peripherals and the flash memory controller for indications that system clocks, bus clock, and/or flash memory clock need to be left enabled to complete a previously initiated operation, effectively stalling entry to the targeted low power mode. When all acknowledges are detected, system clock, bus clock and flash memory clock are turned off simultaneously.
15. SRMC requests PMU to generate isolation signal to core domain.
16. Chip enters STANDBY mode.
17. RTC, PMU and some GPIO will wake up STANDBY mode.
18. A wakeup reset sequence will start then, all power and clock configure change to default setting.
19. Any resets also exit STANDBY mode to execute reset sequence immediately, all power

and clock configure change to default setting.

20. After exiting from STANDBY mode, ISO_CLR in PMU_ISO_CLR register will be asserted to 1'b1. It will isolate the IO pads and ADC/CMP/SPLL modules. PMU_ISO_CLR[ISO_CLR] needs to W1C before any option after exiting from STANDBY mode.

NOTE

The following table show different condition enter the STANDBY mode.

Table 26-4 Phenomenon of entering STANDBY mode

Condition	Phenomena
Correct operation to enter STANDBY.	CPU Standby, device power consumption reduction. After the wake-up signal, device reset, and set SRMC_SRS[WAKEUP].
The NVIC interrupt flag is not cleared, execute STANDBY process.	CPU halt, device power consumption unchanged. After the interrupt signal, the program can continue to execute.
The NVIC interrupt flag is not cleared, wake-up source not configure interrupt, execute STANDBY process.	CPU halt, device power consumption unchanged and device can't wake-up. Users must avoid this situation when enter STANDBY mode.
Device attempts to enter the STANDBY but not all module acknowledge STANDBY request.	For SRMC_PMCTRL[DSMACKTMO] is non-zero, the device will reset and set SRMC_SRS[SACKERR] when it is timeout. For SRMC_PMCTRL [DSMACKTMO] is zero, the device will keep waiting until the module acknowledge.

26.4 Reset and interrupt

If the corresponding interrupt enable bit in SRMC_SRIE is set, when a reset signal is generated, SRMC will trigger the SRMC interrupt. After delaying for certain LPO clock cycles, which is decided by SRMC_SRIE[DELAY], the system will be reset. Due to the flag bits in SRS and SSRS which are set only after reset happened, the SRMC interrupt handler only knows that a reset will happen soon and necessary operations need to be completed, without information of the reset cause. Users can get the last reset cause from SRS after reset.

NOTE

During the delay period before reset, the SRMC interrupt will be triggered continuously. Users need to add the necessary processing logic into SRMC's callback function.

Chapter 27

Flash

27.1 Introduction

The flash module includes the following accessible memory regions:

- Flash0: Program flash memory for vector space and code store
- Flash0 IFR: Separate flash memory for parameter store
- Flash1: Program/Data Flash memory for code store or data store with software emulation

Flash memory is ideal for single-supply applications, for it permits in-the-field erase and reprogramming operations without the need for any external high voltage power sources.

The flash module includes a command controller which is used to execute commands to modify flash memory contents. An erased bit reads '1' and a programmed bit reads '0'. The programming operation is unidirectional, which means it can only move bits from the '1' state (erased) to the '0' state (programmed). Only the erase operation restores bits from '0' to '1'. Bits cannot be programmed from a '0' to a '1'.

NOTE

A flash memory location must be in the erased state before being programmed. Re-programming the programmed location will cause ECC error and will get the wrong data reading.

27.1.1 Glossary

- Command write sequence — A series of MCU writes to the flash controller register group and FLASH_FSTAT[CCIF] that initiates and controls the execution of flash algorithms that are built into the flash module.
- Endurance — The number of times that an aligned flash or IFR phrase can be erased and reprogrammed.
- Flash block — A macro within the flash module that provides the nonvolatile memory storage with an aligned block address being the first address in the block.
- Flash module — All flash blocks plus a flash controller providing command control and an interface to MCU buses.
- Flash page — 128 bytes of flash main memory with an aligned page having byte-address[6:0] = 00h.
- Flash phrase — 16 bytes of flash main memory with an aligned phrase having byte-address[3:0] = 0000b. It represents the smallest portion of the flash memory that can be programmed in one operation.
- Flash sector — 8192 bytes of flash main memory with an aligned sector having byte-address[12:0] = 0000h. It represents the smallest portion of the flash memory that can be erased in one operation.
- IFR — Information flash region separate from the main flash memory array. Each flash block has 32k bytes of user IFR space. Sometimes it is referred to as IFR0.
- IFR page — 128 bytes of IFR space with an aligned IFR page having byte-address[6:0] = 00h.
- IFR phrase — 16 bytes of IFR with an aligned IFR phrase having byte-address[3:0] = 0000b. It represents the smallest portion of IFR space that can be programmed in one operation.
- IFR sector — 8192 bytes of IFR space with an aligned IFR sector having byte-address[12:0] = 0000h. It represents the smallest portion IFR space that can be erased.
- MISR — Multiple-input signature register used to generate a signature based on flash memory contents read.
- NVM — Nonvolatile memory. The flash block is an NVM using NOR-type flash memory technology.

- Program flash — Program flash memory provides nonvolatile storage for vectors and code store. It is also named Flash0 in this chip.
- Data flash — Data flash memory in this project is the same as program flash, and it can provide nonvolatile storage for code or data store. If it is used to store data like EEPROM, software emulation driver is required. It is also named Flash1 in this chip.
- Retention — The length of time that data (erased or programmed) can be kept in the NVM without experiencing errors upon readout.
- RWW — Read-While-Write. If only one flash block available, RWW is not supported.

27.1.2 Features

The flash memory has the following key features:

- Sector size of 8192 bytes
- Single-bit error correction and double-bit error detection for each flash or IFR phrase
- Signature generation for flash contents
- Command-based flash program, erase, and verify operations
- Internal high-voltage supply generator for the flash program and erase operations
- Optional interrupt generation upon flash command completion
- Optional interrupt generation upon single-bit or double-bit error detection during a flash read
- ECC injection function for specified address reading
- Bus error generation for double-bit error on specified address range

NOTE

Flash function clock should be set to 8M. In STOP mode, function clock must select SLOW_CLK as clock source. After MCU reset, FIRC64M, OSC40M, SPILL and SLOW_CLK can be selected.

NOTE

The Flash ECC algorithm does not guarantee the detection of more than two bits error. Multi-bit error detection mentioned in this chapter means double-bit error or more than two bits error has been detected.

27.1.3 Block diagram

The following figure is a block diagram of the flash module.

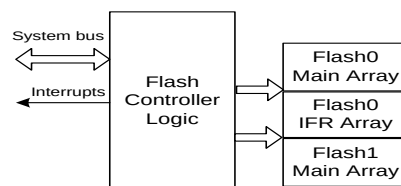


Figure 27-1 Flash block diagram

27.2 Flash memory

Flash0, Flash0 IFR, Flash1 spaces are assigned as below:

- Flash0: 0x00000000 - 0x001FFFFFFF
- Flash0 IFR: 0x02000000 - 0x02007FFF
- Flash1: 0x01000000 - 0x0101FFFF

The access authority among Flash0, Flash0 IFR, Flash1 is shown as below:

Table 27-1 Access authority

	Flash0 reading	Flash0 command executing	Flash0 IFR reading	Flash0 IFR command executing	Flash1 reading	Flash1 command executing
Flash0 reading	--	x	√	x	√	√
Flash0 command executing	x	--	x	x	√	x
Flash0 IFR reading	√	x	--	x	√	√
Flash0 IFR command executing	x	x	x	--	√	x
Flash1 reading	√	√	√	√	--	x
Flash1 command executing	√	x	√	x	x	--

Flash0 contains IFR space separate from the Flash0 main flash memory. Flash0 IFR space is memory mapped (the base address of IFR is 0x02000000). Once programmed, IFR phrases can be reprogrammed after a successful erase using the command operation.

NOTE

The Flash0 IFR space will be erased by running the “erase all” flash command.

Table 27-2 Flash0 IFR map

	Address	IFR Field Size (Bytes)
Flash0 IFR Sector 0	0x02000000-0x02001FFF	8,192
Flash0 IFR Sector 1	0x02002000-0x02003FFF	8,192
Flash0 IFR Sector 2	0x02004000-0x02005FFF	8,192
Flash0 IFR Sector 3	0x02006000-0x02007FFF	8,192

27.3 Memory map and register definition

This section provides the memory map and register definitions of the flash module.

27.3.1 Memory map

Flash controller base address: 0x40020000

Table 27-3 Flash controller memory map

Offset	Register	Width (In bits)	Reset Value
0x0	Flash Status Register (FLASH_FSTAT)	32	0x00000080
0x4	Flash Configuration Register (FLASH_FCNFG)	32	0x00000000
0x8	Flash Control Register (FLASH_FCTRL)	32	0x00000001
0x10	Flash Command Register (FLASH_FCMD)	32	0x00000000
0x18	Flash Address Register (FLASH_FADDR)	32	0x00000000
0x1C	Flash End Address Register (FLASH_FADDR_END)	32	0x00000000
0x20	Flash Data0 Register (FLASH_FDATA0)	32	0x00000000

Offset	Register	Width (In bits)	Reset Value
0x24	Flash Data1 Register (FLASH_FDATA1)	32	0x00000000
0x28	Flash Data2 Register (FLASH_FDATA2)	32	0x00000000
0x2C	Flash Data3 Register (FLASH_FDATA3)	32	0x00000000
0x30	Flash Bus Error Ignored Start Address Register (FLASH_BED_ADDR_STRT)	32	0x00000000
0x34	Flash Bus Error Ignored End Phrase Address Register (FLASH_BED_ADDR_END)	32	0xFFFFFFFF0
0x38	Flash ECC Injection Address Register (FLASH_ECC_IN_ADDR)	32	0x00000000
0x3C	Flash ECC Injection Control Register (FLASH_ECC_IN_CTRL)	32	0x80000000
0x40	Flash Multi-bit Error State Register (FLASH_MBE_STATE)	32	0x00000000
0x44	Flash Single-bit Error State Register (FLASH_SBE_STATE)	32	0x00000000

27.3.2 Flash Status Register (FLASH_FSTAT)

27.3.2.1 Offset

Register	Offset
FLASH_FSTAT	0x0

27.3.2.2 Function

The FSTAT register reports the operational status of the flash module.

27.3.2.3 Fields

Field	Access	Description	Reset Value
0 FAIL	RO	The FAIL flag is set if an error is detected during execution of a flash command or during flash initialization. The FAIL flag is cleared by a command request or erase all request from debug port with CCIF set if the FAIL flag is not set by flash initialization. 0b: Error not detected 1b: Error detected	0x0
1 RSVD_1	RO	Reserved	0x0
2 CMDABT	RW1C	The CMDABT flag indicates FCTRL[ABTREQ] was set during a command operation while CCIF was clear. The CMDABT flag is cleared by writing a 1 to CMDABT while CCIF is set and ABTREQ is clear. 0b: No command abort detected 1b: Command abort detected	0x0

Field	Access	Description	Reset Value
4:3 RSVD_4_3	RO	Reserved	0x0
5 ACCERR	RW1C	The ACCERR flag indicates an illegal attempt was made to launch a flash command. The ACCERR flag is cleared by writing a 1 to ACCERR while CCIF is set. 0b: No access error detected 1b: Access error detected	0x0
6 PREABT	RW1C	The PREABT flag indicates whether a request of FCTLR[ABTREQ] to abort a command writing before command launch has been granted. The PREABT flag is cleared by writing a 1 to PREABT while CCIF is set or clear and ABTREQ is clear. 0b: No command abort request generated before command launch 1b: Command abort request generated before command launch	0x0
7 CCIF	RW1C	The CCIF flag indicates that a flash command has completed. For write operations: The CCIF flag is cleared by writing a 1 to CCIF to launch a flash command. The CCIF flag stays low until the command. Writing 0 has no effect completed. For read operations: 0b: Flash command, initialization, or power mode recovery in progress 1b: Flash command, initialization, or power mode recovery has completed	0x1
10:8 RSVD_10_8	RO	Reserved	0x0
11 CMDWRF	RO	The CMDWRF bit indicates Flash command registers have been written for a new command. It is set by writing FCMD/FADDR/FDATA registers or command launching. And it is cleared after command execution. 0b: No Flash command prepared or executed 1b: A Flash command is prepared or executed	0x0
15:12 RSVD_15_12	RO	Reserved	0x0
16 DFDIF	RW1C	The DFDIF flag indicates an uncorrectable ECC fault was detected during a valid flash read. The DFDIF flag is cleared by writing a 1 to DFDIF. Writing a 0 to DFDIF has no effect. 0b: Double-bit fault not detected during a valid flash read access 1b: Double-bit fault detected during a valid flash read access	0x0
17 SFDIF	RW1C	The SFDIF flag indicates a correctable ECC fault was detected during a valid flash read. The SFDIF flag is cleared by writing a 1 to SFDIF. Writing a 0 to SFDIF has no effect. 0b: Single-bit fault not detected during a valid flash read access 1b: Single-bit fault detected during a valid flash read access	0x0

Field	Access	Description	Reset Value
29:18 RSVD_29_18	RO	Reserved	0x0
30 DBG_ERSALL_DIS	RO	0b: Debug erase all request is allowed to erase flash. 1b: Debug erase all request cannot erase flash (program 0x3EE2 to 0x02000000 high 16bit will disable debug erase flash function after any reset generation).	0x0
31 FSEC	RO	0b: Flash is unsecure, Debug Port is allowed to read AHB-AP memory-map addresses. 1b: Flash is secure (program 0xF5EC to 0x02000000 low 16 bits will enter security mode after any reset generation), and Debug Port is prevented to read any AHB-AP memory-map addresses any more.	0x0

27.3.3 Flash Configuration Register (FLASH_FCNFG)

27.3.3.1 Offset

Register	Offset
FLASH_FCNFG	0x4

27.3.3.2 Function

This register provides information on the current functional state of the flash module

27.3.3.3 Fields

Field	Access	Description	Reset Value
0 BK_QUERY	RW	This bit is used for backdoor query. Writing 1 to this bit will generate a query after writing backdoor keys to FDATA0-2 (96bit keys) with FSTAT[CCIF]=1. 0b: No query to backdoor. 1b: A query to backdoor, return to 0 automatically after current query.	0x0
6:1 RSVD_6_1	RO	Reserved	0x0
7 CCIE	RW	The CCIE bit controls interrupt generation when a flash command completes. 0b: Command complete interrupt disabled. 1b: An interrupt request is generated whenever the FSTAT[CCIF] flag is set.	0x0
8 DBG_ERS_FLG	RO	This bit indicates that a debug request has been received by the command controller to execute the Erase All operation. 0b: No request or request complete. 1b: Request to run the Erase All operation.	0x0
15:9 RSVD_15_9	RO	Reserved	0x0
16 DFDIE	RW	The DFDIE bit controls interrupt generation when an uncorrectable ECC fault is detected during a valid flash read access from the platform flash controller. 0b: Double bit fault detect interrupt disabled.	0x0

Field	Access	Description	Reset Value
		1b: An interrupt request is generated whenever the FSTAT[DFDIF] flag is set.	
17 SFDIE	RW	The SFDIE bit controls interrupt generation when a correctable ECC fault is detected during a valid flash read access from the platform flash controller. 0b: Single-bit fault detect interrupt disabled. 1b: An interrupt request is generated whenever the FSTAT[SFDIF] flag is set.	0x0
31:18 RSVD_31_18	RO	Reserved	0x0

27.3.4 Flash Control Register (FLASH_FCTRL)

27.3.4.1 Offset

Register	Offset
FLASH_FCTRL	0x8

27.3.4.2 Function

The flash control register controls activity associated with flash memory reads and command operations.

27.3.4.3 Fields

Field	Access	Description	Reset Value
3:0 RWSC	RW	These bits control the number of wait-states added to account for the ratio of system clock period to read flash addresses 0h: no additional wait-states are added (single cycle access) 1h: 1 additional wait-state is added. 2h: 2 additional wait-states are added. ... Fh: 15 additional wait-states are added. The recommended setting of RWSC is as below: 0h: CORE_CLK <= 32M 1h: CORE_CLK <= 64M 2h: CORE_CLK <= 96M 3h: CORE_CLK <= 128M 4h: CORE_CLK <= 160M Note: Before switching core clock source or core clock frequency, users should set RWSC to an appropriate value.	0x1
15:4 RSVD_15_4	RO	Reserved	0x0
16 FECCF	RW	The FECCF bit enables the user to emulate the setting of the FSTAT[DFDIF] flag to check the associated interrupt routine. This bit is cleared by writing a 0 to it. 0b: Force ECC fault function is disabled. 1b: Force ECC fault function is enabled, any real read to Flash array will set FSTAT[DFDIF] flag.	0x0

Field	Access	Description	Reset Value
23:17 RSVD_23_17	RO	Reserved	0x0
24 ABTREQ	RW	This bit can request to abort a command write sequence. 0b: No request to abort a command write sequence. 1b: Request to abort a command write sequence.	0x0
31:25 RSVD_31_25	RO	Reserved	0x0

27.3.5 Flash Command Register (FLASH_FCMD)

27.3.5.1 Offset

Register	Offset
FLASH_FCMD	0x10

27.3.5.2 Fields

Field	Access	Description	Reset Value
7:0 CMDCODE	RW	Flash command code for programming, erasing, and so on	0x0
31:8 RSVD_31_8	RO	Reserved	0x0

27.3.6 Flash Address Register (FLASH_FADDR)

27.3.6.1 Offset

Register	Offset
FLASH_FADDR	0x18

27.3.6.2 Fields

Field	Access	Description	Reset Value
31:0 CMDADDR	RW	Flash start address for Flash commands	0x0

27.3.7 Flash End Address Register (FLASH_FADDR_END)

27.3.7.1 Offset

Register	Offset
FLASH_FADDR_END	0x1C

27.3.7.2 Fields

Field	Access	Description	Reset Value
31:0 CMDADDRE	RW	Flash end address for Flash commands	0x0

27.3.8 Flash Data0 Register (FLASH_FDATA0)

27.3.8.1 Offset

Register	Offset
FLASH_FDATA0	0x20

27.3.8.2 Fields

Field	Access	Description	Reset Value
31:0 CMDDATA0	RW	Flash data0 register for commands: data[31:0]	0x0

27.3.9 Flash Data1 Register (FLASH_FDATA1)

27.3.9.1 Offset

Register	Offset
FLASH_FDATA1	0x24

27.3.9.2 Fields

Field	Access	Description	Reset Value
31:0 CMDDATA1	RW	Flash data1 register for commands: data[63:32]	0x0

27.3.10 Flash Data2 Register (FLASH_FDATA2)

27.3.10.1 Offset

Register	Offset
FLASH_FDATA2	0x28

27.3.10.2 Fields

Field	Access	Description	Reset Value
31:0 CMDDATA2	RW	Flash data2 register for commands: data[95:64]	0x0

27.3.11 Flash Data3 Register (FLASH_FDATA3)

27.3.11.1 Offset

Register	Offset
FLASH_FDATA3	0x2C

27.3.11.2 Fields

Field	Access	Description	Reset Value

Field	Access	Description	Reset Value
31:0 CMDDATA3	RW	Flash data3 register for commands: data[127:96]	0x0

27.3.12 Flash Bus Error Ignored Start Address Register (FLASH_BED_ADDR_STRT)

27.3.12.1 Offset

Register	Offset
FLASH_BED_ADDR_STRT	0x30

27.3.12.2 Fields

Field	Access	Description	Reset Value
3:0 RSVD_3_0	RO	Reserved	0x0
31:4 BED_ADDR_STRT	RW	It defines the flash start address of the flash region to ignore bus error if Flash ECC multi-bit error happens during flash reading. It is corresponding to bit 31-bit 4 of the start address. The bit 3-bit 0 of the address is always 0.	0x0

27.3.13 Flash Bus Error Ignored End Phrase Address Register (FLASH_BED_ADDR_END)

27.3.13.1 Offset

Register	Offset
FLASH_BED_ADDR_END	0x34

27.3.13.2 Fields

Field	Access	Description	Reset Value
3:0 RSVD_3_0	RO	Reserved	0x0
31:4 BED_ADDR_END	RW	It defines the start address of the end flash phrase in the flash region to ignore bus error if Flash ECC multi-bit error happens during flash reading. It is corresponding to bit 31-bit 4 of the start address of the end flash phrase. The bit 3-bit 0 of the address is always 0.	0xFFFFFFFF

27.3.14 Flash ECC Injection Address Register (FLASH_ECC_IN_ADDR)

27.3.14.1 Offset

Register	Offset

Register	Offset
FLASH_ECC_IN_ADDR	0x38

27.3.14.2 Fields

Field	Access	Description	Reset Value
3:0 RSVD_3_0	RO	Reserved	0x0
31:4 ECC_IN_ADDR	RW	ECC injection address	0x0

27.3.15 Flash ECC Injection Control Register (FLASH_ECC_IN_CTRL)

27.3.15.1 Offset

Register	Offset
FLASH_ECC_IN_CTRL	0x3C

27.3.15.2 Fields

Field	Access	Description	Reset Value
8:0 ECC_9BIT	RW	It is used for error injection to 9-bit ECC data part of a flash phrase. Each bit is corresponding to a bit in the ECC part of the flash phrase. FDATA0-3 is used for 128-bit flash data error injection. 0b: No injection on this bit 1b: This bit will be inverted on reading	0x0
9 ECC_IN_EN	RW	0b: Disable ECC injection 1b: Enable ECC injection	0x0
30:10 RSVD_30_10	RO	Reserved	0x0
31 LOCK	RW	Write lock bit for FLASH_ECC_IN_ADDR and FLASH_ECC_IN_CTRL. 0b: Not locked. FLASH_ECC_IN_ADDR and FLASH_ECC_IN_CTRL can be written 1b: FLASH_ECC_IN_ADDR and FLASH_ECC_IN_CTRL are locked to write, reading is available. It can be cleared by writing 0x5C to FLASH_ECC_IN_CTRL[31:24]	0x1

27.3.16 Flash Multi-bit Error State Register (FLASH_MBE_STATE)

27.3.16.1 Offset

Register	Offset
FLASH_MBE_STATE	0x40

27.3.16.2 Fields

Field	Access	Description	Reset Value
0 MBE_FLAG	R/W1C	Multi-bit error flag. Writing 1 to this bit clears bit0 and bit1 of this register 0b: No multi-bit error generated 1b: Multi-bit error has generated	0x0
1 MBE_OVERRUN	RO	Multi-bit error overrun 0b: Multi-bit error is not overrun 1b: Multi-bit error has been overrun	0x0
3:2 RSVD_3_2	RO	Reserved	0x0
31:4 MBE_ADDR	RO	Multi-bit error address	0x0

27.3.17 Flash Single-bit Error State Register (FLASH_SBE_STATE)

27.3.17.1 Offset

Register	Offset
FLASH_SBE_STATE	0x44

27.3.17.2 Fields

Field	Access	Description	Reset Value
0 SBE_FLAG	R/W1C	Single-bit error flag. Writing 1 to this bit clears bit0 and bit1 of this register 0b: No single-bit error generated 1b: Single-bit error has generated	0x0
1 SBE_OVERRUN	RO	Single-bit error overrun 0b: Single-bit error is not overrun 1b: Single-bit error has been overrun	0x0
3:2 RSVD_3_2	RO	Reserved	0x0
31:4 SBE_ADDR	RO	Single-bit error address	0x0

27.4 Function description

This section provides the functional information of the flash module.

27.4.1 Interrupts

The flash module can generate interrupt requests to the MCU when a variety of flash module events occur. The following table details these interrupt events and their associated status and control bits.

Table 27-4 Flash module interrupt sources

Flash event	Readable status bit	Interrupt enable bit
Flash Command Complete	FLASH_FSTAT[CCIF]	FLASH_FCNFG[CCIE]
Flash 2-bit ECC Fault Detected	FLASH_FSTAT[DFDIF]	FLASH_FCNFG[DFDIE]

Flash event	Readable status bit	Interrupt enable bit
Flash 1-bit ECC Fault Detected	FLASH_FSTAT[SFDIF]	FLASH_FCNFG[SFDIE]

27.4.2 Bus error due to multi-bit ECC error

The flash controller can be configured to generate bus error to MCU if multi-bit ECC error is detected during flash reading. This feature is configured through FLASH_BED_ADDR_STRT and FLASH_BED_ADDR_END registers, which configures the memory region that does not generate bus error when multi-bit ECC error is detected. FLASH_BED_ADDR_STRT defines the start address of the region that does not generate bus error due to multi-bit error of flash reading. FLASH_BED_ADDR_END defines the end address (the start address of the end flash phrase) of the region that does not generate bus error due to multi-bit error of flash reading. The default value of FLASH_BED_ADDR_STRT[BED_ADDR_STRT] is 0x0. The default value of FLASH_BED_ADDR_END[BED_ADDR_END] is 0xFFFFFFFF, which means the start address of the end phrase of the region with bus error ignored is 0xFFFFFFFF0. The end address of the region with bus error ignored is 0xFFFFFFFF. So bus error is not generated from 0x0 - 0xFFFFFFFF by default.

27.4.3 Flash fixed clock

The Flash fixed clock is used to generate right timing sequence of Flash commands and recover Flash module from STOP mode. The clock source of Flash fixed clock is selectable from system clock gating module (for example PARCC), and its frequency must be fixed or divided to 8M Hz. But there are different limits of Flash fixed clock for the two functions. The first one is to generate timing sequence of Flash commands. Flash fixed clock is asynchronous and it can select any clock source and divide the clock source to 8M Hz. The second one is to recover Flash from STOP mode. The source of Flash fixed clock has to be synchronous with Flash read clock (for example AHB interface clock).

Table 27-5 The limit of Flash fixed clock

Usage of Flash fixed clock	Recommended clock source	Target frequency
Timing sequence generation of Flash commands	FIRC64M, SPLL, OSC40M, Slow Clock	8M
Recovery from STOP mode	Slow Clock	8M

NOTE

Flash functional clock must be enabled in PARCC before MCU enters into STOP mode.

27.4.4 Security and backdoor

The flash module can implement the functions about chip security and backdoor.

Flash and MCU will enter security mode after any reset with Flash0 IFR data[15:0] of address 0x02000000 is programmed to 0xF5EC, and FLASH_FSTAT[FSEC] bit indicates the status of security. Writing 1 to ADM-AP Control[0] through debug port will implement mass erase of MCU and make it un-secure.

Debug mass erase operation of writing 1 to ADM-AP Control[0] will be prohibited after any reset with Flash0 IFR data[31:16] of address 0x02000000 is programmed to 0x3EE2. It is the ultimate security status of MCU, and needs to use with caution.

96-bit backdoor key is setup with LSB-aligned in Flash0 IFR data of address 0x02000004, 0x02000008 and 0x0200000C. To relieve flash security state, writing 1 to FLASH_FCNFG[0] is required with right backdoor key in FLASH_FDATA0, FLASH_FDATA1 and FLASH_FDATA2. The corresponding backdoor key is as below.

Table 27-6 Backdoor key

Backdoor key	Key Store Address	Key query

Backdoor key	Key Store Address	Key query
Key[95:64]	0x0200000C	FLASH_FDATA2
Key[63:32]	0x02000008	FLASH_FDATA1
Key[31:0]	0x02000004	FLASH_FDATA0

27.4.5 ECC injection

The flash module can implement ECC injection function for specified address reading. The operation flow of ECC injection is as below:

1. Unlock FLASH_ECC_IN_ADDR and FLASH_ECC_IN_CTRL by writing 0x5C to FLASH_ECC_IN_CTRL[31:24]
2. Configure ECC injection address by writing FLASH_ECC_IN_ADDR[31:4]
3. Configure ECC injection 128-bit data on FLASH_FDATA0-FLASH_FDATA3 and 9-bit ECC value on FLASH_ECC_IN_CTRL[8:0], and the 137-bit read data from flash arrays will make XOR operation with 137-bit ECC injection data before ECC check circuit.
4. Enable ECC injection function by writing 1 to FLASH_ECC_IN_CTRL[9]
5. Check ECC event status on FLASH_MBE_STATE and FLASH_SBE_STATE, while bus error and interrupt can generate with right configuration on ECC double bit error.

27.4.6 Flash command operations

Flash command operations are typically used to modify the flash memory contents.

The next sections describe the command write sequence used to set flash command parameters and the launch of execution of the available flash command.

27.4.6.1 Command write sequence

Flash commands are accomplished using a command write sequence.

- Flash controller monitors the command write sequence, performs various checks on the command content, and continues with command execution if all requirements are fulfilled.
- Before a command is launched, the ACCERR and CMDABT flags in the FLASH_FSTAT register must be '0' and the CCIF flag must be '1' to verify that any previous command has been completed. The FAIL flag in the FLASH_FSTAT register is cleared by the command controller when a command is launched.
- If CCIF is clear, the previous command execution is still active, a new command write sequence cannot be started, and all writes to the flash command registers are ignored.

27.4.6.2 Load the flash command registers

The user must load the flash command registers with all parameters required by the specific flash command.

The contents of any flash command register that is not used by the specific flash command are ignored.

The individual registers that make up the flash register data set can be written in any order.

27.4.6.2.1 Launch the command by clearing CCIF

Once all relevant command parameters have been loaded, the user launches the command by

clearing the FLASH_FSTAT[CCIF] bit by writing a '1' to it.

The CCIF flag remains clear until the flash command is completed.

The FLASH_FSTAT register contains a blocking mechanism that prevents a new flash command from launching (unable to clear CCIF) if the previous command resulted in the setting of an access error (ACCERR) or active command abort flag (CMDABT).

In the error scenarios, two writes to FSTAT are required to initiate the next command. The first write is for clearing these flags, the second write is for clearing CCIF.

27.4.6.2.2 Command execution and error reporting

The flash command processing contains the following steps:

1. The command controller performs a series of parameter checks, if applicable, which are unique to each command.
 - If the parameter check fails, the FLASH_FSTAT[ACCERR] flag is set. ACCERR reports invalid instruction codes or options, out of bounds or misaligned addresses, or missing flash internal parameters.
 - In the case Flash parameters were not loaded correctly during initialization, the ACCERR and FLASH_FSTAT[FAIL] flags are set.
 - Command processing never proceeds to execution when the internal parameter check fails. Instead, command processing is terminated after setting the FLASH_FSTAT[CCIF] flag.
2. If the parameter check passes, the command proceeds to execution. However, run-time errors, such as failure to verify, may occur during the execution phase and are reported in the FAIL flag.
3. Report back the command execution results, if applicable, to the user via the flash register.
4. The command controller sets FLASH_FSTAT[CCIF], signifying that the command execution has been completed.

NOTE

A flash command may have access errors and run-time errors, but the run-time errors are not seen until all access errors have been corrected.

If FLASH_FCTRL[ABTREQ] sets during command execution, the operation will abort with the FLASH_FSTAT[CMDABT] flag set.

27.4.6.3 Flash commands

The following table summarizes the function of all the flash commands. If any column is marked with an 'X', the flash command is relevant to the particular memory resource.

Table 27-7 Flash command list

FCMD	Command	Program flash	IFR	Function
00h	(RD1ALL)	x	x	Verify that all flash and IFR space is erased
01h	(RD1BLK)	x		Verify that a flash block is erased
02h	(RD1SCR)	x		Verify that a flash sector is erased
03h	(RD1PG)	x		Verify that a flash page is erased
04h	(RD1PHR)	x		Verify that a flash phrase is erased
05h	(RDMISR)	x		Generate MISR signature for range of flash pages

FCMD	Command	Program flash	IFR	Function
12h	(RD1ISCR)		x	Verify that an IFR sector is erased
13h	(RD1IPG)		x	Verify that an IFR page is erased
14h	(RD1IPHR)		x	Verify that an IFR phrase is erased
15h	(RDIMISR)		x	Generate MISR signature for range of IFR pages
24h	(PGMPHR)	x	x	Program data to a flash or IFR phrase
40h	(ERSALL)	x	x	Erase all flash and IFR space
42h	(ERSSCR)	x	x	Erase a flash sector

27.4.7 Flash command descriptions

This section describes all flash commands that can be launched by a valid command write sequence. The flash module sets the FLASH_FSTAT[ACCERR] bit and aborts the command execution if any of the following illegal conditions occur:

- There is an unrecognized command code in the FLASH_FCMD[CMDCODE] field.
- There is an error in flash field for the specific command.

The ACCERR and CMDABT flags in the FLASH_FSTAT register should be cleared before starting a command write sequence. The user cannot launch any new command while these flags are set.

Do not attempt to read a flash block while a flash command is running (CCIF = 0) on that same block. The flash module may return a transfer error to the flash controller, resulting in a bus fault. If more than one flash block is available, the code running in one flash block to execute the flash command of another flash block is allowed and proposed.

27.4.7.1 Command RD1ALL

The command checks if all the flash and IFR space are in the erased state.

Table 27-8 Command register requirements

Register name	Register contents
FLASH_FCMD	(RD1ALL)

Upon clearing CCIF to launch the command, the command controller:

- Verifies that all the flash and IFR space are in the erased state.

If all the flash and IFR space are not in the erased state, the FLASH_FSTAT[FAIL] flag is set. The CCIF flag sets after the operation completes.

Table 27-9 Command error handling

Error condition	Error bit
Flash parameters not loaded correctly during initialization	FLASH_FSTAT[ACCERR]
Operation is aborted	FLASH_FSTAT[CMDABT]
Flash parameters not loaded correctly during initialization	FLASH_FSTAT[FAIL]
Verify fails	FLASH_FSTAT[FAIL]

27.4.7.2 Command RD1BLK

The command checks if a flash block is in the erased state.

Table 27-10 Command register requirements

Register name	Register contents
FLASH_FCMD	(RD1BLK)
FLASH_FADDR	Block address

Upon clearing CCIF to launch the command, the command controller:

- Verifies that the selected flash block is in the erased state.

If the selected flash block is not in the erased state, the FLASH_FSTAT[FAIL] flag is set. The CCIF flag sets after the operation completes.

Note that RD1BLK command cannot verify the IFR region.

Table 27-11 Command error handling

Error condition	Error bit
Flash parameters not loaded correctly during initialization	FLASH_FSTAT[ACCERR]
Address is not valid	FLASH_FSTAT[ACCERR]
Operation is aborted	FLASH_FSTAT[CMDABT]
Flash parameters not loaded correctly during initialization	FLASH_FSTAT[FAIL]
Verify fails	FLASH_FSTAT[FAIL]

27.4.7.3 Command RD1SCR

The command checks if a flash sector is in the erased state.

Table 27-12 Command register requirements

Register name	Register contents
FLASH_FCMD	(RD1SCR)
FLASH_FADDR	Sector address

Upon clearing CCIF to launch the command, the command controller:

- Verifies that the selected flash sector is in the erased state.

If the selected flash sector is not in the erased state, the FLASH_FSTAT[FAIL] flag is set. The CCIF flag sets after the operation completes.

Table 27-13 Command error handling

Error condition	Error bit
Flash parameters not loaded correctly during initialization	FLASH_FSTAT[ACCERR]
Address is not valid	FLASH_FSTAT[ACCERR]
Address is not sector aligned	FLASH_FSTAT[ACCERR]
Operation is aborted	FLASH_FSTAT[CMDABT]
Flash parameters not loaded correctly during initialization	FLASH_FSTAT[FAIL]
Verify fails	FLASH_FSTAT[FAIL]

27.4.7.4 Command RD1PG

The command checks if a flash page is in the erased state.

Table 27-14 Command register requirements

Register name	Register contents

Register name	Register contents
FLASH_FCMD	(RD1PG)
FFLASH_ADDR	Page address

Upon clearing CCIF to launch the command, the command controller:

- Verifies that the selected flash page is in the erased state.

If the selected flash page is not in the erased state, the FLASH_FSTAT[FAIL] flag is set. The CCIF flag sets after the operation completes.

Table 27-15 Command error handling

Error condition	Error bit
Flash parameters not loaded correctly during initialization	FLASH_FSTAT[ACCERR]
Address is not valid	FLASH_FSTAT[ACCERR]
Address is not page aligned	FLASH_FSTAT[ACCERR]
Operation is aborted	FLASH_FSTAT[CMDABT]
Flash parameters not loaded correctly during initialization	FLASH_FSTAT[FAIL]
Verify fails	FLASH_FSTAT[FAIL]

27.4.7.5 Command RD1PHR

The command checks if a flash phrase is in the erased state.

Table 27-16 Command register requirements

Register name	Register contents
FLASH_FCMD	(RD1PHR)
FLASH_FADDR	Phrase address

Upon clearing CCIF to launch the command, the command controller:

- Verifies that the selected flash phrase is in the erased state.

If the selected flash phrase is not in the erased state, the FLASH_FSTAT[FAIL] flag is set. The CCIF flag sets after the operation completes.

Table 27-17 Command error handling

Error condition	Error bit
Flash parameters not loaded correctly during initialization	FLASH_FSTAT[ACCERR]
Address is not valid	FLASH_FSTAT[ACCERR]
Address is not phrase aligned	FLASH_FSTAT[ACCERR]
Operation is aborted	FLASH_FSTAT[CMDABT]
Flash parameters not loaded correctly during initialization	FLASH_FSTAT[FAIL]
Verify fails	FLASH_FSTAT[FAIL]

27.4.7.6 Command RDMISR

The operation generates a signature based on the contents of the selected flash memory using an embedded MISR.

Table 27-18 Command register requirements

Register name	Register contents
FLASH_FCMD	(RDMISR) [7:0]
FLASH_FADDR	Starting flash page address
FLASH_FADDR_END	Ending flash phrase address
FLASH_FDATA0	Word 0 of MISR seed
FLASH_FDATA1	Word 1 of MISR seed
FLASH_FDATA2	Word 2 of MISR seed
FLASH_FDATA3	Word 3 of MISR seed
Returned values	
FLASH_FDATA0	Word 0 of MISR signature
FLASH_FDATA1	Word 1 of MISR signature
FLASH_FDATA2	Word 2 of MISR signature
FLASH_FDATA3	Word 3 of MISR signature

Upon clearing CCIF to launch the command, the command controller:

- Initializes the MISR with the seed provided
- Processes flash data starting with the starting flash page address provided
- Continues until the ending flash phrase address has been processed. The ending flash phrase must be the last phrase in a page.
- Returns the resulting signature into the Flash data register bank unless an uncorrectable ECC fault is detected

The CCIF flag sets after the operation completes. If the operation is aborted by setting FLASH_FCTRL[ABTREQ], a signature will not be returned if the abort occurs prior to the operation of writing the signature.

MISR polynomial is $X^{128} + X^{126} + X^{101} + X^{99} + 1$.

Table 27-19 Command error handling

Error condition	Error bit
Flash parameters not loaded correctly during initialization	FLASH_FSTAT[ACCERR]
Address is not valid	FLASH_FSTAT[ACCERR]
Starting address is not flash page aligned	FLASH_FSTAT[ACCERR]
Ending address is not flash phrase aligned	FLASH_FSTAT[ACCERR]
Ending address is not the last phrase in a page	FLASH_FSTAT[ACCERR]
Starting address is larger than the ending address	FLASH_FSTAT[ACCERR]
Requested range crosses a flash block boundary	FLASH_FSTAT[ACCERR]
Operation is aborted	FLASH_FSTAT[CMDABT]
Flash parameters not loaded correctly during initialization	FLASH_FSTAT[FAIL]
Uncorrectable ECC error is detected	FLASH_FSTAT[FAIL]

27.4.7.7 Command RD1ISCR

The command checks if an IFR sector is in the erased state.

Table 27-20 Command register requirements

Register name	Register contents

Register name	Register contents
FLASH_FCMD	(RD1ISCR)
FLASH_FADDR	IFR sector address

Upon clearing CCIF to launch the command, the command controller:

- Verifies that the selected IFR sector is in the erased state.

If the selected IFR sector is not in the erased state, the FLASH_FSTAT[FAIL] flag is set. The CCIF flag sets after the operation completes.

Table 27-21 Command error handling

Error condition	Error bit
Flash parameters not loaded correctly during initialization	FLASH_FSTAT[ACCERR]
Address is not valid	FLASH_FSTAT[ACCERR]
Address is not IFR sector aligned	FLASH_FSTAT[ACCERR]
Operation is aborted	FLASH_FSTAT[CMDABT]
Flash parameters not loaded correctly during initialization	FLASH_FSTAT[FAIL]
Verify fails	FLASH_FSTAT[FAIL]

27.4.7.8 Command RD1IPG

The command checks if an IFR page is in the erased state.

Table 27-22 Command register requirements

Register name	Register contents
FLASH_FCMD	(RD1IPG)
FLASH_FADDR	IFR page address

Upon clearing CCIF to launch the command, the command controller:

- Verifies that the selected IFR page is in the erased state.

If the selected IFR page is not in the erased state, the FLASH_FSTAT[FAIL] flag is set. The CCIF flag sets after the operation completes.

Table 27-23 Command error handling

Error condition	Error bit
Flash parameters not loaded correctly during initialization	FLASH_FSTAT[ACCERR]
Address is not valid	FLASH_FSTAT[ACCERR]
Address is not IFR page aligned	FLASH_FSTAT[ACCERR]
Operation is aborted	FLASH_FSTAT[CMDABT]
Flash parameters not loaded correctly during initialization	FLASH_FSTAT[FAIL]
Verify fails	FLASH_FSTAT[FAIL]

27.4.7.9 Command RD1IPHR

The command checks if an IFR phrase is in the erased state.

Table 27-24 Command register requirements

Register name	Register contents

Register name	Register contents
FLASH_FCMD	(RD1IPHR)
FLASH_FADDR	IFR phrase address

Upon clearing CCIF to launch the command, the command controller:

- Verifies that the selected IFR phrase is in the erased state.

If the selected IFR phrase is not in the erased state, the FLASH_FSTAT[FAIL] flag is set. The CCIF flag sets after the operation completes.

Table 27-25 Command error handling

Error condition	Error bit
Flash parameters not loaded correctly during initialization	FLASH_FSTAT[ACCERR]
Address is not valid	FLASH_FSTAT[ACCERR]
Address is not IFR phrase aligned	FLASH_FSTAT[ACCERR]
Operation is aborted	FLASH_FSTAT[CMDABT]
Flash parameters not loaded correctly during initialization	FLASH_FSTAT[FAIL]
Verify fails	FLASH_FSTAT[FAIL]

27.4.7.10 Command RDIMISR

The operation generates a signature based on the contents of the selected IFR space using an embedded MISR.

Table 27-26 Command register requirements

Register name	Register contents
FLASH_FCMD	(RDIMISR) [7:0]
FLASH_FADDR	Starting IFR page address
FLASH_FADDR_END	Ending IFR phrase address
FLASH_FDATA0	Word 0 of MISR seed
FLASH_FDATA1	Word 1 of MISR seed
FLASH_FDATA2	Word 2 of MISR seed
FLASH_FDATA3	Word 3 of MISR seed
Returned values	
FLASH_FDATA0	Word 0 of MISR signature
FLASH_FDATA1	Word 1 of MISR signature
FLASH_FDATA2	Word 2 of MISR signature
FLASH_FDATA3	Word 3 of MISR signature

Upon clearing CCIF to launch the command, the command controller:

- Initializes the MISR with the seed provided.
- Processes IFR data starting with the starting IFR page address provided.
- Continues until the ending IFR phrase has been processed. The ending IFR phrase must be the last phrase in a page.
- Returns the resulting signature into the Flash data register bank unless an uncorrectable ECC fault is detected.

The CCIF flag sets after the operation completes. If the operation is aborted by setting

FLASH_FCTRL[ABTREQ] and the abort occurs before the operation of writing the signature, the command controller will not return a signature.
MISR polynomial is $X^{128} + X^{126} + X^{101} + X^{99} + 1$.

Table 27-27 Command error handling

Error condition	Error bit
Flash parameters not loaded correctly during initialization	FLASH_FSTAT[ACCERR]
Address is not valid	FLASH_FSTAT[ACCERR]
Starting address is not IFR page aligned	FLASH_FSTAT[ACCERR]
Ending address is not IFR phrase aligned	FLASH_FSTAT[ACCERR]
Ending address is not the last phrase in a page	FLASH_FSTAT[ACCERR]
Starting address is larger than the ending address	FLASH_FSTAT[ACCERR]
Requested range crosses the IFR block boundary (module with dual flash blocks)	FLASH_FSTAT[ACCERR]
Operation is aborted	FLASH_FSTAT[CMDABT]
Flash parameters not loaded correctly during initialization	FLASH_FSTAT[FAIL]
Uncorrectable ECC error is detected	FLASH_FSTAT[FAIL]

27.4.7.11 Command PGMPHR

The operation programs 4 words of data to a previously erased flash or IFR phrase. The phrase must be in the erased state before being programmed.

Table 27-28 Command register requirements

Register name	Register contents
FLASH_FCMD	(PGMPHR) [7:0]
FLASH_FADDR	Starting phrase address
FLASH_FDATA0	Word 0
FLASH_FDATA1	Word 1
FLASH_FDATA2	Word 2
FLASH_FDATA3	Word 3

The CCIF flag sets after the operation completes. The operation can be aborted by setting FLASH_FCTRL[ABTREQ]. If the operation is aborted, the operation terminates with the FLASH_FSTAT[CMDABT] flag set. Once CCIF is set and the CMDABT flag is cleared, repeat the command with the same addresses and same data to complete the program operation without erasing the sector containing the targeted phrase.

Table 27-29 Command error handling

Error condition	Error bit
Flash parameters not loaded correctly during initialization	FLASH_FSTAT[ACCERR]
Address is not phrase aligned	FLASH_FSTAT[ACCERR]
Number of writes to flash or IFR space is more than expected	FLASH_FSTAT[ACCERR]
Operation is aborted	FLASH_FSTAT[CMDABT]
Flash parameters not loaded correctly during initialization	FLASH_FSTAT[FAIL]
Verify fails	FLASH_FSTAT[FAIL]

27.4.7.12 Command ERSALL

The operation erases all flash and IFR space.

Table 27-30 Command register requirements

Register name	Register contents
FLASH_FCMD	(ERSALL) [7:0]

The CCIF flag sets after the operation completes. If either the flash or IFR space is not in the erased state, FLASH_FSTAT[FAIL] is set.

Table 27-31 Command error handling

Error condition	Error bit
Flash parameters not loaded correctly during initialization	FLASH_FSTAT[ACCERR]
Operation is aborted	FLASH_FSTAT[CMDABT]
Flash parameters not loaded correctly during initialization	FLASH_FSTAT[FAIL]
Verify fails	FLASH_FSTAT[FAIL]

27.4.7.13 Command ERSSCR

The operation erases a flash or IFR sector to prepare the sector for programming.

Table 27-32 Command register requirements

Register name	Register contents
FLASH_FCMD	(ERSSCR) [7:0]
FLASH_FADDR	Address in the sector

If the selected flash or IFR sector is not in the erased state, FLASH_FSTAT[FAIL] is set. The CCIF flag sets after the operation completes. The operation can be aborted by setting FLASH_FCTRL[ABTREQ]. If the operation is aborted, the operation terminates by setting the FLASH_FSTAT[CMDABT]. Once CCIF is set, repeat the command to complete the erase operation.

Table 27-33 Command error handling

Error condition	Error bit
Flash parameters not loaded correctly during initialization	FLASH_FSTAT[ACCERR]
Address is not valid	FLASH_FSTAT[ACCERR]
Operation is aborted	FLASH_FSTAT[CMDABT]
Flash parameters not loaded correctly during initialization	FLASH_FSTAT[FAIL]
Verify fails	FLASH_FSTAT[FAIL]

27.5 Initialization application information

On each reset, the command controller initializes the following:

- FLASH_FCTRL[RWSC]
- Flash Parameters

The FLASH_FSTAT[CCIF] flag is cleared throughout the initialization flow. The flash module also holds off all access to flash registers and flash memory during initialization. Completion of the initialization flow is marked by setting FLASH_FSTAT[CCIF]. If a double-bit error is detected during initialization, the FLASH_FSTAT[FAIL] flag will be set.

The FAIL flag will also set if the flash parameters are not loaded correctly during initialization. In this case, attempts to launch a flash command will terminate by setting the ACCERR and FAIL flags.

If a reset occurs while any flash command is in progress, that flash command is immediately aborted. There is no guarantee on which state the flash memory is in and whether the IFR space is being programmed or erased. The command operations do not automatically resume after exiting reset.

Chapter 28

Register File (REGFILE)

28.1 Introduction

The register file is always located on domain. Registers in the register file block are all readable and writable and they can be used to store the pre-written data in STANDBY mode.

28.1.1 Features

- Contain 32-bit registers, the total number of which is 32
- Access through APB bus

28.2 Memory Map and Register Definition

28.2.1 Memory Map

REGFILE base address: 0x4007C000

Offset	Register	Width (In bits)	Reset Value
0x0+4*n (n=0~31)	REGFILE_REGn	32	0x00000000

28.2.2 REGFILE_REGn

28.2.2.1 Offset

Register	Offset
REGFILE_REGn	0x0+4*n(n=0~31)

28.2.2.2 Function

This register is used to store data that will not be lost during STANDBY mode.

28.2.2.3 Fields

Field	Access	Description	Reset Value
31:0 DATA	RW	Store data that the user needs	0x0

Chapter 29

Motor Control PWM (MCPWM)

29.1 Introduction

The purpose of this chapter is to define a Motor Control PWM (MCPWM).

29.1.1 Features

The MCPWM block provides four 16-bit counters to generate PWM output signals of 8 channels. The 8 channels are divided to 4 pairs. For each pair of channels, output of the two channels can be independent or combined as complementary. For each pair of channels, it can operate with global counter or independent counter.

In complementary mode, dead time delay can be added to ensure no one pair of output signals drives the active state at the same time. Deadtime delay value in each pair of channels is configurable independently.

MCPWM supports four fault inputs, fault 0 and fault 1 inputs are regarded as fault type A, fault 2 and fault 3 inputs are regarded as fault type B. When fault A/B input event is detected in fault input, MCPWM can force output channels to safe state to ensure motor control safety. The safe status of channels in fault A event and fault B event are configurable independently. The MCPWM supports the following key features:

- Time base:
 - Support multiple counter clock sources, core clock, function clock and external clock from pads.
 - Prescaler divided the clock by 1/2/4/8/16/32/64/128.
- Counter:
 - Integrated four 16-bit counters 0/1/2/3.
 - Every counter controls the output of one pair of channels.
 - Counter 0 can also be selected as global counter to control the output of all channels.
 - Each counter supports two counting modes: up and up-down counting.
 - Each counter supports programmable overflow value, not fixed to 0xFFFF.
- Channels:
 - Implemented 8 output channels.
 - 8 channels are divided to 4 pairs. For each pair of channels, the output of the two channels can be independent or combined as complementary.
 - The polarity of each channel is configurable.
 - Each channel output software control is available.
 - Each channel output status is configurable when counter is off.
- Support 7 PWM output mode:
 - Independent Edge-aligned PWM output.
 - Independent Center-aligned PWM output.
 - Complementary Edge-aligned PWM output.
 - Complementary Symmetric Center-aligned PWM output.
 - Complementary Asymmetric Center-aligned PWM output.
 - Complementary Variable Edge Placement PWM output.
 - Complementary Double Switching PWM output.
- Support PWM output period dithering and pulse width dithering.
- Deadtime insertion:
 - Deadtime insertion is available for each pair of channels in complementary mode.
 - Deadtime delay value for each pair of channels is configurable independently.

- Fault protection:
 - Support 4 fault inputs 0/1/2/3, regard fault inputs 0/1 as fault type A and fault input 2/3 as fault type B.
 - Fault A has high priority over fault B.
 - Each fault input polarity is configurable.
 - Each fault input enable is selectable.
 - Each pair of channel fault control enable is selectable.
 - A configurable glitch filter can be selected for each fault input.
 - Output status of channels is configurable independently when fault A event and fault B event are detected.
 - Output of channels supports hardware auto recovery and manual recovery in next PWM period after fault event is inactive.
- Generation of match triggers and initial triggers.
- Register access protection:
 - Synchronized loading of write buffered registers
 - Register reload capacity supports half cycle reload for up-down counting mode and full cycle reload for all modes, reload frequency is also configurable.
 - Write protection for critical registers bits.
 - Write buffer for MODn and CVn registers.
- Interrupt
 - Channel match event interrupt
 - Timer overflow interrupt
 - Fault event interrupt
 - Register reload interrupt
- Support DMA request generation and acknowledge with DMA done.

29.1.2 Block diagram

The following figure is a block diagram of the Motor Control PWM (MCPWM) module.

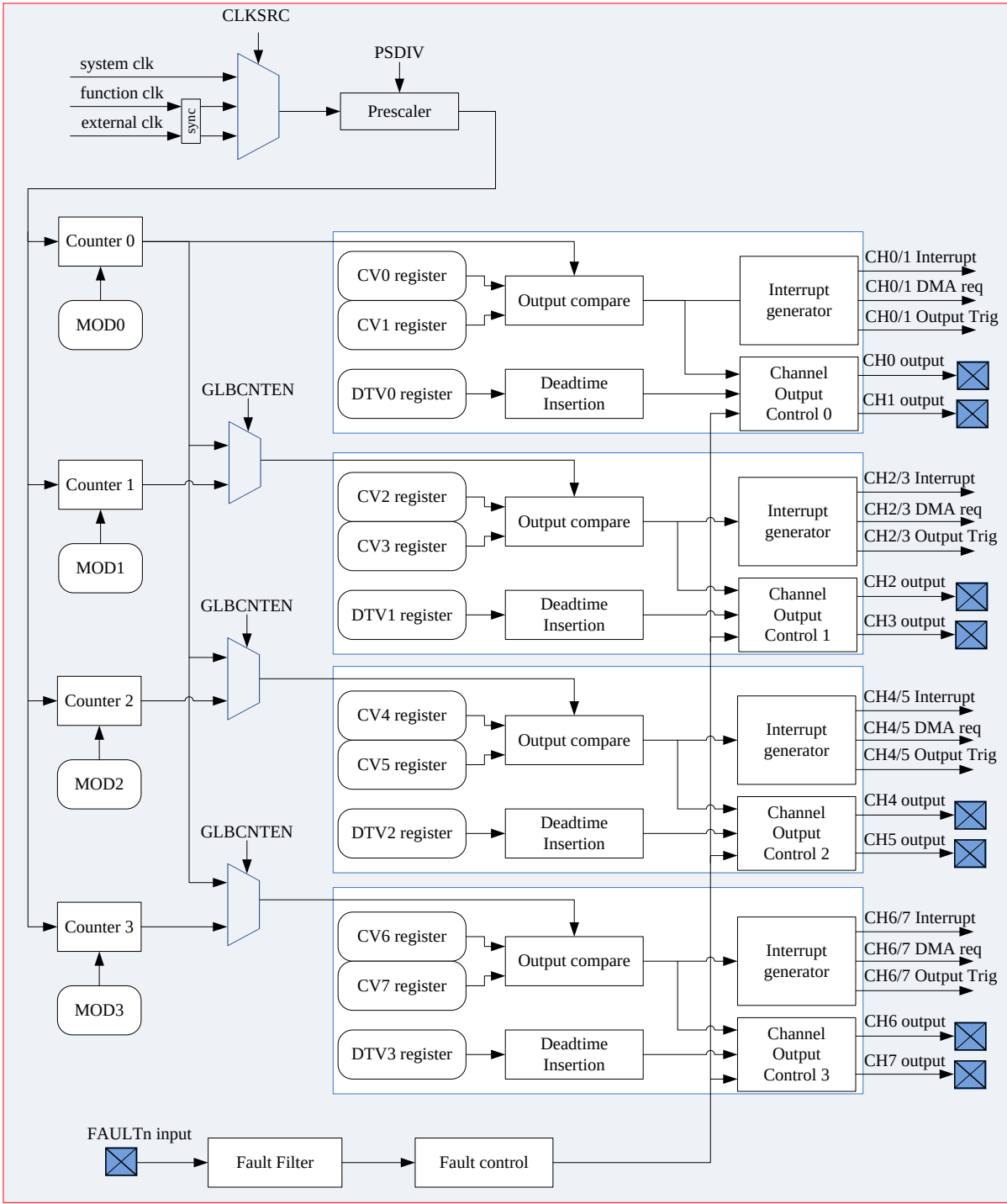


Figure 29-1 MCPWM block diagram

29.1.3 Instantiation and channel information

The following table lists the MCPWM instantiation and channel information for each chip in the product series.

Configuration	K144MC	K146MC	K148MC
Instance(s)	MCPWM0	MCPWM0	MCPWM0

Configuration	K144MC	K146MC	K148MC
	MCPWM1	MCPWM1	MCPWM1
Channel(s)	8	8	8

29.2 Memory map and register definition

This section provides the memory map and register definitions of the MCPWM module.

29.2.1 Memory map

MCPWM0 base address: 0x4000E000

MCPWM1 base address: 0x4001A000

Table 29-1 MCPWM Memory map

Offset	Register	Width (In bits)	Reset Value
0x00	MCPWM Time Base Register (MCPWM_TIMEBASE)	32	0x00000000
0x04 + n*0x4 (n=0~3)	MCPWM Counter n Register (MCPWM_CNTn)	32	0x00000000
0x14 + n*0x4 (n=0~3)	MCPWM Counter n Modulo Value Register (MCPWM_MODn)	32	0x00000000
0x24 + n*0x4 (n=0~3)	MCPWM Deadtime Value n Register (MCPWM_DTVALn)	32	0x00000000
0x34	MCPWM Global Control Register (MCPWM_GLBCR)	32	0x00000000
0x38	MCPWM Global Status Register (MCPWM_GLBSR)	32	0x00010000
0x3C + n*0x4, (n=0~7)	MCPWM Channel n Mode Configuration Register (MCPWM_CFGn)	32	0x00000000
0x5C + n*0x4, (n=0~7)	MCPWM Channel n Value Register (MCPWM_CVn)	32	0x00000000
0x7C	MCPWM Output Control Register (MCPWM_OUTCR)	32	0x00000000
0x80	MCPWM Output Trigger Register (MCPWM_OUTTRIG)	32	0x00000000
0x84	MCPWM Channel Output Software Control Register (MCPWM_OUTSWCR)	32	0x00000000
0x88	MCPWM Pairs Of Channel Control Register (MCPWM_PCR)	32	0x00000000
0x8C	MCPWM Fault Control Register (MCPWM_FLTCR)	32	0x00000000
0x90	MCPWM Fault Status Register (MCPWM_FLTSR)	32	0x00000000
0x94	MCPWM Fault A Safe Status Value Register (MCPWM_FLTASS)	32	0x00000000
0x98	MCPWM Fault B Safe Status Value Register (MCPWM_FLTBSS)	32	0x00000000
0x9C	MCPWM Synchronization Register (MCPWM_SYNC)	32	0x00000000
0xA0	MCPWM Reload Register (MCPWM_RELOAD)	32	0x00000000
0xA4	MCPWM Reload Frequency Register (MCPWM_LDFREQ)	32	0x00000000

Offset	Register	Width (In bits)	Reset Value
0x100 + n*0x4 (n=0~3)	MCPWM MODn Dither Register (MCPWM_MODn_DITHER)	32	0x00000000
0x110 + n*0x4 (n=0~7)	MCPWM CVn Dither Register (MCPWM_CVn_DITHER)	32	0x00000000

29.2.2 MCPWM Time Base Register (MCPWM_TIMEBASE)

29.2.2.1 Offset

Register	Offset
MCPWM_TIMEBASE	0x00

29.2.2.2 Function

This register selects the counter clock source and prescaler divider factors. It also determines the counter behavior in debug mode.

29.2.2.3 Fields

Field	Access	Description	Reset Value
1:0 CKSRC	RW	Select the MCPWM counter clock source. This field is write protected, it can only be written when write protection is disabled (WPDIS=1). 00b: MCPWM counter is disabled 01b: Core clock is selected as MCPWM counter clock 10b: Function clock is selected as MCPWM counter clock 11b: External clock is selected as MCPWM counter clock	0x0
4:2 PSDIV	RW	Select one divide factor to divide the clock selected by CKSRC. This field is write protected, it can only be written when write protection is disabled (WPDIS=1). 000b: Divide by 1 001b: Divide by 2 010b: Divide by 4 011b: Divide by 8 100b: Divide by 16 101b: Divide by 32 110b: Divide by 64 111b: Divide by 128	0x0
6:5 DBGM	RW	This field selects the counter behavior in chip debug mode	0x0
31:7 RSVD	RO	Reserved and read as 0	0x0

29.2.3 MCPWM Counter n Register (MCPWM_CNTn)

29.2.3.1 Offset

Register	Offset

Register	Offset
MCPWM_CNTn	0x04+4*n (n=0~3)

29.2.3.2 Fields

Field	Access	Description	Reset Value
15:0 CNT	RW	This field is the 16-bit counter value, write to this field will load the counter with initial value 0, reset will clear counter value.	0x0
31:16 RSVD	RO	Reserved and read as 0	0x0

29.2.4 MCPWM Counter n Modulo Value Register (MCPWM_MODn)

29.2.4.1 Offset

Register	Offset
MCPWM_MODn	0x14+4*n (n=0~3)

29.2.4.2 Fields

Field	Access	Description	Reset Value
15:0 MOD	RW	When the 16-bit CNTn reaches the MODn value and continues to increment, the counter n timer overflow flag TOF is asserted. Writing a value to this field will latch the write value into a write buffer.	0x0
31:16 RSVD	RO	Reserved and read as 0	0x0

29.2.5 MCPWM Deadtime Value n Register (MCPWM_DTVALn)

29.2.5.1 Offset

Register	Offset
MCPWM_DTVALn	0x24+4*n (n=0~3)

29.2.5.2 Fields

Field	Access	Description	Reset Value
11:0 DTVAL	RW	Configure the deadtime insertion value for the deadtime counter. This field is write protected, it can only be written when write protection is disabled.	0x0
31:12 RSVD	RO	Reserved and read as 0	0x0

29.2.6 MCPWM Global Control Register (MCPWM_GLBCCR)

29.2.6.1 Offset

Register	Offset
MCPWM_GLBCR	0x34

29.2.6.2 Fields

Field	Access	Description	Reset Value
0 CHOE0	RW	0b: Channel 0 output was disabled. 1b: Channel 0 output was enabled.	0x0
1 CHOE1	RW	0b: Channel 1 output was disabled. 1b: Channel 1 output was enabled.	0x0
2 CHOE2	RW	0b: Channel 2 output was disabled. 1b: Channel 2 output was enabled.	0x0
3 CHOE3	RW	0b: Channel 3 output was disabled. 1b: Channel 3 output was enabled.	0x0
4 CHOE4	RW	0b: Channel 4 output was disabled. 1b: Channel 4 output was enabled.	0x0
5 CHOE5	RW	0b: Channel 5 output was disabled. 1b: Channel 5 output was enabled.	0x0
6 CHOE6	RW	0b: Channel 6 output was disabled. 1b: Channel 6 output was enabled.	0x0
7 CHOE7	RW	0b: Channel 7 output was disabled. 1b: Channel 7 output was enabled.	0x0
8 CNTM0	RW	This field is write protected, it can only be written when write protection is disabled (WPDIS=1). 0b: MCPWM counter 0 operates in up counting mode. 1b: MCPWM counter 0 operates in up-down counting mode.	0x0
9 CNTM1	RW	This field is write protected, it can only be written when write protection is disabled (WPDIS=1). 0b: MCPWM counter 1 operates in up counting mode. 1b: MCPWM counter 1 operates in up-down counting mode.	0x0
10 CNTM2	RW	This field is write protected, it can only be written when write protection is disabled (WPDIS=1). 0b: MCPWM counter 2 operates in up counting mode. 1b: MCPWM counter 2 operates in up-down counting mode.	0x0
11 CNTM3	RW	This field is write protected, it can only be written when write protection is disabled (WPDIS=1). 0b: MCPWM counter 3 operates in up counting mode. 1b: MCPWM counter 3 operates in up-down counting mode.	0x0
12 CNTEN0	RW	When GLBCNTEN is disabled, this bit is used to enable counter 0; when GLBCNTEN is enabled, counter 0 will be automatically enabled and used as global counter, CNTEN0 does not make sense, and MOD0 will also be used as global MOD. This field is write protected, it can only be written when write protection is disabled (WPDIS=1). 0b: Disable counter 0. 1b: Enable counter 0.	0x0
13 CNTEN1	RW	When GLBCNTEN is disabled, this bit is used to enable counter 1; when GLBCNTEN is enabled, this	0x0

Field	Access	Description	Reset Value
		bit will not make sense, counter 1 will not count, and MOD1 will also not make sense. This field is write protected, it can only be written when write protection is disabled (WPDIS=1). 0b: Disable counter 1. 1b: Enable counter 1.	
14 CNTEN2	RW	When GLBCNTEN is disabled, this bit is used to enable counter 2; when GLBCNTEN is enabled, this bit will not make sense, counter 2 will not count, and MOD2 will also not make sense. This field is write protected, it can only be written when write protection is disabled (WPDIS=1). 0b: Disable counter 2. 1b: Enable counter 2.	0x0
15 CNTEN3	RW	When GLBCNTEN is disabled, this bit is used to enable counter 3; when GLBCNTEN is enabled, this bit will not make sense, counter 3 will not count, and MOD3 will also not make sense. This field is write protected, it can only be written when write protection is disabled (WPDIS=1). 0b: Disable counter 3. 1b: Enable counter 3.	0x0
16 RLDIE0	RW	0b: Disable reload interrupt. 1b: Enable reload interrupt.	0x0
17 RLDIE1	RW	0b: Disable reload interrupt. 1b: Enable reload interrupt.	0x0
18 RLDIE2	RW	0b: Disable reload interrupt. 1b: Enable reload interrupt.	0x0
19 RLDIE3	RW	0b: Disable reload interrupt. 1b: Enable reload interrupt.	0x0
20 TOIE0	RW	0b: Disable counter 0 overflow interrupt. 1b: Enable counter 0 overflow interrupt.	0x0
21 TOIE1	RW	0b: Disable counter 1 overflow interrupt. 1b: Enable counter 1 overflow interrupt.	0x0
22 TOIE2	RW	0b: Disable counter 2 overflow interrupt. 1b: Enable counter 2 overflow interrupt.	0x0
23 TOIE3	RW	0b: Disable counter 3 overflow interrupt. 1b: Enable counter 3 overflow interrupt.	0x0
29:24 RSVD	RO	Reserved and read as 0	0x0
30 WPEN	RW	The WPEN bit is the negation of the WPDIS bit. WPEN is set when 1 is written to it. WPEN is cleared when WPEN bit is read as a 1 and then 1 is written to WPDIS. Writing 0 to WPEN has no effect. 0b: Write protection is disabled. Write protected bits can be written. 1b: Write protection is enabled. Write protected bits cannot be written.	0x0
31 GLBCNTEN	RW	This field is write protected, it can be written only when write protection is disabled (WPDIS=1). When this bit is enabled, the CNT0 will be used as global counter, and MOD0 is also shared by all channels. 0b: Global counter is disabled. 1b: Global counter is enabled.	0x0

29.2.7 MCPWM Global Status Register (MCPWM_GLBSR)

29.2.7.1 Offset

Register	Offset
MCPWM_GLBSR	0x38

29.2.7.2 Fields

Field	Access	Description	Reset Value
0 CH0F	RO	Read only bit. CH0F bit is the mirror of MCPWM_CFG0[CHF] bit. This flag is cleared by writing 0 to MCPWM_CFG0[CHF] after reading MCPWM_CFG0[CHF] while MCPWM_CFG0[CHF] bit is set. Writing a 1 to MCPWM_CFG0[CHF] has no effect. 0b: Counter match didn't occur in this channel 0. 1b: Counter match occurred in this channel 0.	0x0
1 CH1F	RO	Read only bit. CH1F bit is the mirror of MCPWM_CFG1[CHF] bit. This flag is cleared by writing 0 to MCPWM_CFG1[CHF] after reading MCPWM_CFG1[CHF] while MCPWM_CFG1[CHF] bit is set. Writing a 1 to MCPWM_CFG1[CHF] has no effect. 0b: Counter match didn't occur in this channel 1. 1b: Counter match occurred in this channel 1.	0x0
2 CH2F	RO	Read only bit. CH2F bit is the mirror of MCPWM_CFG2[CHF] bit. This flag is cleared by writing 0 to MCPWM_CFG2[CHF] after reading MCPWM_CFG2[CHF] while MCPWM_CFG2[CHF] bit is set. Writing a 1 to MCPWM_CFG2[CHF] has no effect. 0b: Counter match didn't occur in this channel 2. 1b: Counter match occurred in this channel 2.	0x0
3 CH3F	RO	Read only bit. CH3F bit is the mirror of MCPWM_CFG3[CHF] bit. This flag is cleared by writing 0 to MCPWM_CFG3[CHF] after reading MCPWM_CFG3[CHF] while MCPWM_CFG3[CHF] bit is set. Writing a 1 to MCPWM_CFG3[CHF] has no effect. 0b: Counter match didn't occur in this channel 3. 1b: Counter match occurred in this channel 3.	0x0
4 CH4F	RO	Read only bit. CH4F bit is the mirror of MCPWM_CFG4[CHF] bit. This flag is cleared by writing 0 to MCPWM_CFG4[CHF] after reading MCPWM_CFG4[CHF] while MCPWM_CFG4[CHF] bit is set. Writing a 1 to MCPWM_CFG4[CHF] has no effect. 0b: Counter match didn't occur in this channel 4. 1b: Counter match occurred in this channel 4.	0x0
5 CH5F	RO	Read only bit. CH5F bit is the mirror of MCPWM_CFG5[CHF] bit. This flag is cleared by writing 0 to MCPWM_CFG5[CHF] after reading MCPWM_CFG5[CHF] while MCPWM_CFG5[CHF] bit is set. Writing a 1 to MCPWM_CFG5[CHF] has no	0x0

Field	Access	Description	Reset Value
		effect. 0b: Counter match didn't occur in this channel 5. 1b: Counter match occurred in this channel 5.	
6 CH6F	RO	Read only bit. CH6F bit is the mirror of MCPWM_CFG6[CHF] bit. This flag is cleared by writing 0 to MCPWM_CFG6[CHF] after reading MCPWM_CFG6[CHF] while MCPWM_CFG6[CHF] bit is set. Writing a 1 to MCPWM_CFG6[CHF] has no effect. 0b: Counter match didn't occur in this channel 6. 1b: Counter match occurred in this channel 6.	0x0
7 CH7F	RO	Read only bit. CH7F bit is the mirror of MCPWM_CFG7[CHF] bit. This flag is cleared by writing 0 to MCPWM_CFG7[CHF] after reading MCPWM_CFG7[CHF] while MCPWM_CFG7[CHF] bit is set. Writing a 1 to MCPWM_CFG7[CHF] has no effect. 0b: Counter match didn't occur in this channel 7. 1b: Counter match occurred in this channel 7.	0x0
8 RLDF0	R/W0C	When counter 0 reaches the reload opportunity, the reload flag RLDF0 can be set. But RLDF0 is not set at every reload opportunity, it depends on the configured reload frequency LDFREQ[LDFREQ0] fields. RLDF0 is cleared by software writing a 0 to this bit after reading RLDF0 flag while RLDF0 is set. Writing a 1 to RLDF0 has no effect. 0b: Counter 0 didn't reach the reload point. 1b: Counter 0 reached the reload point.	0x0
9 RLDF1	R/W0C	When counter 1 reaches the reload opportunity, the reload flag RLDF1 can be set. But RLDF1 is not set at every reload opportunity, it depends on the configured reload frequency LDFREQ[LDFREQ1] fields. RLDF1 is cleared by software writing a 0 to this bit after reading RLDF1 flag while RLDF1 is set. Writing a 1 to RLDF1 has no effect. 0b: Counter 1 didn't reach the reload point. 1b: Counter 1 reached the reload point.	0x0
10 RLDF2	R/W0C	When counter 2 reaches the reload opportunity, the reload flag RLDF2 can be set. But RLDF2 is not set at every reload opportunity, it depends on the configured reload frequency LDFREQ[LDFREQ2] fields. RLDF2 is cleared by software writing a 0 to this bit after reading RLDF2 flag while RLDF2 is set. Writing a 1 to RLDF2 has no effect. 0b: Counter 2 didn't reach the reload point. 1b: Counter 2 reached the reload point.	0x0
11 RLDF3	R/W0C	When counter 3 reaches the reload opportunity, the reload flag RLDF3 can be set. But RLDF3 is not set at every reload opportunity, it depends on the configured reload frequency LDFREQ[LDFREQ3] fields. RLDF3 is cleared by software writing a 0 to this bit after reading RLDF3 flag while RLDF3 is set. Writing a 1 to RLDF3 has no effect. 0b: Counter 3 didn't reach the reload point. 1b: Counter 3 reached the reload point.	0x0

Field	Access	Description	Reset Value
12 TOF0	R/W0C	Set by hardware, TOF0 is asserted when MCPWM counter 0 reaches the MOD0 value and continues to count. TOF0 is cleared by software writing a 0 to this bit after reading TOF0 flag while TOF0 is set. Writing a 1 to TOF0 has no effect. 0b: Counter 0 timer overflow didn't occur. 1b: Counter 0 timer overflow occurred.	0x0
13 TOF1	R/W0C	Set by hardware, TOF1 is asserted when MCPWM counter 1 reaches the MOD1 value and continues to count. TOF1 is cleared by software writing a 0 to this bit after reading TOF1 flag while TOF1 is set. Writing a 1 to TOF1 has no effect. 0b: Counter 1 timer overflow didn't occur. 1b: Counter 1 timer overflow occurred.	0x0
14 TOF2	R/W0C	Set by hardware, TOF2 is asserted when MCPWM counter 2 reaches the MOD2 value and continues to count. TOF2 is cleared by software writing a 0 to this bit after reading TOF2 flag while TOF2 is set. Writing a 1 to TOF2 has no effect. 0b: Counter 2 timer overflow didn't occur. 1b: Counter 2 timer overflow occurred.	0x0
15 TOF3	R/W0C	Set by hardware, TOF3 is asserted when MCPWM counter 3 reaches the MOD3 value and continues to count. TOF3 is cleared by software writing a 0 to this bit after reading TOF3 flag while TOF3 is set. Writing a 1 to TOF3 has no effect. 0b: Counter 3 timer overflow didn't occur. 1b: Counter 3 timer overflow occurred.	0x0
16 WPDIS	RW	This bit indicates whether the write protected bits can be written or not. The WPDIS bit is negation of the WPEN bit. WPDIS is cleared when 1 is written to WPEN, WPDIS is set when WPEN bit is read as 1 and then 1 is written to WPDIS. Writing 0 to WPDIS has no effect. 0b: Write protection is enabled. 1b: Write protection is disabled.	0x1
31:17 RSVD	RO	Reserved and read as 0	0x0

29.2.8 MCPWM Channel n Mode Configuration Register (MCPWM_CFGn)

29.2.8.1 Offset

Register	Offset
MCPWM_CFGn	0x3C+4*n (n=0~7)

29.2.8.2 Fields

Field	Access	Description	Reset Value
0 CPP	RW	This bit defines the channel output pulse polarity when counter matches the CV value. This field is write	0x0

Field	Access	Description	Reset Value
		protected, it can only be written when write protection is disabled (WPDIS=1). For low pulse polarity, channel output is inactive status when counter value is more than CV value, and channel output is active status when counter value is less than CV value. For high pulse polarity, channel output is active status when counter value is more than CV value, and channel output is inactive status when counter value is less than CV value. NOTE Channel output active/inactive status is decided by channel output polarity OUTCR[POLx] bit. 0b: Channel n output is low pulse polarity. 1b: Channel n output is high pulse polarity.	
3:1 RSVD	RO	Reserved	0x0
4 CHIE	RW	0b: Disable this channel interrupt. 1b: Enable this channel interrupt.	0x0
5 CHF	R/W0C	Set by hardware when this channel operating counter matches its channel value CV. CHF is cleared by writing a 0 to the CHF bit after reading CHF while it's set. Writing a 1 to CHF has no effect. 0b: Counter match didn't occur in this channel. 1b: Counter match occurred in this channel.	0x0
6 DMA	RW	0b: DMA transfer for this channel is disabled. 1b: DMA transfer for this channel is enabled. NOTE DMA request is only available if channel interrupt is also enabled.	0x0
31:7 RSVD	RO	Reserved and read as 0	0x0

29.2.9 MCPWM Channel n Value Register (MCPWM_CVn)

29.2.9.1 Offset

Register	Offset
MCPWM_CVn	0x5C+4*n (n=0~7)

29.2.9.2 Fields

Field	Access	Description	Reset Value
15:0 CV	RW	When related counter reaches channel n CV value, channel n event occurs, channel n flag is set. When channel n operating counter is counting, writing a value to this field will latch the write value into a write buffer.	0x0
31:16 RSVD	RO	Reserved and read as 0	0x0

29.2.10 MCPWM Output Control Register (MCPWM_OUTCR)

29.2.10.1 Offset

Register	Offset
MCPWM_OUTCR	0x7C

29.2.10.2 Fields

Field	Access	Description	Reset Value
0 POL0	RW	This bit defines the channel 0 active state in output mode. This field is write protected, it can only be written when write protection is disabled. NOTE: OUTCR[POL0] and CFG0[CPP] bits will both determine the PWM output status. 0b: Channel 0 output polarity is active high, so the logic one is an active state. 1b: Channel 0 output polarity is active low, so the logic zero is an active state.	0x0
1 POL1	RW	This bit defines the channel 1 active state in output mode. This field is write protected, it can only be written when write protection is disabled. NOTE: OUTCR[POL1] and CFG1[CPP] bits will both determine the PWM output status. 0b: Channel 1 output polarity is active high, so the logic one is an active state. 1b: Channel 1 output polarity is active low, so the logic zero is an active state.	0x0
2 POL2	RW	This bit defines the channel 2 active state in output mode. This field is write protected, it can only be written when write protection is disabled. NOTE: OUTCR[POL2] and CFG2[CPP] bits will both determine the PWM output status. 0b: Channel 2 output polarity is active high, so the logic one is an active state. 1b: Channel 2 output polarity is active low, so the logic zero is an active state.	0x0
3 POL3	RW	This bit defines the channel 3 active state in output mode. This field is write protected, it can only be written when write protection is disabled. NOTE: OUTCR[POL3] and CFG3[CPP] bits will both determine the PWM output status. 0b: Channel 3 output polarity is active high, so the logic one is an active state. 1b: Channel 3 output polarity is active low, so the logic zero is an active state.	0x0
4 POL4	RW	This bit defines the channel 4 active state in output mode. This field is write protected, it can only be written when write protection is disabled. NOTE: OUTCR[POL4] and CFG4[CPP] bits will both determine the PWM output status. 0b: Channel 4 output polarity is active high, so the logic one is an active state. 1b: Channel 4 output polarity is active low, so the logic zero is an active state.	0x0
5 POL5	RW	This bit defines the channel 5 active state in output mode. This field is write protected, it can only be written when write protection is disabled.	0x0

Field	Access	Description	Reset Value
		NOTE: OUTCR[POL5] and CFG5[CPP] bits will both determine the PWM output status. 0b: Channel 5 output polarity is active high, so the logic one is an active state. 1b: Channel 5 output polarity is active low, so the logic zero is an active state.	
6 POL6	RW	This bit defines the channel 6 active state in output mode. This field is write protected, it can only be written when write protection is disabled. NOTE: OUTCR[POL6] and CFG6[CPP] bits will both determine the PWM output status. 0b: Channel 6 output polarity is active high, so the logic one is an active state. 1b: Channel 6 output polarity is active low, so the logic zero is an active state.	0x0
7 POL7	RW	This bit defines the channel 7 active state in output mode. This field is write protected, it can only be written when write protection is disabled. NOTE: OUTCR[POL7] and CFG7[CPP] bits will both determine the PWM output status. 0b: Channel 7 output polarity is active high, so the logic one is an active state. 1b: Channel 7 output polarity is active low, so the logic zero is an active state.	0x0
8 OFFVAL0	RW	This bit defines the channel 0 output status when counter is off. This field is write protected, it can only be written when write protection is disabled. 0b: Channel 0 output is 0 in counter off status. 1b: Channel 0 output is 1 in counter off status.	0x0
9 OFFVAL1	RW	This bit defines the channel 1 output status when counter is off. This field is write protected, it can only be written when write protection is disabled. 0b: Channel 1 output is 0 in counter off status. 1b: Channel 1 output is 1 in counter off status.	0x0
10 OFFVAL2	RW	This bit defines the channel 2 output status when counter is off. This field is write protected, it can only be written when write protection is disabled. 0b: Channel 2 output is 0 in counter off status. 1b: Channel 2 output is 1 in counter off status.	0x0
11 OFFVAL3	RW	This bit defines the channel 3 output status when counter is off. This field is write protected, it can only be written when write protection is disabled. 0b: Channel 3 output is 0 in counter off status. 1b: Channel 3 output is 1 in counter off status.	0x0
12 OFFVAL4	RW	This bit defines the channel 4 output status when counter is off. This field is write protected, it can only be written when write protection is disabled. 0b: Channel 4 output is 0 in counter off status. 1b: Channel 4 output is 1 in counter off status.	0x0
13 OFFVAL5	RW	This bit defines the channel 5 output status when counter is off. This field is write protected, it can only be written when write protection is disabled. 0b: Channel 5 output is 0 in counter off status. 1b: Channel 5 output is 1 in counter off status.	0x0
14	RW	This bit defines the channel 6 output status when counter is off. This field is write protected, it can only	0x0

Field	Access	Description	Reset Value
OFFVAL6		be written when write protection is disabled. 0b: Channel 6 output is 0 in counter off status. 1b: Channel 6 output is 1 in counter off status.	
15 OFFVAL7	RW	This bit defines the channel 7 output status when counter is off. This field is write protected, it can only be written when write protection is disabled. 0b: Channel 7 output is 0 in counter off status. 1b: Channel 7 output is 1 in counter off status.	0x0
31:16 RSVD	RO	Reserved and read as 0	0x0

29.2.11 MCPWM Output Trigger Register (MCPWM_OUTTRIG)

29.2.11.1 Offset

Register	Offset
MCPWM_OUTTRIG	0x80

29.2.11.2 Fields

Field	Access	Description	Reset Value
0 TRIGE0	RW	Enable the generation of match trigger when counter reaches CV0 value. 0b: Disable channel 0 match trigger generation. 1b: Enable channel 0 match trigger generation.	0x0
1 TRIGE1	RW	Enable the generation of match trigger when counter reaches CV1 value. 0b: Disable channel 1 match trigger generation. 1b: Enable channel 1 match trigger generation.	0x0
2 TRIGE2	RW	Enable the generation of match trigger when counter reaches CV2 value. 0b: Disable channel 2 match trigger generation. 1b: Enable channel 2 match trigger generation.	0x0
3 TRIGE3	RW	Enable the generation of match trigger when counter reaches CV3 value. 0b: Disable channel 3 match trigger generation. 1b: Enable channel 3 match trigger generation.	0x0
4 TRIGE4	RW	Enable the generation of match trigger when counter reaches CV4 value. 0b: Disable channel 4 match trigger generation. 1b: Enable channel 4 match trigger generation.	0x0
5 TRIGE5	RW	Enable the generation of match trigger when counter reaches CV5 value. 0b: Disable channel 5 match trigger generation. 1b: Enable channel 5 match trigger generation.	0x0
6 TRIGE6	RW	Enable the generation of match trigger when counter reaches CV6 value. 0b: Disable channel 6 match trigger generation. 1b: Enable channel 6 match trigger generation.	0x0
7 TRIGE7	RW	Enable the generation of match trigger when counter reaches CV7 value. 0b: Disable channel 7 match trigger generation. 1b: Enable channel 7 match trigger generation.	0x0

Field	Access	Description	Reset Value
8 INITRIGE	RW	Enable the generation of initial trigger when counter reaches value 0. 0b: Disable initialization trigger generation. 1b: Enable initialization trigger generation.	0x0
9 TRIGF	R/W0C	When any channel match trig is generated, this flag is set by hardware. TRIGF is cleared when writing 0 to TRIGF after reading TRIGF while TRIGF is 1. 0b: No channel match trigger is generated. 1b: At least one channel match trigger is generated.	0x0
31:10 RSVD	RO	Reserved and read as 0	0x0

29.2.12 MCPWM Channel Output Software Control Register (MCPWM_OUTSWCR)

29.2.12.1 Offset

Register	Offset
MCPWM_OUTSWCR	0x84

29.2.12.2 Fields

Field	Access	Description	Reset Value
0 CH0OC	RW	0b: Disable channel 0 output software control. 1b: Enable channel 0 output software control.	0x0
1 CH1OC	RW	0b: Disable channel 1 output software control. 1b: Enable channel 1 output software control.	0x0
2 CH2OC	RW	0b: Disable channel 2 output software control. 1b: Enable channel 2 output software control.	0x0
3 CH3OC	RW	0b: Disable channel 3 output software control. 1b: Enable channel 3 output software control.	0x0
4 CH4OC	RW	0b: Disable channel 4 output software control. 1b: Enable channel 4 output software control.	0x0
5 CH5OC	RW	0b: Disable channel 5 output software control. 1b: Enable channel 5 output software control.	0x0
6 CH6OC	RW	0b: Disable channel 6 output software control. 1b: Enable channel 6 output software control.	0x0
7 CH7OC	RW	0b: Disable channel 7 output software control. 1b: Enable channel 7 output software control.	0x0
8 CH0OCV	RW	0b: Channel 0 output is forced to inactive by software control. 1b: Channel 0 output is forced to active by software control.	0x0
9 CH1OCV	RW	0b: Channel 1 output is forced to inactive by software control. 1b: Channel 1 output is forced to active by software control.	0x0
10 CH2OCV	RW	0b: Channel 2 output is forced to inactive by software control.	0x0

Field	Access	Description	Reset Value
		1b: Channel 2 output is forced to active by software control.	
11 CH3OCV	RW	0b: Channel 3 output is forced to inactive by software control. 1b: Channel 3 output is forced to active by software control.	0x0
12 CH4OCV	RW	0b: Channel 4 output is forced to inactive by software control. 1b: Channel 4 output is forced to active by software control.	0x0
13 CH5OCV	RW	0b: Channel 5 output is forced to inactive by software control. 1b: Channel 5 output is forced to active by software control.	0x0
14 CH6OCV	RW	0b: Channel 6 output is forced to inactive by software control. 1b: Channel 6 output is forced to active by software control.	0x0
15 CH7OCV	RW	0b: Channel 7 output is forced to inactive by software control. 1b: Channel 7 output is forced to active by software control.	0x0
31:16 RSVD	RO	Reserved and read as 0	0x0

29.2.13 MCPWM Pairs Of Channel Control Register (MCPWM_PCR)

29.2.13.1 Offset

Register	Offset
MCPWM_PCR	0x88

29.2.13.2 Fields

Field	Access	Description	Reset Value
0 FCTLEN0	RW	Enable the fault control in the channels (n) and (n+1), where n=0. This field is write protected. It can be written only when write protection is disabled. 0b: The fault control in this pair of channels is disabled. 1b: The fault control in this pair of channels is enabled.	0x0
1 DTEN0	RW	Enable the deadtime insertion in the channels (n) and (n+1), where n=0. This field is write protected. It can be written only when write protection is disabled. 0b: The deadtime insertion in this pair of channels is disabled. 1b: The deadtime insertion in this pair of channels is enabled.	0x0
2 SYNCEN0	RW	Enable the synchronization of CV(n) and CV(n+1) register value, where n=0. 0b: Synchronization in this pair of channels is	0x0

Field	Access	Description	Reset Value
		disabled. 1b: Synchronization in this pair of channels is enabled.	
3 ASYMEN0	RW	Enable the asymmetric PWM output in complementary CPWM output mode for channels(n) and (n+1), where n=0. This bit only makes sense when channel(n) and channel(n+1) are in complementary CPWM output mode. This field is write protected, it can only be written when write protection is disabled (WPDIS=1). 0b: Channel(n) and channel(n+1) are symmetric PWM output in complementary CPWM mode. 1b: Channel(n) and channel(n+1) are asymmetric PWM output in complementary CPWM mode.	0x0
4 CVSELO	RW	This bit selects either CV(n) or CV(n+1) as channel value to generate PWM output in complementary mode, where n=0. CVSELO is valid only in complementary EPWM mode and complementary CPWM mode. This field is write protected, it can be written only when write protection is disabled. 0b: CV(n) is used to generate PWM output in complementary mode. 1b: CV(n+1) is used to generate PWM output in complementary mode.	0x0
5 PEC0	RW	In complementary mode, this bit defines whether one of CV(n)/CV(n+1) is or both CV(n)/CV(n+1) are used to set/clear PWM output during EPWM cycle or CPWM half cycle, where n=0. PEC0 is valid only when COMB0=1. This field is write protected, it can be written only when write protection is disabled. 0b: In complementary mode, during one EPWM cycle or CPWM half cycle, either CV(n) or CV(n+1) is used to set or clear channel(n) PWM output, channel(n+1) output is complementary with channel(n). 1b: In complementary mode, during one EPWM cycle or CPWM half cycle, one of CV(n) and CV(n+1) is used to set channel(n) PWM output, the other is used to clear channel(n) PWM output, channel(n+1) output is complementary with channel(n).	0x0
6 COMB0	RW	Enable the combination of channel(n) and channel(n+1), where n=0. When this pair of channels is combined, outputs of their channels are complementary. This field is write protected, it can be written only when write protection is disabled. 0b: The channel(n) and channel(n+1) are independent. 1b: The channel(n) and channel(n+1) are combined, channel(n) and channel(n+1) outputs are complementary.	0x0
7 RSVD	RO	Reserved and read as 0	0x0
8 FCTLEN1	RW	Enable the fault control in the channels (n) and (n+1), where n=2. This field is write protected. It can be written only when write protection is disabled. 0b: The fault control in this pair of channels is disabled. 1b: The fault control in this pair of channels is enabled.	0x0

Field	Access	Description	Reset Value
9 DTEN1	RW	Enable the deadtime insertion in the channels (n) and (n+1), where n=2. This field is write protected. It can be written only when write protection is disabled. 0b: The deadtime insertion in this pair of channels is disabled. 1b: The deadtime insertion in this pair of channels is enabled.	0x0
10 SYNCEN1	RW	Enable the synchronization of CV(n) and CV(n+1) register value, where n=2. 0b: Synchronization in this pair of channels is disabled. 1b: Synchronization in this pair of channels is enabled.	0x0
11 ASYMEN1	RW	Enable the asymmetric PWM output in complementary CPWM output mode for channels(n) and (n+1), where n=2. This bit only makes sense when channel(n) and channel(n+1) are in complementary CPWM output mode. This field is write protected, it can only be written when write protection is disabled (WPDIS=1). 0b: Channel(n) and channel(n+1) are symmetric PWM output in complementary CPWM mode. 1b: Channel(n) and channel(n+1) are asymmetric PWM output in complementary CPWM mode.	0x0
12 CVSEL1	RW	This bit selects either CV(n) or CV(n+1) as channel value to generate PWM output in complementary mode, where n=2. CVSEL1 is valid only in complementary EPWM mode and complementary CPWM mode. This field is write protected, it can be written only when write protection is disabled. 0b: CV(n) is used to generate PWM output in complementary mode. 1b: CV(n+1) is used to generate PWM output in complementary mode.	0x0
13 PEC1	RW	In complementary mode, this bit defines whether one of CV(n)/CV(n+1) is or both CV(n)/CV(n+1) are used to set/clear PWM output during EPWM cycle or CPWM half cycle, where n=2. PEC1 is valid only when COMB1=1. This field is write protected, it can be written only when write protection is disabled. 0b: In complementary mode, during one EPWM cycle or CPWM half cycle, either CV(n) or CV(n+1) is used to set or clear channel(n) PWM output, channel(n+1) output is complementary with channel(n). 1b: In complementary mode, during one EPWM cycle or CPWM half cycle, one of CV(n) and CV(n+1) is used to set channel(n) PWM output, the other is used to clear channel(n) PWM output, channel(n+1) output is complementary with channel(n).	0x0
14 COMB1	RW	Enable the combination of channel(n) and channel(n+1), where n=2. When this pair of channels is combined, outputs of their channels are complementary. This field is write protected, it can be written only when write protection is disabled. 0b: The channel(n) and channel(n+1) are independent. 1b: The channel(n) and channel(n+1) are combined, channel(n) and channel(n+1) outputs are	0x0

Field	Access	Description	Reset Value
		complementary.	
15 RSVD	RO	Reserved and read as 0	0x0
16 FCTLEN2	RW	Enable the fault control in the channels (n) and (n+1), where n=4. This field is write protected. It can be written only when write protection is disabled. 0b: The fault control in this pair of channels is disabled. 1b: The fault control in this pair of channels is enabled.	0x0
17 DTEN2	RW	Enable the deadtime insertion in the channels (n) and (n+1), where n=4. This field is write protected. It can be written only when write protection is disabled. 0b: The deadtime insertion in this pair of channels is disabled. 1b: The deadtime insertion in this pair of channels is enabled.	0x0
18 SYNCEN2	RW	Enable the synchronization of CV(n) and CV(n+1) register value, where n=4. 0b: Synchronization in this pair of channels is disabled. 1b: Synchronization in this pair of channels is enabled.	0x0
19 ASYMEN2	RW	Enable the asymmetric PWM output in complementary CPWM output mode for channels(n) and (n+1), where n=4. This bit only makes sense when channel(n) and channel(n+1) are in complementary CPWM output mode. This field is write protected, it can only be written when write protection is disabled (WPDIS=1). 0b: Channel(n) and channel(n+1) are symmetric PWM output in complementary CPWM mode. 1b: Channel(n) and channel(n+1) are asymmetric PWM output in complementary CPWM mode.	0x0
20 CVSEL2	RW	This bit selects either CV(n) or CV(n+1) as channel value to generate PWM output in complementary mode, where n=4. CVSEL2 is valid only in complementary EPWM mode and complementary CPWM mode. This field is write protected, it can be written only when write protection is disabled. 0b: CV(n) is used to generate PWM output in complementary mode. 1b: CV(n+1) is used to generate PWM output in complementary mode.	0x0
21 PEC2	RW	In complementary mode, this bit defines whether one of CV(n)/CV(n+1) is or both CV(n)/CV(n+1) are used to set/clear PWM output during EPWM cycle or CPWM half cycle, where n=4. PEC2 is valid only when COMB2=1. This field is write protected, it can be written only when write protection is disabled. 0b: In complementary mode, during one EPWM cycle or CPWM half cycle, either CV(n) or CV(n+1) is used to set or clear channel(n) PWM output, channel(n+1) output is complementary with channel(n). 1b: In complementary mode, during one EPWM cycle or CPWM half cycle, one of CV(n) and CV(n+1) is used to set channel(n) PWM output, the other is used to clear channel(n) PWM output, channel(n+1) output is complementary with channel(n).	0x0

Field	Access	Description	Reset Value
22 COMB2	RW	Enable the combination of channel(n) and channel(n+1), where n=4. When this pair of channels is combined, outputs of their channels are complementary. This field is write protected, it can be written only when write protection is disabled. 0b: The channel(n) and channel(n+1) are independent. 1b: The channel(n) and channel(n+1) are combined, channel(n) and channel(n+1) outputs are complementary.	0x0
23 RSVD	RO	Reserved and read as 0	0x0
24 FCTLEN3	RW	Enable the fault control in the channels (n) and (n+1), where n=6. This field is write protected. It can be written only when write protection is disabled. 0b: The fault control in this pair of channels is disabled. 1b: The fault control in this pair of channels is enabled.	0x0
25 DTEN3	RW	Enable the deadtime insertion in the channels (n) and (n+1), where n=6. This field is write protected. It can be written only when write protection is disabled. 0b: The deadtime insertion in this pair of channels is disabled. 1b: The deadtime insertion in this pair of channels is enabled.	0x0
26 SYNCEN3	RW	Enable the synchronization of CV(n) and CV(n+1) register value, where n=6. 0b: Synchronization in this pair of channels is disabled. 1b: Synchronization in this pair of channels is enabled.	0x0
27 ASYMEN3	RW	Enable the asymmetric PWM output in complementary CPWM output mode for channels(n) and (n+1), where n=6. This bit only makes sense when channel(n) and channel(n+1) are in complementary CPWM output mode. This field is write protected, it can only be written when write protection is disabled (WPDIS=1). 0b: Channel(n) and channel(n+1) are symmetric PWM output in complementary CPWM mode. 1b: Channel(n) and channel(n+1) are asymmetric PWM output in complementary CPWM mode.	0x0
28 CVSEL3	RW	This bit selects either CV(n) or CV(n+1) as channel value to generate PWM output in complementary mode, where n=6. CVSEL3 is valid only in complementary EPWM mode and complementary CPWM mode. This field is write protected, it can be written only when write protection is disabled. 0b: CV(n) is used to generate PWM output in complementary mode. 1b: CV(n+1) is used to generate PWM output in complementary mode.	0x0
29 PEC3	RW	In complementary mode, this bit defines whether one of CV(n)/CV(n+1) is or both CV(n)/CV(n+1) are used to set/clear PWM output during EPWM cycle or CPWM half cycle, where n=6. PEC3 is valid only when COMB3=1. This field is write protected, it can be written only when write protection is disabled.	0x0

Field	Access	Description	Reset Value
		0b: In complementary mode, during one EPWM cycle or CPWM half cycle, either CV(n) or CV(n+1) is used to set or clear channel(n) PWM output, channel(n+1) output is complementary with channel(n). 1b: In complementary mode, during one EPWM cycle or CPWM half cycle, one of CV(n) and CV(n+1) is used to set channel(n) PWM output, the other is used to clear channel(n) PWM output, channel(n+1) output is complementary with channel(n).	
30 COMB3	RW	Enable the combination of channel(n) and channel(n+1), where n=6. When this pair of channels is combined, outputs of their channels are complementary. This field is write protected, it can be written only when write protection is disabled. 0b: The channel(n) and channel(n+1) are independent. 1b: The channel(n) and channel(n+1) are combined, channel(n) and channel(n+1) outputs are complementary.	0x0
31 RSVD	RO	Reserved and read as 0	0x0

29.2.14 MCPWM Fault Control Register (MCPWM_FLTCR)

29.2.14.1 Offset

Register	Offset
MCPWM_FLTCR	0x8C

29.2.14.2 Fields

Field	Access	Description	Reset Value
0 FIFEN0	RW	Enable the filter for fault input channel 0. This field is write protected, it can only be written when write protection is disabled. 0b: The filter for fault input channel 0 is disabled. 1b: The filter for fault input channel 0 is enabled.	0x0
1 FIFEN1	RW	Enable the filter for fault input channel 1. This field is write protected, it can only be written when write protection is disabled. 0b: The filter for fault input channel 1 is disabled. 1b: The filter for fault input channel 1 is enabled.	0x0
2 FIFEN2	RW	Enable the filter for fault input channel 2. This field is write protected, it can only be written when write protection is disabled. 0b: The filter for fault input channel 2 is disabled. 1b: The filter for fault input channel 2 is enabled.	0x0
3 FIFEN3	RW	Enable the filter for fault input channel 3. This field is write protected, it can only be written when write protection is disabled. 0b: The filter for fault input channel 3 is disabled. 1b: The filter for fault input channel 3 is enabled.	0x0
4	RW	Enable the fault input channel 0. This field is write	0x0

Field	Access	Description	Reset Value
FLTEN0		protected, it can only be written when write protection is disabled. 0b: The fault input channel 0 is disabled. 1b: The fault input channel 0 is enabled.	
5 FLTEN1	RW	Enable the fault input channel 1. This field is write protected, it can only be written when write protection is disabled. 0b: The fault input channel 1 is disabled. 1b: The fault input channel 1 is enabled.	0x0
6 FLTEN2	RW	Enable the fault input channel 2. This field is write protected, it can only be written when write protection is disabled. 0b: The fault input channel 2 is disabled. 1b: The fault input channel 2 is enabled.	0x0
7 FLTEN3	RW	Enable the fault input channel 3. This field is write protected, it can only be written when write protection is disabled. 0b: The fault input channel 3 is disabled. 1b: The fault input channel 3 is enabled.	0x0
8 FLTPOL0	RW	This field is write protected, it can only be written when write protection is disabled. 0b: The fault input channel 0 polarity is active high, a logic one at fault input indicates a fault event. 1b: The fault input channel 0 polarity is active low, a logic zero at fault input indicates a fault event.	0x0
9 FLTPOL1	RW	This field is write protected, it can only be written when write protection is disabled. 0b: The fault input channel 1 polarity is active high, a logic one at fault input indicates a fault event. 1b: The fault input channel 1 polarity is active low, a logic zero at fault input indicates a fault event.	0x0
10 FLTPOL2	RW	This field is write protected, it can only be written when write protection is disabled. 0b: The fault input channel 2 polarity is active high, a logic one at fault input indicates a fault event. 1b: The fault input channel 2 polarity is active low, a logic zero at fault input indicates a fault event.	0x0
11 FLTPOL3	RW	This field is write protected, it can only be written when write protection is disabled. 0b: The fault input channel 3 polarity is active high, a logic one at fault input indicates a fault event. 1b: The fault input channel 3 polarity is active low, a logic zero at fault input indicates a fault event.	0x0
12 FLTIE	RW	Enable the interrupt generation when a fault is detected and fault control is enabled. 0b: Fault detection interrupt was disabled. 1b: Fault detection interrupt was enabled.	0x0
13 FLTCM	RW	This bit defines when fault input is deasserted, the fault condition is manually cleared or automatically cleared. When the fault flag is set and fault input is deasserted, for automatically clear mode, the fault condition is automatically cleared and channel output is not disabled after fault input is deasserted; for manually clear mode, the user must clear the fault flag by software, otherwise the fault condition is not cleared and channel output is still disabled by previous fault	0x0

Field	Access	Description	Reset Value
		condition. This field is write protected, it can only be written when write protection is disabled. 0b: Manually fault clear mode. 1b: Automatically fault clear mode.	
15:14 RSVD	RO	Reserved and read as 0	0x0
19:16 FIFVAL	RW	Configure the filter value for fault inputs. The fault input filter is disabled when this field value is zero.	0x0
20 FSTATE	RW	This bit allows MCPWM output to be tri-stated when a fault event is detected. This field is write protected, it can only be written when write protection is disabled. 0b: When fault event is on-going, MCPWM output will be inactive state. 1b: When fault event is on-going, MCPWM output will be tri-stated.	0x0
31:21 RSVD	RO	Reserved and read as 0	0x0

29.2.15 MCPWM Fault Status Register (MCPWM_FLTSR)

29.2.15.1 Offset

Register	Offset
MCPWM_FLTSR	0x90

29.2.15.2 Fields

Field	Access	Description	Reset Value
0 FAULTF0	W0C	When the corresponding fault input is enabled and a fault condition is detected at the fault input, the corresponding fault flag was set. FAULTF0 was cleared by writing a 0 to FAULTF0 while there is no existing fault condition at the corresponding fault inputs. Writing a 1 to FAULTF0 has no effect. 0b: No fault event was detected at fault input channel 0. 1b: Fault event was detected at fault input channel 0.	0x0
1 FAULTF1	W0C	When the corresponding fault input is enabled and a fault condition is detected at the fault input, the corresponding fault flag was set. FAULTF1 was cleared by writing a 0 to FAULTF1 while there is no existing fault condition at the corresponding fault inputs. Writing a 1 to FAULTF1 has no effect. 0b: No fault event was detected at fault input channel 1. 1b: Fault event was detected at fault input channel 1.	0x0
2 FAULTF2	W0C	When the corresponding fault input is enabled and a fault condition is detected at the fault input, the corresponding fault flag was set. FAULTF2 was cleared by writing a 0 to FAULTF2 while there is no existing fault condition at the corresponding fault	0x0

Field	Access	Description	Reset Value
		inputs. Writing a 1 to FAULTF2 has no effect. 0b: No fault event was detected at fault input channel 2. 1b: Fault event was detected at fault input channel 2.	
3 FAULTF3	W0C	When the corresponding fault input is enabled and a fault condition is detected at the fault input, the corresponding fault flag was set. FAULTF3 was cleared by writing a 0 to FAULTF3 while there is no existing fault condition at the corresponding fault inputs. Writing a 1 to FAULTF3 has no effect. 0b: No fault event was detected at fault input channel 3. 1b: Fault event was detected at fault input channel 3.	0x0
4 FAULTFA	W0C	Represents the logic OR of the FAULTF0 and FAULTF1. FAULTFA was cleared by writing a 0 to FAULTFA while there is no existing fault condition at the enabled fault input 0 and fault input 1. Writing a 1 to FAULTFA has no effect. 0b: No fault event was detected at fault input 0 and fault input 1. 1b: At least one fault event was detected at fault input 0 and fault input 1.	0x0
5 FAULTFB	W0C	Represents the logic OR of the FAULTF2 and FAULTF3. FAULTFB was cleared by writing a 0 to FAULTFB while there is no existing fault condition at the enabled fault input 2 and fault input 3. Writing a 1 to FAULTFB has no effect. 0b: No fault event was detected at fault input 2 and fault input 3. 1b: At least one fault event was detected at fault input 2 and fault input 3.	0x0
6 FAULTINA	RO	Represents the logic OR of the enabled fault input 0 value and fault input 1 value after their filter (if their filter is enabled) when fault control is enabled. 0b: The logic OR of filtered fault input 0/1 value is 0. 1b: The logic OR of filtered fault input 0/1 value is 1.	0x0
7 FAULTINB	RO	Represents the logic OR of the enabled fault input 2 value and fault input 3 value after their filter (if their filter is enabled) when fault control is enabled. 0b: The logic OR of filtered fault input 2/3 value is 0. 1b: The logic OR of filtered fault input 2/3 value is 1.	0x0
31:8 RSVD	RO	Reserved and read as 0	0x0

29.2.16 MCPWM Fault A Safe Status Value Register (MCPWM_FLTASS)

29.2.16.1 Offset

Register	Offset
MCPWM_FLTASS	0x94

29.2.16.2 Function

This register defines the channel output safe status value when fault A (fault input 0 or fault input 1) is detected.

29.2.16.3 Fields

Field	Access	Description	Reset Value
0 FLTASS0	RW	This field is write protected, it can only be written when write protection is disabled (WPDIS=1). 0b: Channel 0 output safe value is 0 when fault A (fault input 0 or fault input 1) is detected. 1b: Channel 0 output safe value is 1 when fault A (fault input 0 or fault input 1) is detected.	0x0
1 FLTASS1	RW	This field is write protected, it can only be written when write protection is disabled (WPDIS=1). 0b: Channel 1 output safe value is 0 when fault A (fault input 0 or fault input 1) is detected. 1b: Channel 1 output safe value is 1 when fault A (fault input 0 or fault input 1) is detected.	0x0
2 FLTASS2	RW	This field is write protected, it can only be written when write protection is disabled (WPDIS=1). 0b: Channel 2 output safe value is 0 when fault A (fault input 0 or fault input 1) is detected. 1b: Channel 2 output safe value is 1 when fault A (fault input 0 or fault input 1) is detected.	0x0
3 FLTASS3	RW	This field is write protected, it can only be written when write protection is disabled (WPDIS=1). 0b: Channel 3 output safe value is 0 when fault A (fault input 0 or fault input 1) is detected. 1b: Channel 3 output safe value is 1 when fault A (fault input 0 or fault input 1) is detected.	0x0
4 FLTASS4	RW	This field is write protected, it can only be written when write protection is disabled (WPDIS=1). 0b: Channel 4 output safe value is 0 when fault A (fault input 0 or fault input 1) is detected. 1b: Channel 4 output safe value is 1 when fault A (fault input 0 or fault input 1) is detected.	0x0
5 FLTASS5	RW	This field is write protected, it can only be written when write protection is disabled (WPDIS=1). 0b: Channel 5 output safe value is 0 when fault A (fault input 0 or fault input 1) is detected. 1b: Channel 5 output safe value is 1 when fault A (fault input 0 or fault input 1) is detected.	0x0
6 FLTASS6	RW	This field is write protected, it can only be written when write protection is disabled (WPDIS=1). 0b: Channel 6 output safe value is 0 when fault A (fault input 0 or fault input 1) is detected. 1b: Channel 6 output safe value is 1 when fault A (fault input 0 or fault input 1) is detected.	0x0
7 FLTASS7	RW	This field is write protected, it can only be written when write protection is disabled (WPDIS=1). 0b: Channel 7 output safe value is 0 when fault A (fault input 0 or fault input 1) is detected. 1b: Channel 7 output safe value is 1 when fault A (fault input 0 or fault input 1) is detected.	0x0

Field	Access	Description	Reset Value
31:8 RSVD	RO	Reserved and read as 0	0x0

29.2.17 MCPWM Fault B Safe Status Value Register (MCPWM_FLTBSS)

29.2.17.1 Offset

Register	Offset
MCPWM_FLTBSS	0x98

29.2.17.2 Function

This register defines the channel output safe status value when fault B (fault input 2 or fault input 3) is detected.

29.2.17.3 Fields

Field	Access	Description	Reset Value
0 FLTBSS0	RW	This field is write protected, it can only be written when write protection is disabled (WPDIS=1). 0b: Channel 0 output safe value is 0 when fault B (fault input 2 or fault input 3) is detected. 1b: Channel 0 output safe value is 1 when fault B (fault input 2 or fault input 3) is detected.	0x0
1 FLTBSS1	RW	This field is write protected, it can only be written when write protection is disabled (WPDIS=1). 0b: Channel 1 output safe value is 0 when fault B (fault input 2 or fault input 3) is detected. 1b: Channel 1 output safe value is 1 when fault B (fault input 2 or fault input 3) is detected.	0x0
2 FLTBSS2	RW	This field is write protected, it can only be written when write protection is disabled (WPDIS=1). 0b: Channel 2 output safe value is 0 when fault B (fault input 2 or fault input 3) is detected. 1b: Channel 2 output safe value is 1 when fault B (fault input 2 or fault input 3) is detected.	0x0
3 FLTBSS3	RW	This field is write protected, it can only be written when write protection is disabled (WPDIS=1). 0b: Channel 3 output safe value is 0 when fault B (fault input 2 or fault input 3) is detected. 1b: Channel 3 output safe value is 1 when fault B (fault input 2 or fault input 3) is detected.	0x0
4 FLTBSS4	RW	This field is write protected, it can only be written when write protection is disabled (WPDIS=1). 0b: Channel 4 output safe value is 0 when fault B (fault input 2 or fault input 3) is detected. 1b: Channel 4 output safe value is 1 when fault B (fault input 2 or fault input 3) is detected.	0x0
5 FLTBSS5	RW	This field is write protected, it can only be written when write protection is disabled (WPDIS=1). 0b: Channel 5 output safe value is 0 when fault B (fault	0x0

Field	Access	Description	Reset Value
		input 2 or fault input 3) is detected. 1b: Channel 5 output safe value is 1 when fault B (fault input 2 or fault input 3) is detected.	
6 FLTBS6	RW	This field is write protected, it can only be written when write protection is disabled (WPDIS=1). 0b: Channel 6 output safe value is 0 when fault B (fault input 2 or fault input 3) is detected. 1b: Channel 6 output safe value is 1 when fault B (fault input 2 or fault input 3) is detected.	0x0
7 FLTBS7	RW	This field is write protected, it can only be written when write protection is disabled (WPDIS=1). 0b: Channel 7 output safe value is 0 when fault B (fault input 2 or fault input 3) is detected. 1b: Channel 7 output safe value is 1 when fault B (fault input 2 or fault input 3) is detected.	0x0
31:8 RSVD	RO	Reserved and read as 0	0x0

29.2.18 MCPWM Synchronization Register (MCPWM_SYNC)

29.2.18.1 Offset

Register	Offset
MCPWM_SYNC	0x9C

29.2.18.2 Fields

Field	Access	Description	Reset Value
0 SWTRIG	RW	Writing a 1 to this bit will generate one software trigger pulse to trigger register synchronization. SWTRIG bit will be automatically cleared after one system clock cycle pulse. 0b: Software trigger is not selected. 1b: Software trigger is selected.	0x0
1 SYNCOSWC	RW	Selects when the OUTSWCR register value is updated with its buffer value. 0b: OUTSWCR register is updated with its buffer value on all rising edges of the system clock. 1b: OUTSWCR register is updated with its buffer value only by PWM synchronization.	0x0
2 SWWRBUF	RW	Software trigger activates the registers that contain write buff (MODn, CVn, OUTSWCR, MODn_DITHER, CVn_DITHER) to be synchronized with their written buffer value. 0b: Software trigger does not activate registers synchronization with their written buffer value. 1b: Software trigger activates registers synchronization with their written buffer value.	0x0
3 SWRSTCNT	RW	Software trigger activates counter restart count from 0. 0b: Software trigger does not reset counter. 1b: Software trigger resets counter.	0x0
31:4 RSVD	RO	Reserved and read as 0	0x0

29.2.19 MCPWM Reload Register (MCPWM_RELOAD)

29.2.19.1 Offset

Register	Offset
MCPWM_RELOAD	0xA0

29.2.19.2 Function

This register controls the reload mechanisms, and enables the reload of the MODn, CVn, MODn_DITHER, CVn_DITHER registers with their write buffer value when a reload event happens. The reload event includes EPWM full cycle reload, CPWM full cycle reload, and CPWM half cycle reload.

29.2.19.3 Fields

Field	Access	Description	Reset Value
0 CPWMFCR0	RW	For up-down counting mode, when counter 0 changes from 0 to 1, it's full of CPWM cycle. This bit defines whether it is a reload opportunity for CPWM output when counter 0 changes from 0 to 1. 0b: Disable counter 0 CPWM full cycle reload. 1b: Enable counter 0 CPWM full cycle reload.	0x0
1 CPWMFCR1	RW	For up-down counting mode, when counter 1 changes from 0 to 1, it's full of CPWM cycle. This bit defines whether it is a reload opportunity for CPWM output when counter 1 changes from 0 to 1. 0b: Disable counter 1 CPWM full cycle reload. 1b: Enable counter 1 CPWM full cycle reload.	0x0
2 CPWMFCR2	RW	For up-down counting mode, when counter 2 changes from 0 to 1, it's full of CPWM cycle. This bit defines whether it is a reload opportunity for CPWM output when counter 2 changes from 0 to 1. 0b: Disable counter 2 CPWM full cycle reload. 1b: Enable counter 2 CPWM full cycle reload.	0x0
3 CPWMFCR3	RW	For up-down counting mode, when counter 3 changes from 0 to 1, it's full of CPWM cycle. This bit defines whether it is a reload opportunity for CPWM output when counter 3 changes from 0 to 1. 0b: Disable counter 3 CPWM full cycle reload. 1b: Enable counter 3 CPWM full cycle reload.	0x0
4 CPWMHCR0	RW	For up-down counting mode, when counter 0 changes from MOD0 to (MOD0-1), it's half of CPWM cycle. This bit defines whether it is a reload opportunity for CPWM output when counter changes from MOD0 to (MOD0-1). 0b: Disable counter 0 CPWM half cycle reload. 1b: Enable counter 0 CPWM half cycle reload.	0x0
5 CPWMHCR1	RW	For up-down counting mode, when counter 1 changes from MOD1 to (MOD1-1), it's half of CPWM cycle. This bit defines whether it is a reload opportunity for CPWM output when counter changes from MOD1 to (MOD1-1). 0b: Disable counter 1 CPWM half cycle reload. 1b: Enable counter 1 CPWM half cycle reload.	0x0

Field	Access	Description	Reset Value
6 CPWMHCR2	RW	For up-down counting mode, when counter 2 changes from MOD2 to (MOD2-1), it's half of CPWM cycle. This bit defines whether it is a reload opportunity for CPWM output when counter changes from MOD2 to (MOD2-1). 0b: Disable counter 2 CPWM half cycle reload. 1b: Enable counter 2 CPWM half cycle reload.	0x0
7 CPWMHCR3	RW	For up-down counting mode, when counter 3 changes from MOD3 to (MOD3-1), it's half of CPWM cycle. This bit defines whether it is a reload opportunity for CPWM output when counter changes from MOD3 to (MOD3-1). 0b: Disable counter 3 CPWM half cycle reload. 1b: Enable counter 3 CPWM half cycle reload.	0x0
8 LOADEN0	RW	Enables the loading of MOD0, MOD0_DITHER, CV0, CV1, CV0_DITHER and CV1_DITHER registers with their write buffer value when a reload event happens. This bit is automatically cleared after the reload is completed. 0b: Disable MOD0, MOD0_DITHER, CV0, CV1, CV0_DITHER and CV1_DITHER registers to load the write buffer value. 1b: Enable MOD0, MOD0_DITHER, CV0, CV1, CV0_DITHER and CV1_DITHER registers to load the write buffer value.	0x0
9 LOADEN1	RW	Enables the loading of MOD1, MOD1_DITHER, CV2, CV3, CV2_DITHER and CV3_DITHER registers with their write buffer value when a reload event happens. This bit is automatically cleared after the reload is completed. 0b: Disable MOD1, MOD1_DITHER, CV2, CV3, CV2_DITHER and CV3_DITHER registers to load the write buffer value. 1b: Enable MOD1, MOD1_DITHER, CV2, CV3, CV2_DITHER and CV3_DITHER registers to load the write buffer value.	0x0
10 LOADEN2	RW	Enables the loading of MOD2, MOD2_DITHER, CV4, CV5, CV4_DITHER and CV5_DITHER registers with their write buffer value when a reload event happens. This bit is automatically cleared after the reload is completed. 0b: Disable MOD2, MOD2_DITHER, CV4, CV5, CV4_DITHER and CV5_DITHER registers to load the write buffer value. 1b: Enable MOD2, MOD2_DITHER, CV4, CV5, CV4_DITHER and CV5_DITHER registers to load the write buffer value.	0x0
11 LOADEN3	RW	Enables the loading of MOD3, MOD3_DITHER, CV6, CV7, CV6_DITHER and CV7_DITHER registers with their write buffer value when a reload event happens. This bit is automatically cleared after the reload is completed. 0b: Disable MOD3, MOD3_DITHER, CV6, CV7, CV6_DITHER and CV7_DITHER registers to load the write buffer value. 1b: Enable MOD3, MOD3_DITHER, CV6, CV7,	0x0

Field	Access	Description	Reset Value
		CV6_DITHER and CV7_DITHER registers to load the write buffer value.	
31:12 RSVD	RO	Reserved and read as 0	0x0

29.2.20 MCPWM Reload Frequency Register (MCPWM_LDFREQ)

29.2.20.1 Offset

Register	Offset
MCPWM_LDFREQ	0xA4

29.2.20.2 Function

This register configures the PWM reload frequency. Reload event can happen periodically at every several reload opportunities.

29.2.20.3 Fields

Field	Access	Description	Reset Value
4:0 LDFREQ0	RW	This field defines the PWM reload frequency for counter 0. LDFREQ = n indicates RLDF0 is set every (n+1) reload opportunities, n ranges from 0 to 31.	0x0
7:5 RSVD	RO	Reserved and read as 0	0x0
12:8 LDFREQ1	RW	This field defines the PWM reload frequency for counter 1. LDFREQ = n indicates RLDF1 is set every (n+1) reload opportunities, n ranges from 0 to 31.	0x0
15:13 RSVD	RO	Reserved and read as 0	0x0
20:16 LDFREQ2	RW	This field defines the PWM reload frequency for counter 2. LDFREQ = n indicates RLDF2 is set every (n+1) reload opportunities, n ranges from 0 to 31.	0x0
23:21 RSVD	RO	Reserved and read as 0	0x0
28:24 LDFREQ3	RW	This field defines the PWM reload frequency for counter 3. LDFREQ = n indicates RLDF3 is set every (n+1) reload opportunities, n ranges from 0 to 31.	0x0
31:29 RSVD	RO	Reserved and read as 0	0x0

29.2.21 MCPWM MODn Dither Register (MCPWM_MODn_DITHER)

29.2.21.1 Offset

Register	Offset
MCPWM_MODn_DITHER	0x100+4*n (n=0~3)

29.2.21.2 Fields

Field	Access	Description	Reset Value
4:0 MOD_DITHER	RW	This value is used to control one pair of channel(2n) and channel(2n+1) PWM output period dithering, where n=0,1,2,3. See the chapter of period dithering for details.	0x0
31:5 RSVD	RO	Reserved and read as 0	0x0

29.2.22 MCPWM CVn Dither Register (MCPWM_CVn_DITHER)

29.2.22.1 Offset

Register	Offset
MCPWM_CVn_DITHER	0x110+4*n (n=0~7)

29.2.22.2 Fields

Field	Access	Description	Reset Value
4:0 CV_DITHER	RW	This value is used to control channel n PWM output edge dithering, where n=0,1,2,3,4,5,6,7. See the chapter of edge dithering for details.	0x0
31:5 RSVD	RO	Reserved and read as 0	0x0

29.3 Function description

This section provides the functional information of the MCPWM module.

NOTE: In this chapter, system clock represents core clock.

29.3.1 Time base unit

The MCPWM implements four independent 16-bit counters, and the clock sources of these counters are provided by the time base unit. The time base unit selects one clock source and this clock is divided by prescaler. Time base unit also determines the counter behavior in debug mode.

29.3.1.1 Clock source selection

The four counters will use the same counting clock which is selected by TIMEBASE[CKSRC] register. Three clock sources can be selected as counter clock, core clock, function clock and external clock from pad. It's not recommended to switch counter clock source when counter is counting.

- CKSRC=2'b00, no clock source is selected so counter is disabled. Writing 2'b00 to CKSRC will disable counter but not affect counter register value.

- CKSRC=2'b01, core clock source is selected.
- CKSRC=2'b10, function clock source is selected.
- CKSRC=2'b11, external clock source is selected.

The function clock source and external clock from pad are asynchronous clock source, so it must pass through a synchronizer clocked by the core clock to assure counter transitions are properly aligned to core clock transitions. Therefore, to meet Nyquist criteria and consider the jitter, when the function clock or external clock is selected as counter clock, the frequency of clock source must not exceed 1/4 of the core clock frequency.

NOTE:

In this SoC chip, there are three external clock sources from pads: TCLK0, TCLK1 and TCLK2. Any external clock source can be selected as MCPWM external clock source by configuring SCM_MISCTL2 register.

29.3.1.2 Prescaler description

The prescaler implements a 7-bit prescaler counter to divide selected counter clock by factor 1/2/4/8/16/32/64/128. This divided factor is selected by PSDIV[2:0] fields.

29.3.1.3 Debug mode

When the chip enters into debug mode, the counter behavior is determined by DBGm fields based on the following table.

Table 29-2 MCPWM behavior in debug mode

DBGm	Counter	CHF	Channel output
00	stopped	can be set	function mode
01	stopped	is not set	channel output is forced to safe value
10	stopped	is not set	channel output is frozen
11	running	can be set	function mode

29.3.2 Counter

The MCPWM implements four independent 16-bit counters 0/1/2/3. Each counter is enabled or disabled by individual control bit GLBCR[CNTE_N]. The overflow value of each counter is configured by individual MOD_N register.

29.3.2.1 Global counter

When GLBCR[GLBCNTE] bit is enabled, the global counter is enabled. Counter 0 is used as global counter, all channels select global counter as operating counter. Register CNT0 and MOD0 value will determine output of all channels. When global counter is enabled, GLBCR[CNTE_N] bit will not make sense, counter 1/2/3 are all off.

When GLBCR[GLBCNTE] bit is disabled, all channels use individual counter and overflow value.

- Counter 0 is used as channel 0/1 operating counter, MOD0 is the counter overflow value.
- Counter 1 is used as channel 2/3 operating counter, MOD1 is the counter overflow value.
- Counter 2 is used as channel 4/5 operating counter, MOD2 is the counter overflow value.
- Counter 3 is used as channel 6/7 operating counter, MOD3 is the counter overflow value.

value.

29.3.2.2 Counting mode

Each counter can count with two modes: up counting mode and up-down counting mode. The counting mode is selected by GLBCR[CNTMn] bit.

- CNTMn = 0, up counting mode for counter n
- CNTMn = 1, up-down counting mode for counter n

In up counting mode, the counter increments from 0 to MOD value, then restarts counting from 0 and generates counter overflow event. In up counting mode, the counter's full cycle period is counter clock period multiplied by (MOD+1).

In up-down counting mode, the counter increments from 0 to MOD value, then decrements to 0, and then restarts counting from 0. Counter overflow event is generated when counter decrements from MOD to (MOD-1). In up-down counting mode, the counter's full cycle period is counter clock period multiplied by 2xMOD.

When the counter is counting, updating MOD register will load the write value to its write buffer, MOD register value will be updated from its write buffer at proper point. See the chapter of register synchronization for details about register update.

29.3.3 Channels

The MCPWM implements 8 output channels. The GLBCR[CHOEn] bit controls channel(n) output enable. Only when channel output is enabled can MCPWM channel output be driven to pin.

MCPWM 8 channels are grouped as pairs of output channels for motor control, that is:

- Channel 0 and channel 1 are grouped as one pair of channels.
- Channel 2 and channel 3 are grouped as one pair of channels.
- Channel 4 and channel 5 are grouped as one pair of channels.
- Channel 6 and channel 7 are grouped as one pair of channels.

29.3.3.1 Channels polarity

MCPWM provides flexible configuration for channels' output polarity, and OUTCR[POLn] and CFGn[CPP] bits will both determine the PWM output status when counter matches CV(n) value.

The OUTCR[POLn] bit controls the channel(n) output polarity:

- POL = 0, the channel output polarity is active high, so the logic one is an active state and the logic zero is an inactive state.
- POL = 1, the channel output polarity is active low, so the logic one is an inactive state and the logic zero is an active state.

The CFGn[CPP] bit defines the channel(n) pulse polarity:

- CPP = 0, channel(n) output is low pulse polarity, channel(n) output is inactive state when counter value is more than CV(n) value, and channel(n) output is active state when counter value is less than CV(n) value.
- CPP = 1, channel(n) output is high pulse polarity, channel(n) output is active state when counter value is more than CV(n) value, and channel(n) output is inactive state when counter value is less than CV(n) value.

NOTE

In the following chapter description, if there is no special clarification, POL and CPP are set as 0 by default, so when counter is up counting, channel output is clear when counter matches CV value and it is set when counter reaches 0. When counter is down counting, channel output is set when counter matches CV value and it is clear when counter reaches 0. If either of POL and CPP is reverse, then the channel output behavior is reverse.

29.3.3.2 Channel output software control

The channel output can be controlled by software when OUTSWCR[CHnOC] is enabled. In this mode, channel(n) software control value CH(n)OCV is forced to channel(n) output.

- When pairs of channel(n) and channel(n+1) work in independent mode, that is COMB = 0, software output control directly forces CH(n)OCV value to channel(n) output and forces CH(n+1)OCV to channel(n+1) output, channel(n) and channel(n+1) output are independent.
- When pairs of channel(n) and channel(n+1) work in complementary mode (COMB = 1), software output control directly forces CH(n)OCV value to channel(n) output, but channel(n+1) value depends on CH(n)OCV and CH(n+1)OCV values. See the following table.

Table 29-3 Channel software output control when COMB = 1

CH(n)OCV	CH(n+1)OCV	Channel(n) output	Channel(n+1) output
0	0	force to 0	force to 0
0	1	force to 0	force to 1
1	x	force to 1	force to 0

NOTE

For software output control, POL(n) and POL(n+1) should both be 0. CH(n)OC and CH(n+1)OC bits should be the same and COMB bit should NOT be modified after software output control is enabled to avoid getting the unexpected result.

29.3.3.3 Channels output off status

The OUTCR[OFFVALn] bits control the channel(n) output status when its operating counter is off. Channels' operating counter is selected by GLBCR[GLBCNTEN]. The operating counter is off in either of the following conditions:

- CKSRC = 2'b00, counter clock is disabled.
- GLBCR[GLBCNTEN] = GLBCR[CENTENn] = 0, both global counter and individual counter are not enabled.

When channel(n) operating counter is off, this channel output status is only determined by OUTCR[OFFVALn], regardless of channel output polarity and software control.

- OFFVAL = 0, the channel output value is always 0 when counter is off.
- OFFVAL = 1, the channel output value is always 1 when counter is off.

29.3.4 PWM output mode

Pulse Width Modulation (PWM) mode allows generating an output signal with configurable frequency and duty cycle. When channels' software control is disabled and counter is not off, MCPWM channels will generate PWM output. MCPWM supports 7 PWM output modes. See the following table for selection of different PWM output modes.

Table 29-4 PWM output modes selection

COMB	PEC	CNTM	ASYMEN	PWM Output Mode
0	-	0	-	Independent Edge-aligned PWM(EPWM)
		1	-	Independent Center-aligned PWM(CPWM)
1	0	0	-	Complementary

COMB	PEC	CNTM	ASYMEN	PWM Output Mode
				Edge-aligned PWM(EPWM)
		1	0	Complementary Symmetric Center-aligned PWM
			1	Complementary Asymmetric Center-aligned PWM
	1	0	-	Complementary Variable Edge Placement PWM
		1	-	Complementary Double Switching PWM

In independent PWM output mode, for each channel, when counter matches its channel value CV(n), channel(n) PWM output status is determined by channel(n) output polarity POL(n) and pulse polarity CPP(n). See the chapter of channels polarity for details.

In complementary mode, for each pair of channels, when counter matches CV(n) or CV(n+1) value, even channel(n) PWM output status is determined by even channel output polarity POL(n) and pulse polarity CPP(n), odd channel(n+1) output is always complementary with even channel(n). Odd channel pulse polarity CPP(n+1) is not used, but odd channel output polarity will affect odd channel output status. Therefore, in complementary mode, to generate one pair of complementary PWM output, even channel polarity POL(n) must be the same with odd channel polarity POL(n+1). Otherwise, even channel and odd channel outputs are always the same.

29.3.4.1 Independent Edge-aligned PWM output mode

The independent edge-aligned PWM (EPWM) output mode is selected when:

- COMB = 0
- CNTM = 0

In independent EPWM output mode, counter counts in up counting mode. Counter increments from 0, when counter matches the CV value, channel output is set or clear. When counter reaches MOD value, counter restarts to increment from 0, channel output is clear or set. Therefore, for one pair of channels, they have individual CV values but share the same CNT and MOD value, so the PWM output edges of two channels are aligned when counter reaches 0. If the global counter is enabled, then PWM outputs of all channels are edge-aligned.

For channel(n), the EPWM output pulse width and period are shown as following. The cnt_clock_period represents the counter clock period.

- pulse_width = cnt_clock_period x CV(n)
- period = cnt_clock_period x (MOD+1).

When counter matches the CV(n) value, channel(n) flag CHF is set. See the following figure for EPWM output when POL=CPP=0.

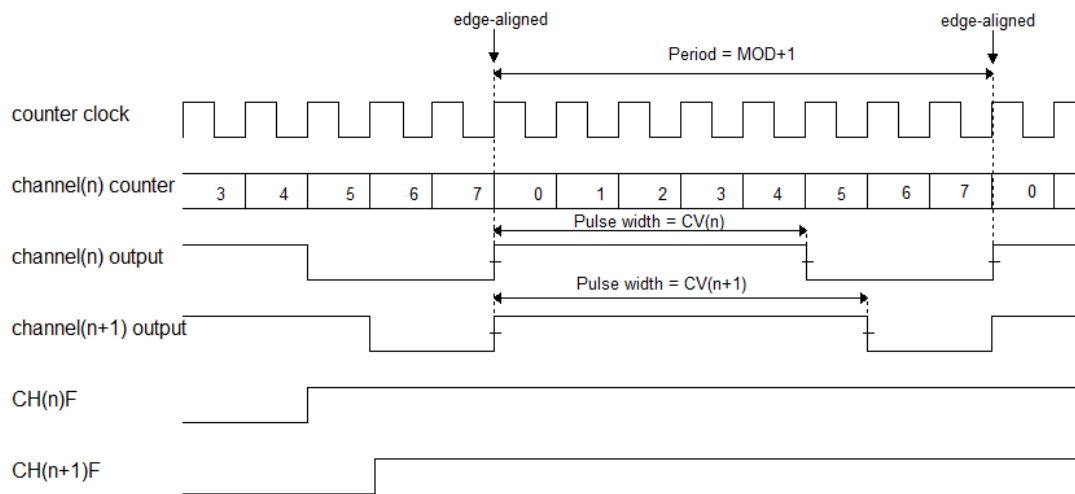


Figure 29-2 EPWM output (MOD=7, CV(n)=5, CV(n+1)=6)

For EPWM mode, the recommended configuration is $0 < CV(n) \leq MOD$. $CV(n)=0$ will generate 0% duty cycle EPWM output, $CV(n) > MOD$ will generate 100% duty cycle EPWM output.

29.3.4.2 Independent Center-aligned PWM output mode

The independent center-aligned PWM (CPWM) output mode is selected when:

- COMB = 0
- CNTM = 1

In independent CPWM output mode, counter counts in up-down counting mode. Counter increments from 0 in up counting. When counter matches the CV value, channel output is set or clear. When counter reaches MOD value, counter decrements from MOD in down counting. When counter matches the CV value, channel output is clear or set. When counter reaches 0, it restarts up counting for next cycle. Therefore, for one pair of channels, the PWM output pulse centers of two channels are aligned when counter reaches 0. If the global counter is enabled, then PWM outputs of all channels are center-aligned.

For channel(n), the CPWM output pulse width and period are shown as following.

- $\text{pulse\_width} = \text{cnt\_clock\_period} \times 2 \times CV(n)$
- $\text{period} = \text{cnt\_clock\_period} \times 2 \times MOD$.

No matter counter is counting up or counting down, when counter matches with CV(n) value, channel(n) flag CHF is set. See the following figure for CPWM output when POL=CPP=0.

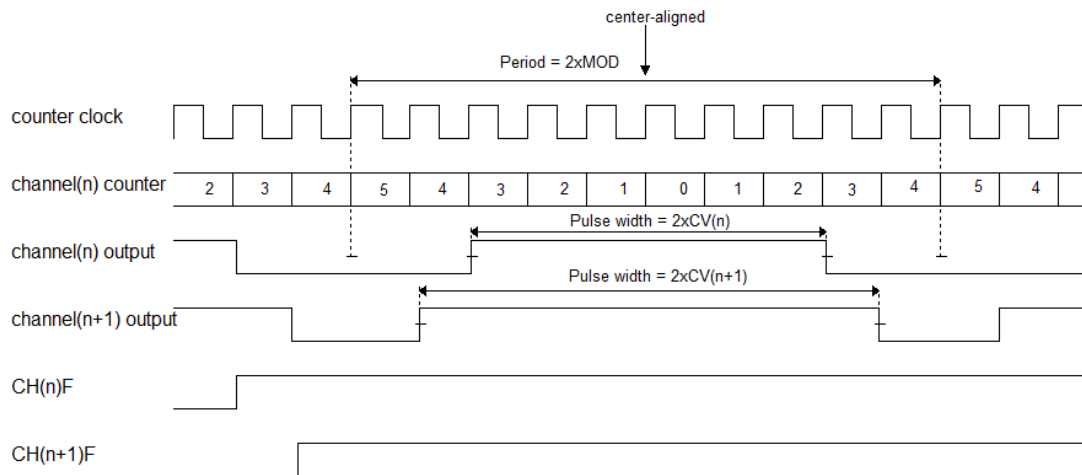


Figure 29-3 CPWM output (MOD=5, CV(n)=3, CV(n+1)=4)

For CPWM mode, the recommended configuration is $0 < CV(n) < MOD$. $CV(n)=0$ will generate 0% duty cycle CPWM output, $CV(n) \geq MOD$ will generate 100% duty cycle CPWM output.

29.3.4.3 Complementary Edge-aligned PWM output mode

The complementary edge-aligned PWM mode is selected when:

- COMB = 1
- PEC = 0
- CNTM = 0

In complementary EPWM mode, for each pair of channels, even channel(n) and odd channel(n+1) output are always complementary edge-aligned PWM. Either even channel value $CV(n)$ or odd channel value $CV(n+1)$ is used to generate PWM output. PCR[CVSELn] bit determines which channel value is selected:

- CVSELn = 0, even channel value $CV(n)$ is used to generate complementary PWM.
- CVSELn = 1, odd channel value $CV(n)$ is used to generate complementary PWM.

In this mode, counter counts in up counting mode. Even channel output is set or clear when counter matches the selected CV value, even channel output is clear or set when counter reaches 0. Odd channel output is always complementary with even channel.

For even channel(n), the EPWM output pulse width and period are shown as following. $CV(n)$ represents the selected channel value.

- Pulse_width = cnt_clock_period $\times$ $CV(n)$
- Period = cnt_clock_period $\times$ (MOD+1).

In this mode, no matter CVSELn is 0 or 1, even channel(n) flag is set only when counter matches even channel value $CV(n)$, odd channel(n+1) flag is set only when counter matches odd channel value $CV(n+1)$. See the following figure for complementary EPWM output when $POL(n)=POL(n+1)=CPP(n)=0$.

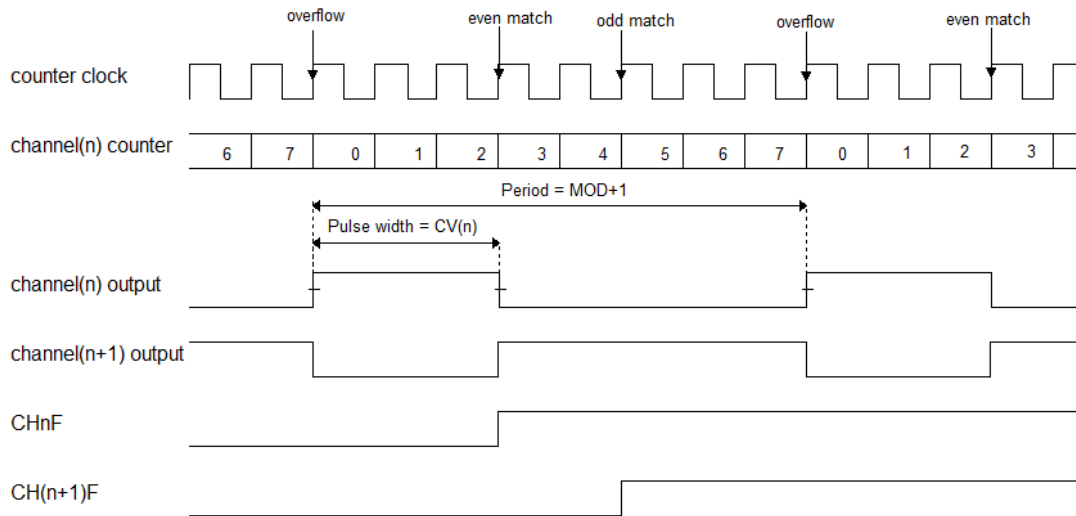


Figure 29-4 Complementary EPWM(MOD=7, CV(n)=3,CV(n+1)=5,CVSEL=0)

For complementary EPWM mode, the recommended configuration for selected CV is $0 < CV \leq MOD$. $CV=0$ will generate 0% duty cycle EPWM output for even channel, $CV > MOD$ will generate 100% duty cycle EPWM output for even channel. Odd channel is complementary with even channel.

29.3.4.4 Complementary symmetric center-aligned PWM output mode

The complementary symmetric center-aligned PWM mode is selected when:

- COMB = 1
- PEC = 0
- CNTM = 1
- ASYMEN = 0

In complementary symmetric CPWM mode, for each pair of channels, even channel(n) and odd channel(n+1) output are always complementary symmetric center-aligned PWM. Either even channel value CV(n) or odd channel value CV(n+1) is used to generate PWM output. PCR[CVSELn] bit determines which channel value is selected:

- CVSELn = 0, even channel value CV(n) is used to generate complementary PWM.
- CVSELn = 1, odd channel value CV(n) is used to generate complementary PWM.

In this mode, counter counts in up-down counting mode. Even channel output is set or clear when counter matches the selected CV value in up counting, even channel output is clear or set when counter matches the selected CV value in down counting. Odd channel output is always complementary with even channel.

For even channel(n), the CPWM output pulse width and period are shown as following. CV(n) represents the selected channel value.

- Pulse_width = cnt_clock_period $\times$ 2 $\times$ CV(n)
- Period = cnt_clock_period $\times$ 2 $\times$ MOD.

In this mode, no matter CVSELn is 0 or 1, even channel(n) flag is set only when counter matches even channel value CV(n), odd channel(n+1) flag is set only when counter matches odd channel value CV(n+1). See the following figure for complementary CPWM output when POL(n)=POL(n+1)=CPP(n)=0.

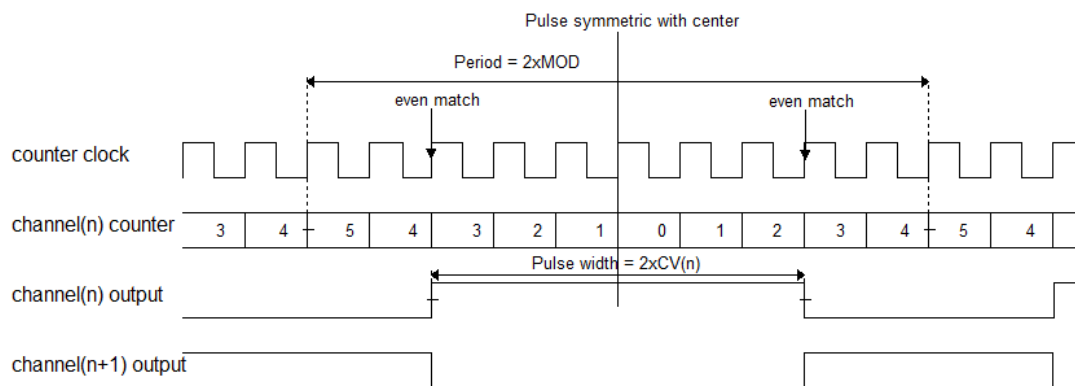


Figure 29-5 Complementary symmetric CPWM(MOD=5,CV(n)=3,CV(n+1)=2,CVSEL=0)

For complementary symmetric CPWM mode, the recommended configuration for selected CV is $0 < CV < MOD$. $CV=0$ will generate 0% duty cycle CPWM output for even channel, $CV \geq MOD$ will generate 100% duty cycle CPWM output for even channel. Odd channel output is always complementary with even channel.

29.3.4.5 Complementary asymmetric center-aligned PWM output mode

The complementary asymmetric center-aligned PWM mode is selected when:

- COMB = 1
- PEC = 0
- CNTM = 1
- ASYMEN = 1

In complementary asymmetric center-aligned PWM mode, for each pair of channels, even channel(n) and odd channel(n+1) output are always complementary PWM, but both of the two PWM outputs are asymmetric. Both even channel value CV(n) and odd channel value CV(n+1) are used to generate PWM output whatever CVSELn is.

In this mode, counter counts in up-down counting mode. Even channel output is set or clear when counter matches even channel value $CV(n)$ in up counting, even channel output is clear or set when counter matches odd channel value $CV(n+1)$. Odd channel output is always complementary with even channel. Therefore, each channel PWM output pulse is asymmetric.

For even channel(n), the PWM output pulse width and period are shown as following.

- $\text{Pulse\_width} = \text{cnt\_clock\_period} \times [\text{CV}(n) + \text{CV}(n+1)]$
- $\text{Period} = \text{cnt\_clock\_period} \times 2 \times \text{MOD}$

In this mode, even channel(n) flag is set when counter matches even channel value CV(n), odd channel(n+1) flag is set when counter matches odd channel value CV(n+1). See the following figure for complementary asymmetric PWM output when POL(n)=POL(n+1)=CPP(n)=0.

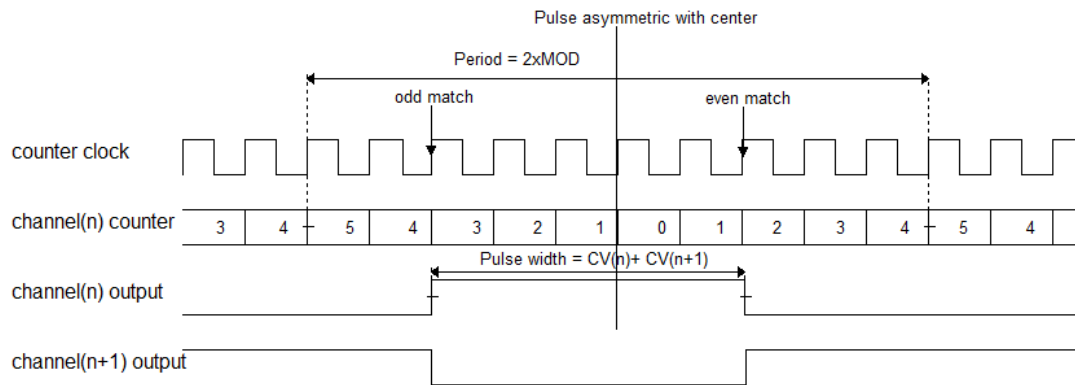


Figure 29-6 Complementary Asymmetric CPWM(MOD=5,CV(n)=2,CV(n+1)=3)

For complementary asymmetric center-aligned PWM mode, the recommended configuration is $0 < CV(n) < MOD$ and $0 < CV(n+1) < MOD$. $CV(n)=0$ and $CV(n+1) \geq MOD$ will generate 0% duty cycle PWM output for even channel, $CV(n+1)=0$ and $CV(n) \geq MOD$ will generate 100% duty cycle PWM output for even channel. Odd channel output is always complementary with even channel.

29.3.4.6 Complementary variable edge placement PWM output mode

The complementary variable edge placement PWM mode is selected when:

- COMB = 1
- PEC = 1
- CNTM = 0

In complementary variable edge placement PWM mode, the timing of both edges of the PWM output can be controlled by even channel value $CV(n)$ and odd channel value $CV(n+1)$.

In this mode, counter counts in up counting mode. Even channel output is set or clear when counter matches even channel value $CV(n)$ and clear or set when counter matches odd channel value $CV(n+1)$. Odd channel output is always complementary with even channel.

For even channel(n), the PWM output pulse width and period are shown as following.

- $Pulse\_width = cnt\_clock\_period \times [MOD+1+CV(n)-CV(n+1)]$
- $Period = cnt\_clock\_period \times (MOD+1)$

See the following figure for complementary variable edge placement PWM output when $POL(n)=POL(n+1)=CPP(n)=0$.

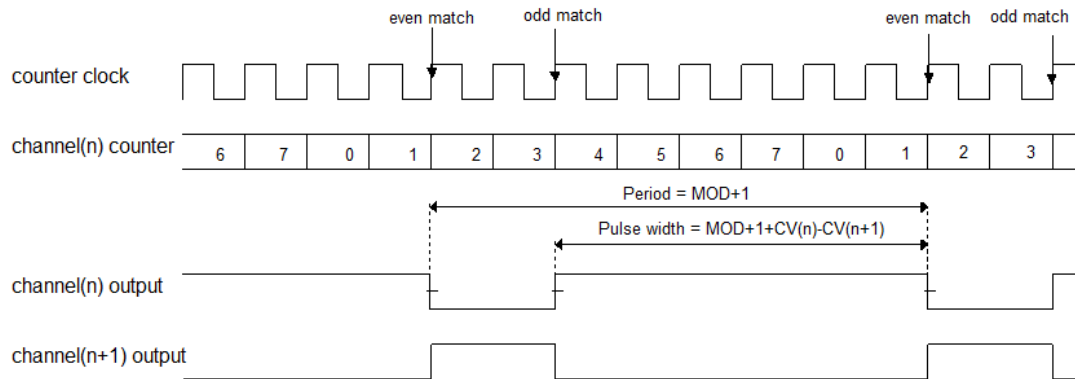


Figure 29-7 Complementary Variable Edge Placement PWM (MOD=7, CV(n)=2, CV(n+1)=4)

For complementary variable edge placement PWM mode, the recommended configuration is $0 < CV(n) < CV(n+1) < MOD$. $CV(n)=0$ and $CV(n+1) > MOD$ will generate 0% duty cycle PWM output for even channel, $0 < CV(n+1) \leq CV(n)$ will generate 100% duty cycle PWM output for even channel. Odd channel output is always complementary with even channel.

29.3.4.7 Complementary double switching PWM output mode

The complementary double switching PWM mode is selected when:

- COMB = 1
- PEC = 1
- CNTM = 1

In complementary double switching PWM mode, counter counts in up-down counting mode. In the first half of PWM cycle while counter is counting up, even channel output is set or clear when counter matches even channel value $CV(n)$ and clear or set when counter matches odd channel value $CV(n+1)$. In the second half of PWM cycle while counter is counting down, even channel output is set or clear when counter matches odd channel value $CV(n+1)$ and clear or set when counter matches even channel value $CV(n)$. Odd channel output is always complementary with even channel.

Therefore, MCPWM will generate two pulses during a full PWM cycle. The timing of both edges of the PWM output during a half PWM cycle can be controlled by even channel value $CV(n)$ and odd channel value $CV(n+1)$.

For even channel(n), the PWM output period and the width of two pulses are shown as following.

- $Pulse1\_width = cnt\_clock\_period \times 2 \times CV(n)$
- $Pulse2\_width = cnt\_clock\_period \times 2 \times [MOD - CV(n+1)]$
- $Period = cnt\_clock\_period \times 2 \times MOD$

See the following figure for complementary double switching PWM output when $POL(n)=POL(n+1)=CPP(n)=0$.

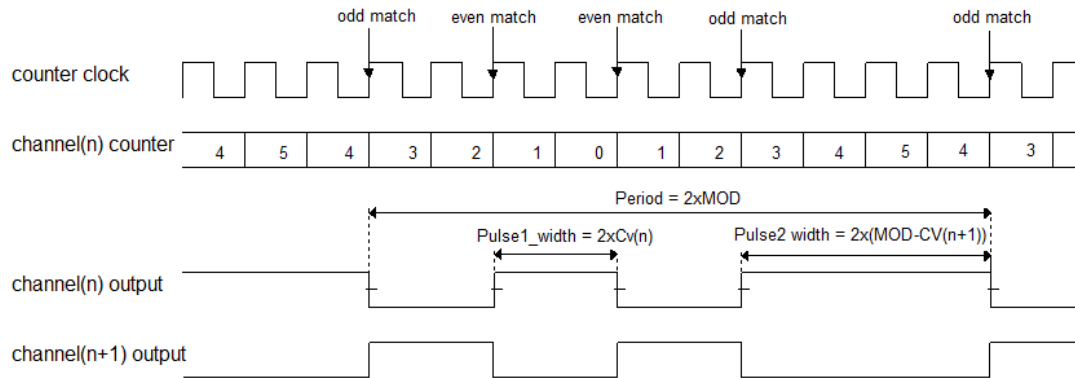


Figure 29-8 Complementary Double Switching PWM (MOD=5,CV(n)=1,CV(n+1)=3)

For complementary double switching PWM mode, the recommended configuration is $0 < \text{CV}(n) < \text{CV}(n+1) < \text{MOD}$.

29.3.5 Deadtime insertion

For PWM output application in motor control, when channel output switches from inactive to active state, it is allowed to keep channel output in inactive state for a short time before it is forced to active state. This inactive state is defined by channel polarity POL bit. This delay time is called deadtime delay region. For complementary PWM output, during the deadtime delay region, channel(n) and channel(n+1) outputs are both inactive. Deadtime delay ensures that no two complementary output signals drive the active state at the same time.

For each pair of channels, the deadtime insertion is enabled when $\text{DTENn} = 1$ and DTVALn is not zero. A 12-bit deadtime counter is implemented to realize this delay, this counter is clocked by system clock. When the deadtime counter matches the value of DTVALn , the delay is end. Therefore, the deadtime delay period is system clock period multiplied by DTVALn . See the following table for output status of complementary EPWM channels when deadtime insertion is enabled.

Table 29-5 Deadtime delay in complementary EPWM mode with $\text{POL}(n) = \text{POL}(n+1) = 0$

CPP(n)	Counter Reach 0		Counter Match CV(n)	
	channel(n) output	channel(n+1) output	channel(n) output	channel(n+1) output
0	delay before force to 1	force to 0 immediately	force to 0 immediately	delay before force to 1
1	force to 0 immediately	delay before force to 1	delay before force to 1	force to 0 immediately

From the table, it can be seen that when $\text{POL}(n) = \text{POL}(n+1) = 0$, channel(n) and channel(n+1) will delay when switching to active state logic one, but will not delay when switching to inactive state logic zero. If $\text{POL}(n) = \text{POL}(n+1) = 1$, channel(n) and channel(n+1) will delay when switching to logic zero and not delay when switching to logic 1. See the following figure for deadtime delay for complementary EPWM when $\text{POL}(n) = \text{POL}(n+1) = \text{CPP}(n) = 0$.

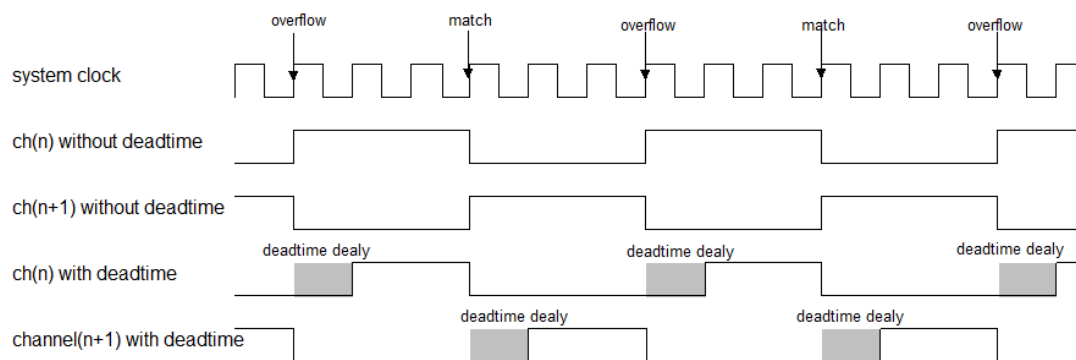


Figure 29-9 Deadtime delay for complementary EPWM(DTVALn=1)

29.3.6 Generation of match trigger and initial trigger

When the counter matches the value in CV(n) register, the MCPWM is allowed to generate a match trigger if channel(n) TRIGE is enabled. At the beginning of a PWM cycle, when the counter increments from 0 to 1, the MCPWM is allowed to generate an initial trigger if INITRIGE is enabled. The match trigger pulse width and initial trigger pulse width are both equal to one system clock period.

As MCPWM implements 8 independent channels and 4 independent counters, so MCPWM can generate 8 match triggers and 4 initial triggers. The triggers are used in SoC to trigger other modules.

The following figure shows the match trigger and initial trigger generation case.

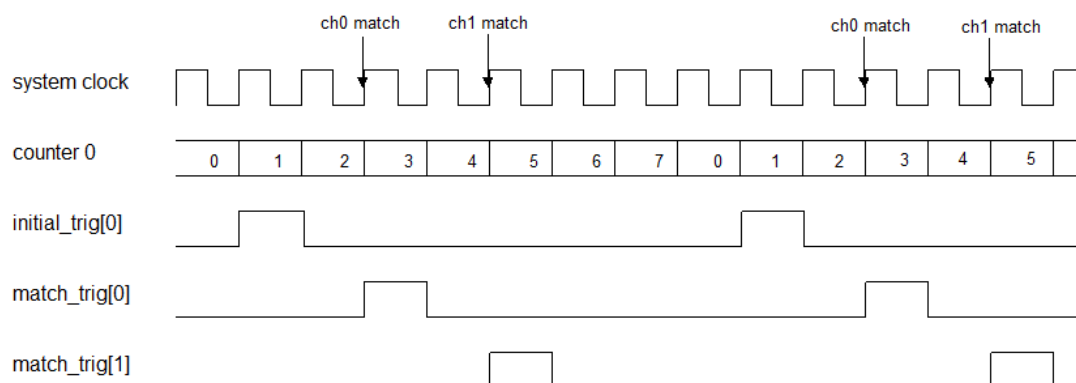


Figure 29-10 Match trigger and initial trigger generation of channel 0/1(CV(0)=3,CV(1)=5)

29.3.7 Fault protection

For motor control application, MCPWM implements four fault input channels. Fault input 0 and 1 are regarded as fault A, fault input 2 and 3 are regarded as fault B. When an active fault event is detected, then outputs of each pair of channels are allowed to force to safe status. See the following block diagram of fault input control.

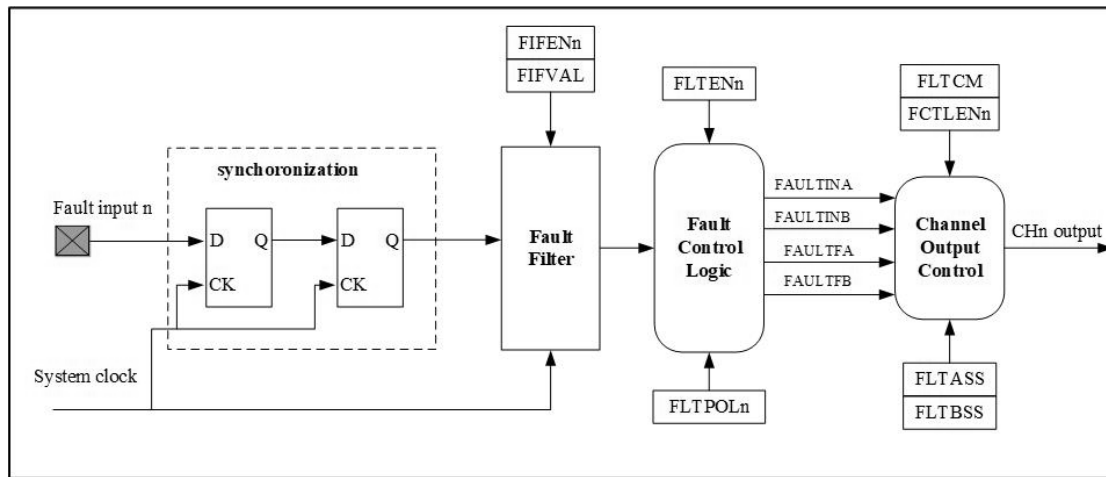


Figure 29-11 Fault input control

From the block figure, it can be seen that for fault input(n), it first passes through two flip-flops to be synchronized by system clock, then enters into fault input filter. Only when fault input channel(n) filter enables $FLTCR[FIFENn]$ is enabled and fault input filter value $FLTCR[FIFVAL]$ is not zero is this fault input channel filter enabled.

The fault input filter implements a 4-bit counter which is clocked by system clock. When a fault input edge is detected, the filter counter starts to count from 0 to $FIFVAL$ if fault input is stable, the counter will restart when fault input state changes. Therefore, any fault input pulse longer than $FIFVAL * (\text{system clock cycle})$ can pass through the filter, pulse shorter than $(FIFVAL - 1) * (\text{system clock cycle})$ will be regarded as a glitch and not pass to next stage, pulse between $FIFVAL * (\text{system clock cycle})$ and $(FIFVAL - 1) * (\text{system clock cycle})$ may pass through the filter. If fault input filter is not enabled, the synchronized fault input will directly pass to fault control logic.

In fault control logic, each fault input can be enabled by $FLTCR[FLTENn]$ and each fault input polarity can be configured by $FLTCR[FLTPOLn]$. When fault input(n) is enabled, an active state in filtered fault input(n) will be detected as a fault event. The fault A flag $FLTSR[FAULTFA]$ will set when a fault A event is detected and fault B flag $FLTSR[FAULTFB]$ will set when a fault B event is detected.

When a fault event is detected, for each pair of channels, the even channel(n) and odd channel(n+1) outputs are allowed to force to safe status when $PCR[FCTLENn]$ is enabled. Each channel output safe status is configurable independently when fault A or fault B event happens. $FLTASS$ register defines channels' safe status in fault A event and $FLTBSS$ register defines channels in fault B event. In this condition, channel output polarity $POL(n)$ does not affect channel output status.

When all fault input is deasserted, the channels PWM output recovery depends on fault clear mode $FLTCM$.

29.3.7.1 Automatical fault clear mode

When $FLTCR[FLTCM] = 1$, automatical fault clear mode is selected. After fault flag is set, the fault condition asserts. But this fault condition will be cleared automatically when all fault inputs are deasserted. Channel output resumes to generate PWM output from the next counter cycle when the fault condition is cleared automatically. In this mode, after fault input is deasserted, channel output automatically recovers but the fault flag is still set if it is not cleared by software. See the following figure for EPWM output with automatical fault clear mode.

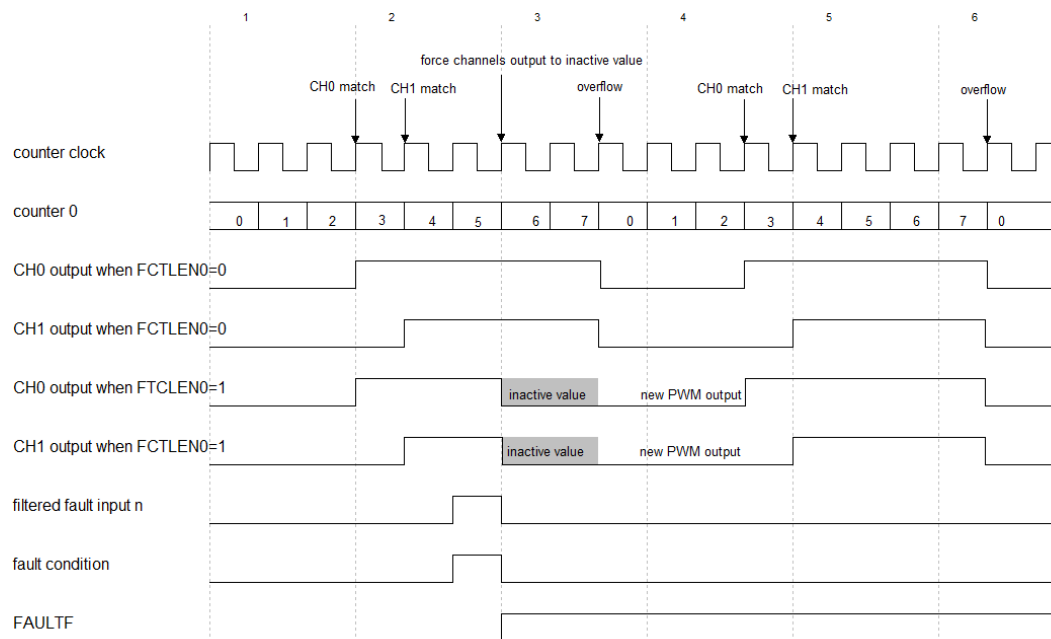


Figure 29-12 EPWM output for automatic fault clear mode

29.3.7.2 Manual fault clear mode

When FLTCM = 0, manual fault clear mode is selected. After fault flag is set, the fault condition is latched. If all fault flags are not cleared by software manually, this fault condition cannot be cleared even if all fault inputs are deasserted. Channels output resumes to generate PWM output from the next counter cycle when the fault condition is cleared manually. See the following figure for EPWM output with manual fault clear mode.

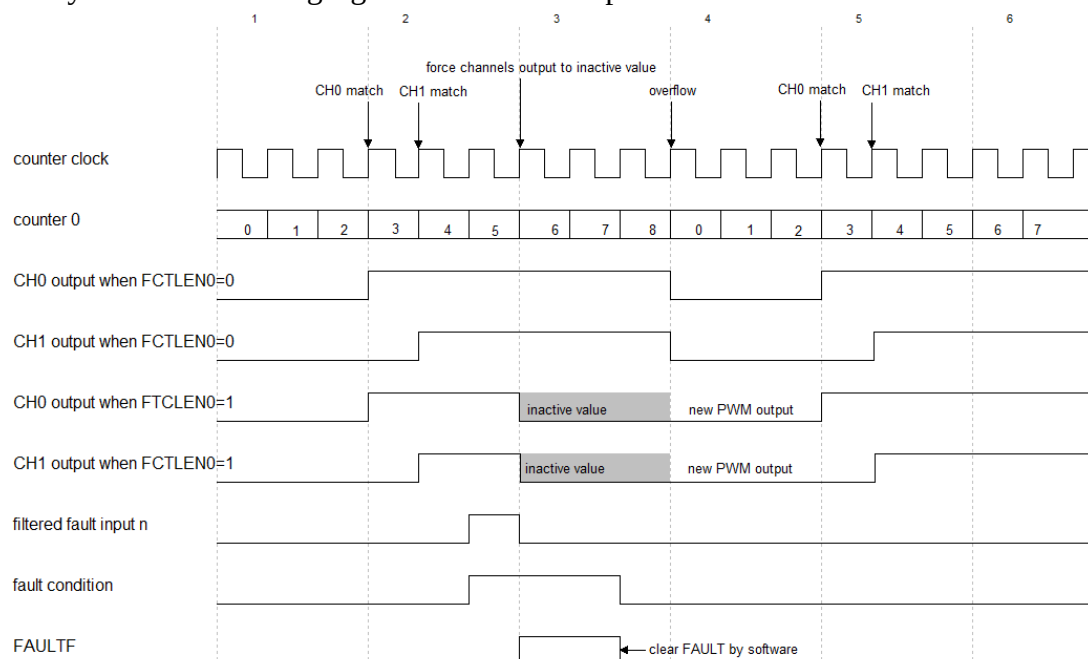


Figure 29-13 EPWM output for manual fault clear mode

29.3.7.3 Fault priority

Fault A has higher priority than fault B, which means that when fault A and fault B events are

both detected, channels' output status is determined by fault A safe status FLTASS. The configurable fault A and fault B safe status provides flexible application for users, for example, users can apply fault A as over-voltage protection and fault B as over-current protection.

29.3.8 Registers synchronization

The MCPWM implements write buffer for some critical registers, such as MOD(n), CV(n), MOD(n)_DITHER, CV(n)_DITHER and OUTSWCR. The write buffer can protect the current PWM output from being affected by software updating register. And the registers can synchronize from their write buffers at some certain points.

For each pair of channels, when its operating counter is off, software writing a value to these registers will directly update these registers' value. But when the operating counter is counting, writing a value to these registers will not update these registers' value immediately, but write the value to their write buffers instead. And these registers are allowed to be synchronized to their write buffer value by different synchronization strategies.

29.3.8.1 CNT(n) register synchronization

MCPWM counter(n) is allowed to restart counting from 0 at a certain point before counter overflows, which will restart generating a new PWM output. CNT(n) register can be synchronized to 0 by the following two methods:

1. Software writing a value to CNT(n) register.
2. Software trigger CNT(n) register reset when SYNC[SWRSTCNT]=1.

When counter(n) is counting, software writing a value to CNT(n) register will reset counter(n) and CNT(n) register value is synchronized to 0 on the next system clock rising edge.

For software trigger, when SYNC[SWRSTCNT] is enabled, writing a 1 to SYNC[SWTRIG] bit will generate one system clock cycle pulse software trigger to trigger CNT(n) register synchronization.

29.3.8.2 MOD(n) and MOD(n)_DITHER register synchronization

MOD(n) and MOD(n)_DITHER registers are allowed to update from their write buffers by two strategies according to different configurations:

1. Synchronization by software trigger when:
 - SYNC[SWRSTCNT] =1
 - SYNC[SWWRBUF]= 1
2. Synchronization by PWM reload when:
 - RELOAD[LOADENn] = 1

When configured as synchronization by software trigger, after MOD(n) or MOD(n)_DITHER register is written during counter(n) is counting, the write value is loaded into its write buffer. Then writing a 1 to SYNC[SWTRIG] bit will generate one system clock cycle pulse software trigger to trigger MOD(n) or MOD(n)_DITHER register synchronization. SWTRIG bit will be automatically cleared by hardware on the next rising edge of system clock. After SWTRIG asserts, MOD(n) or MOD(n)_DITHER register will update from its write buffer and counter restarts to count. See the following figure.

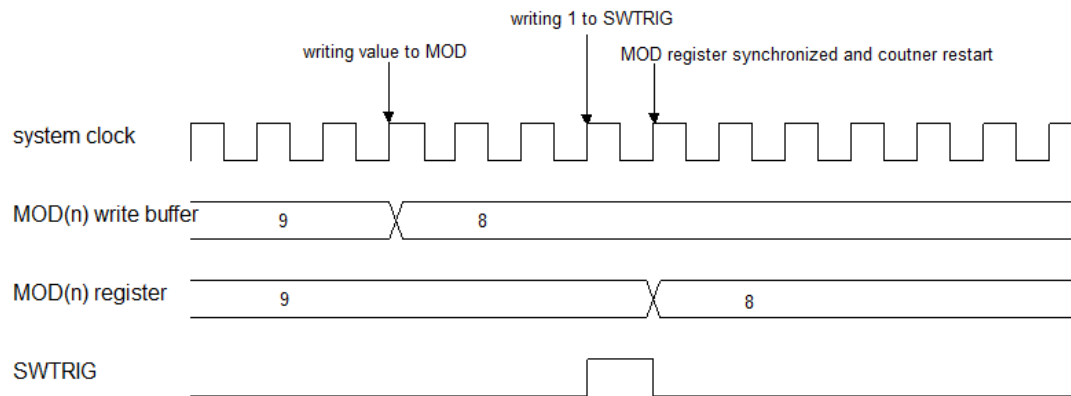


Figure 29-14 MOD register synchronization by software trigger

When configured as synchronization by PWM reload, after MOD(n) or MOD(n)_DITHER register is written during counter(n) is counting, the write value is loaded into its write buffer and MOD(n) or MOD(n)_DITHER register will update from its write buffer at the following reload opportunities:

1. EPWM full cycle reload opportunity in up counting mode. That is when counter(n) reaches the value of MOD(n).
2. CPWM half cycle reload opportunity in up-down counting mode. That is counter(n) reaches the value of MOD(n) when counting up.
3. CPWM full cycle reload opportunity in up-down counting mode. That is counter(n) reaches the value of 0 when counting down.

When counter operates in up-down counting mode, MCPWM is allowed to reload at both of the CPWM half cycle and full cycle reload opportunity, or reload only at one opportunity. It is determined by RELOAD[CPWMHCRn] and RELOAD[CPWMFCRn]. See the following table.

Table 29-6 CPWM reload opportunity

CPWMHCR(n)	CPWMFCR(n)	Is half cycle point a reload opportunity?	Is full cycle point a reload opportunity?
0	0	Yes	No
0	1	No	Yes
1	0	Yes	No
1	1	Yes	Yes

NOTE

In EPWM full cycle reload and CPWM full cycle reload, at the first period beginning (when counter first starts to count) and CNT(n) synchronization point, they are also reload opportunities.

When counter(n) reaches the reload opportunity, the reload flag RLDF(n) can be set. But RLDF(n) is not set at every reload opportunity, it depends on the configured reload frequency LDFREQ[LDFREQn] fields. LDFREQ(n) = x indicates RLDF(n) will be set every (x+1) reload opportunities occur. LDFREQ(n) default value is zero, which means RLDF(n) is set at every reload opportunity. When PWM reload is enabled, MOD(n) register completes the reload from its write buffer when reload flag RLDF(n) is set.

Note: For MOD(n) and MOD(n)_DITHER register synchronization by software trigger, it is

not necessary to enable SWRSTCNT bit, and disabling this bit can also finish register synchronization by software trigger. BUT we recommend users to enable this bit for MOD(n) register sync, because updating MOD(n) register will change the output PWM period, counter should be reset to restart counting; otherwise users may get the unexpected PWM output.

29.3.8.3 CV(n) and CV(n)_DITHER register synchronization

CV(n) and CV(n)_DITHER registers are allowed to update from their write buffers by two strategies according to different configurations:

1. Synchronization by software trigger when:

- PCR[SYNCENn] = 1
- SYNC[SWRSTCNT] = 1
- SYNC[SWWRBUF] = 1

2. Synchronization by PWM reload when:

- PCR[SYNCENn] = 1
- RELOAD[LOADENn] = 1

The CV(n) and CV(n)_DITHER register PWM software trigger and PWM reload mechanism are the same with MOD(n) register. SYNCENn (where n=0, 1, 2, 3) bit enables one pair of channel CV and CV_DITHER registers synchronization.

Note: For CV(n) and CV(n)_DITHER register synchronization by software trigger, it is not necessary to enable SWRSTCNT bit, and disabling this bit can also finish register synchronization by software trigger. BUT we recommend users to enable this bit for CV(n) register sync, because updating CV(n) register will change the output PWM duty, counter should reset to restart counting; otherwise users may get the unexpected PWM output

29.3.8.4 OUTSWCR register synchronization

OUTSWCR register is allowed to update from its write buffer by two strategies according to different configurations:

1. Synchronization at the next system clock after OUTSWCR was written when:

- SYNC[SYNCOSWC] = 0

2. Synchronization by software trigger when:

- SYNC[SYNCOSWC] = 1
- SYNC[SWWRBUF] = 1

When OUTSWCR register is configured as synchronization at the next system clock, after OUTSWCR register is written, the write value is loaded into its write buffer, and OUTSWCR register will update from its write buffer on the next rising edge of system clock.

OUTSWCR register synchronization by software trigger mechanism is also the same with MOD(n) register.

Synchronization by PWM reload is not supported for OUTSWCR register.

29.3.9 Dithering

MCPWM implements a fractional delay to achieve fine resolution on the generated PWM signals using dithering. MCPWM provides two kinds of dithering: PWM period dithering and edge dithering.

29.3.9.1 PWM period dithering

Each MCPWM_MODn_DITHER register implemented a write buffer register.

When counter n is off, it means that counter is disabled or counter clock is disabled. Writing MCPWM_MODn_DITHER register will directly update this register value, and writing MCPWM_MODn register will reset MCPWM_MODn_DITHER register value.

When counter n is on, it means that counter is enabled and counter clock is enabled. Writing MCPWM_MODn_DITHER register will load the write value into its write buffer. And Writing MCPWM_MODn register will reset MCPWM_MODn_DITHER write buffer register value. MCPWM_MODn_DITHER register is allowed to be synchronized to its write buffer value by synchronization strategy.

The even channel(n) and odd channel(n+1) PWM period dithering is enabled when MOD(n)_DITHER value is not zero. For every counter, a 5-bit accumulator is implemented to fulfil the dithering. This accumulator is reset to 0 when:

- Counter(n) is off
- MOD(n) register is synchronized from its write buffer
- CNT(n) register is synchronized to 0

After PWM period dithering is enabled, at the end of each PWM period, the MOD(n)_DITHER value is added to the internal 5-bit accumulator. When this accumulator overflows, that is, the result of adding is greater than 0x1F, at this PWM period, the counter(n) overflow value will be MOD(n)+1. And the accumulator remains with the rest of the subtraction, the result of this adding minus 0x20.

Therefore, for up counting mode, one unit of counter cycle is added to the end of the current dithering PWM period. For up-down counting mode, one unit of counter cycle is added to the end of current dithering PWM period and the beginning of next PWM period.

See the following two figures for EPWM and CPWM period dithering.

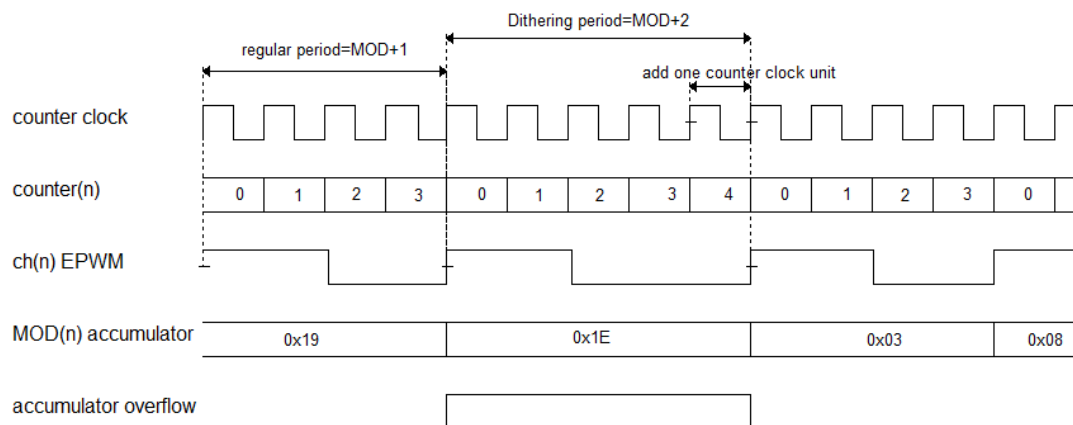


Figure 29-15 EPWM period dithering(MOD(n)=3,MOD(n)_DITHER=5)

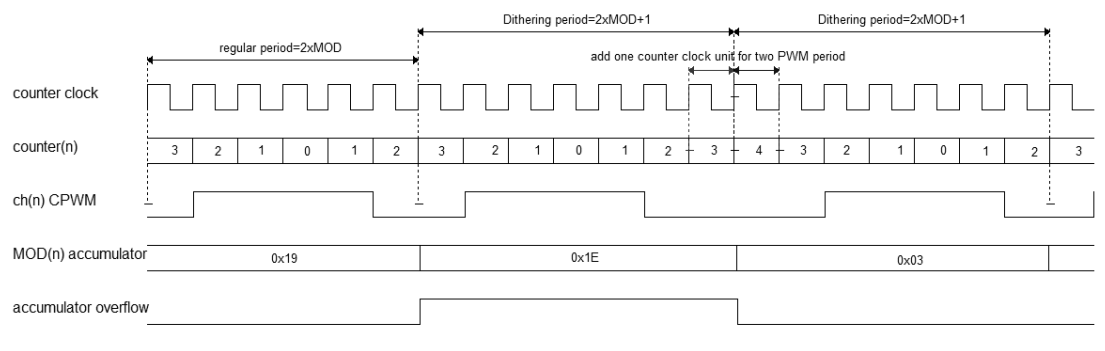


Figure 29-16 CPWM period dithering(MOD(n)=3,MOD(n)_DITHER=5)

29.3.9.2 PWM edge dithering

Each MCPWM_CVn_DITHER register implemented a write buffer register.

When counter n is off, it means that counter is disabled or counter clock is disabled. Writing MCPWM_CVn_DITHER register will directly update this register value, and writing MCPWM_CVn register will reset MCPWM_CVn_DITHER register value.

When counter n is on, it means that counter is enabled and counter clock is enabled. Writing MCPWM_CVn_DITHER register will load the write value into its write buffer. And Writing MCPWM_CVn register will reset MCPWM_CVn_DITHER write buffer register value. MCPWM_CVn_DITHER register is allowed to be synchronized to its write buffer value by synchronization strategy.

The channel(n) PWM edge dithering is enabled when CV(n)_DITHER value is not zero. Each channel implements a 5-bit accumulator to fulfil the channel output edge dithering. This accumulator is reset to 0 when:

- Counter(n) is off
- CV(n) register is synchronized from its write buffer

After PWM edge dithering is enabled, at the end of each PWM period, the CV(n)_DITHER value is added to the internal 5-bit accumulator. When this accumulator overflows, that is, the result of adding is greater than 0x1F, at this PWM period, the counter(n) match value will be CV(n)+1. And the accumulator remains with the rest of the subtraction, the result of this adding minus 0x20.

Therefore, for up counting mode, one unit of counter cycle is added to pulse duty cycle of the current dithering PWM period. For up-down counting mode, two units of counter cycles are added to the pulse duty cycle of current dithering PWM period.

See the following two figures for EPWM and CPWM period dithering.

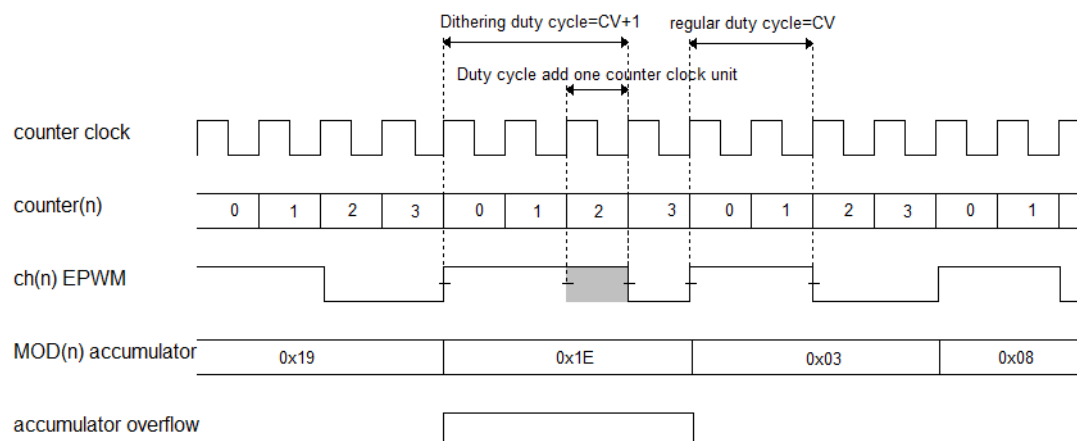


Figure 29-17 EPWM edge dithering($CV(n)=2, CV(n)\_DITHER=5$)

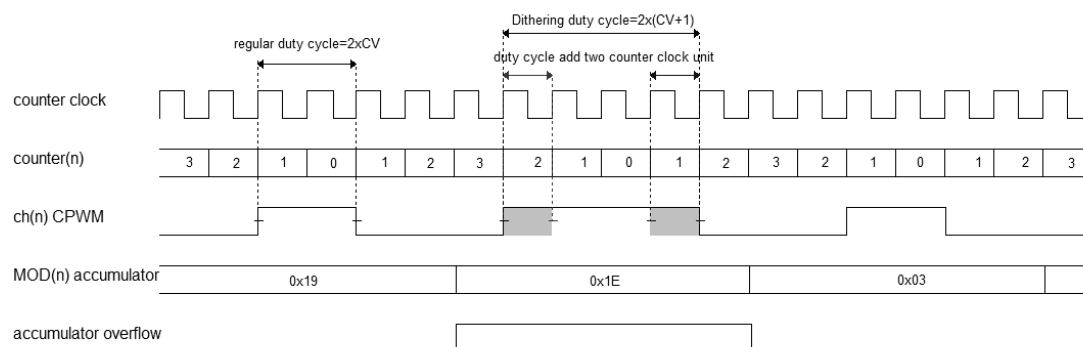


Figure 29-18 CPWM edge dithering($CV(n)=1, CV(n)\_DITHER=5$)

29.4 Interrupt and DMA signals

29.4.1 DMA

Every MCPWM channel supports to generate a DMA request to activate DMA transfer. Channel(n) DMA request is enabled when channel(n) CFG[DMA] and CFG[CHIE] are both enabled, after channel(n) channel flag CHF_n is asserted, the channel(n) DMA request is generated. For any channel(n), its channel flag is set when its operating counter matches CV(n) value.

When the channel(n) DMA request is asserted, channel(n) flag CHF_n is cleared automatically after this channel DMA transfer is done, then DMA request is deasserted.

29.4.2 Interrupt

MCPWM can generate the following four different interrupts:

- Timer overflow interrupt. This interrupt is generated when GLBCR[TOIE_n] is enabled and GLBSR[TOF_n] is asserted.
- Channel interrupt. This interrupt is generated when CFG_n[CHIE] is enabled but CFG_n[DMA] is disabled, and CFG_n[CHF] is asserted.
- Fault interrupt. This interrupt is generated when FLTCR[FLTIE] is enabled and

- FLTSR[FAULTFn] is asserted.
- Reload interrupt. This interrupt is generated when GLBCR[RLDIEn] is enabled and GLBSR[RLDFn] is asserted.

Chapter 30

Timer (TIM)

30.1 Introduction

The TIM block provides a 16-bit counter to support an 8-channel operation. Every channel can be selected as either an input channel or an output channel.

- The input channel supports the input capture function.
- The output channel supports the output compare function and can generate PWM output signals.

The 8 channels can work independently or be combined as pairs to work in complementary mode. Dead-time delay insertion ensures that no two complementary output signals drive the active state at the same time.

TIM supports two fault inputs control. When fault input event is detected in fault input, TIM can force channel outputs to inactive state or tri-state to ensure safety state.

The TIM contains an input filter to filter the input glitch. For example, it can filter the glitch in input channel 0 ~ channel 3, and filter the glitch in fault inputs.

30.1.1 Features

The TIM supports the following key features:

- Time base:
 - Support multi-input clock sources, core clock, function clock and external clock from pads
 - Prescaler divided the clock by 1/2/4/8/16/32/64/128
- Counter:
 - One integrated 16-bit counter
 - Support two counting modes: up and up-down counting
 - Support programmable overflow value
 - Support programmable initial counter value
- Every channel is independent, and can work in four different modes: input capture mode, output compare mode, center-aligned PWM mode and edge-aligned PWM mode
- Input capture mode:
 - The input pulse can be captured in rising edge, falling edge, or both edges
 - An input filter can be selected for channel 0~3
 - Dual-edge capture for pulse and period width measurement
- Output compare mode:
 - The output signal can be set, cleared, or toggled when the counter matches the configured value
 - Generation of match triggers and initial triggers
 - The polarity of each channel is configurable
- PWM output mode:
 - Center-aligned PWM output
 - Edge-aligned PWM output
 - Software control of PWM outputs
- Pairs of channels:
 - All channels can be combined into pairs, and each pair has independent control of both edges of generated PWM signal
 - Each pair of channels can operate in complementary mode

- Motor control:
 - Deadtime insertion is available for each complementary pair
 - Up to 2 fault inputs for global fault control
- Register access protection:
 - Synchronized loading of write buffered registers
 - Half cycle and full-cycle register reload capacity
 - Write protection for critical registers
 - Write buffer for match and counter registers
- Interrupt in case of the following:
 - Generation of interrupt per channel
 - Timer overflow interrupt
 - Fault condition has been detected
 - Register reload point occurs

30.1.2 Block diagram

The following figure is a block diagram of the Timer (TIM) module.

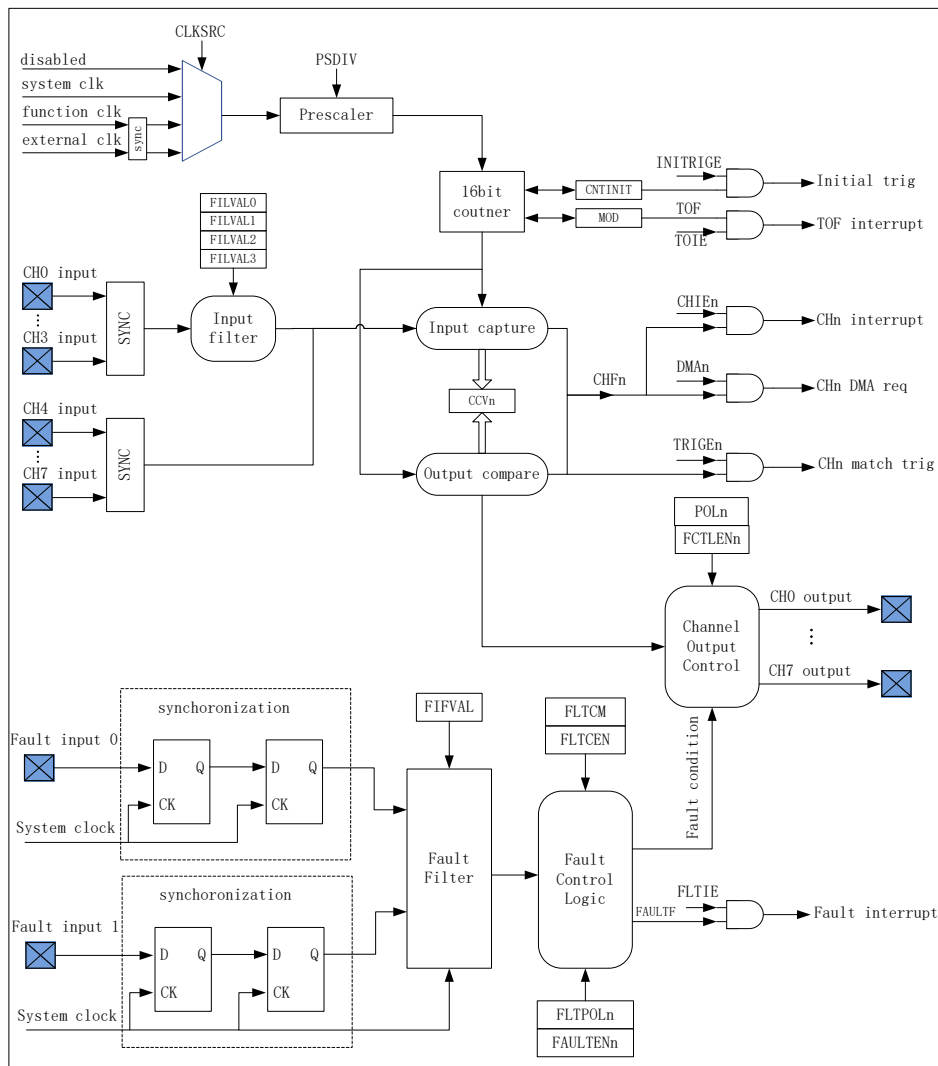


Figure 30-1 TIM block diagram

30.1.3 Instantiation and channel information

The following table lists the TIM instantiation and channel information for each chip in the product series.

Configuration	K144MC	K146MC	K148MC
Instance(s)	TIM0	TIM0	TIM0
	TIM1	TIM1	TIM1
	TIM2	TIM2	TIM2
	TIM3	TIM3	TIM3
Channel(s)	8	8	8

30.2 Signal description

Each TIM module supports up to 8 channels. Every channel can be selected as an input channel or an output channel. The input channel supports the input capture function. The output channel supports the output compare function and can generate PWM output signals. The 8 channels can work independently or be combined as pairs to work in complementary mode.

Each TIM module supports 2 fault inputs. When fault input event is detected in fault input, TIM can force channel outputs to the inactive state or tri-state to ensure a safe state.

Table 30-1 TIM signal descriptions

Signal	Description
Channeln, n= 0,...,7	Every channel can be selected as an input channel or an output channel
FLTn, n= 0,1	Fault control input

30.3 Memory map and register definition

This section provides the memory map and register definitions of the TIM module.

30.3.1 Memory map

TIM0 base address: 0x4000C000

TIM1 base address: 0x4000D000

TIM2 base address: 0x40018000

TIM3 base address: 0x40019000

Table 30-2 TIM memory map

Offset	Register	Width (In bits)	Reset Value
0x00	TIM Time Base Register (TIM_TIMEBASE)	32	0x00000000
0x04	TIM Counter Register (TIM_CNT)	32	0x00000000
0x08	TIM Counter Initial Value Register (TIM_CNTINIT)	32	0x00000000
0x0C	TIM Counter Modulo Value Register (TIM_MOD)	32	0x00000000
0x10	TIM Global Control Register (TIM_GLBCCR)	32	0x00000000
0x14	TIM Global Status Register (TIM_GLBSTR)	32	0x00000400
0x18 + 4*n	TIM Channel n Mode Configuration	32	0x00000000

Offset	Register	Width (In bits)	Reset Value
(n=0~7)	register (TIM_CMCn)		
0x38 + 4*n (n=0~7)	TIM Channel n Capture/Compare Value Register (TIM_CCVn)	32	0x00000000
0x58	TIM Output Control Register (TIM_OUTCR)	32	0x00000000
0x5C	TIM Channel Output Software Control register (TIM_OUTSWCR)	32	0x00000000
0x60	TIM Pairs of Channel Control Register (TIM_PCR)	32	0x00000000
0x64	TIM Input Filter Register (TIM_FILTER)	32	0x00000000
0x68	TIM Fault Control Register (TIM_FLTCR)	32	0x00000000
0x6C	TIM Fault Status Register (TIM_FLTSR)	32	0x00000000
0x70	TIM Synchronization Register (TIM_SYNC)	32	0x00000000
0x74	TIM Reload Register (TIM_RELOAD)	32	0x00000000
0x78	TIM Half Cycle Reload Value Register (TIM_HCV)	32	0x00000000

30.3.2 TIM Time Base Register (TIM_TIMEBASE)

30.3.2.1 Offset

Register	Offset
TIM_TIMEBASE	0x00

30.3.2.2 Function

This register selects the counter clock source and prescaler divider factors. It also determines the counter behavior in debug mode.

30.3.2.3 Fields

Field	Access	Description	Reset Value
1:0 CKSRC	RW	Clock source Select the TIM counter clock source. This field is write protected. It can only be written when write protection is disabled by setting WPDIS to 1. 00b: TIM counter is disabled 01b: Core clock is selected as TIM counter clock 10b: Function clock is selected as TIM counter clock 11b: The external clock is selected as TIM counter clock	0x0
4:2 PSDIV	RW	Prescaler divider Select one divide factor to divide the clock selected by CКСRC. This field is write protected. It can only be written when write protection is disabled by setting WPDIS to 1.	0x0

Field	Access	Description	Reset Value
		000b: Divide by 1 001b: Divide by 2 010b: Divide by 4 011b: Divide by 8 100b: Divide by 16 101b: Divide by 32 110b: Divide by 64 111b: Divide by 128	
6:5 DBGM	RW	Debug mode This field selects the counter behavior in chip debug mode.	0x0
31:7 RSVD_31_7	RO	Reserved	0x0

30.3.3 TIM Counter Register (TIM_CNT)

30.3.3.1 Offset

Register	Offset
TIM_CNT	0x04

30.3.3.2 Function

This is the counter register.

30.3.3.3 Fields

Field	Access	Description	Reset Value
15:0 CNT	RW	This field is the 16-bit counter value. Write to this field will load the counter with the CNTINIT value. Reset will clear the counter value.	0x0
31:16 RSVD_31_16	RO	Reserved	0x0

30.3.4 TIM Counter Initial Value Register (TIM_CNTINIT)

30.3.4.1 Offset

Register	Offset
TIM_CNTINIT	0x08

30.3.4.2 Function

The initial value is stored in the register. The CNTINIT register contains the initial value for the TIM counter. Writing to the CNTINIT register latches the value into its write buffer, see the section of CNTINIT register synchronization for more details.

When TIM counter is enabled by selecting a counter clock source, the TIM counter first starts from the value 0x0000, not CNTINIT value. If user expects the timer to count from CNTINIT value, not zero, when timer is enabled, any value should be written to CNT register to load the CNTINIT value in the write buffer after writing the expected CNTINIT value into CNTINIT register.

30.3.4.3 Fields

Field	Access	Description	Reset Value
15:0 CNTINIT	RW	The 16-bit counter initial value is programmable, not fixed to 16'h0000. Writing a value to this field will latch the write value into a write buffer.	0x0
31:16 RSVD_31_16	RO	Reserved	0x0

30.3.5 TIM Counter Modulo Value Register (TIM_MOD)

30.3.5.1 Offset

Register	Offset
TIM_MOD	0x0C

30.3.5.2 Function

The modulo value is stored in this register.

30.3.5.3 Fields

Field	Access	Description	Reset Value
15:0 MOD	RW	When the 16-bit counter reaches the MOD value and increments, the timer overflow flag TOF asserts. Writing a value to this field will latch the write value into a write buffer.	0x0
31:16 RSVD_31_16	RO	Reserved	0x0

30.3.6 TIM Global Control Register (TIM_GLBCR)

30.3.6.1 Offset

Register	Offset
TIM_GLBCR	0x10

30.3.6.2 Function

The register is a global control register.

30.3.6.3 Fields

Field	Access	Description	Reset Value
0 CHOE0	RW	Channel 0 output enable 0b: Channel 0 output port was disabled. 1b: Channel 0 output port was enabled. NOTE When the input mode was selected, this bit must be cleared.	0x0
1 CHOE1	RW	Channel 1 output enable 0b: Channel 1 output port was disabled.	0x0

Field	Access	Description	Reset Value
		1b: Channel 1 output port was enabled. NOTE When the input mode was selected, this bit must be cleared.	
2 CHOE2	RW	Channel 2 output enable 0b: Channel 2 output port was disabled. 1b: Channel 2 output port was enabled. NOTE When the input mode was selected, this bit must be cleared.	0x0
3 CHOE3	RW	Channel 3 output enable 0b: Channel 3 output port was disabled. 1b: Channel 3 output port was enabled. NOTE When the input mode was selected, this bit must be cleared.	0x0
4 CHOE4	RW	Channel 4 output enable 0b: Channel 4 output port was disabled. 1b: Channel 4 output port was enabled. NOTE When the input mode was selected, this bit must be cleared.	0x0
5 CHOE5	RW	Channel 5 output enable 0b: Channel 5 output port was disabled. 1b: Channel 5 output port was enabled. NOTE When the input mode was selected, this bit must be cleared.	0x0
6 CHOE6	RW	Channel 6 output enable 0b: Channel 6 output port was disabled. 1b: Channel 6 output port was enabled. NOTE When the input mode was selected, this bit must be cleared.	0x0
7 CHOE7	RW	Channel 7 output enable 0b: Channel 7 output port was disabled. 1b: Channel 7 output port was enabled. NOTE When the input mode was selected, this bit must be cleared.	0x0
8 RLDIE	RW	Reload interrupt enable 0b: Disable reload interrupt. 1b: Enable reload interrupt.	0x0
9 TOIE	RW	Timer overflow interrupt enable 0b: Disable timer overflow interrupt. 1b: Enable timer overflow interrupt.	0x0
10 CNTMODE	RW	16-bit counter counting mode This field is write protected. It can only be written when write protection is disabled by setting WPDIS to 1. 0b: TIM counter operates in up counting mode. 1b: TIM counter operates in up-down counting mode.	0x0
11 INIT	WO	Initialize the channels output Writing a 1 to INIT bit will initialize the channel output according to their corresponding INIT bit in the OUTCR register. Writing a 0 to INIT bit has no effect. The INIT bit is always read as 0.	0x0

Field	Access	Description	Reset Value
31:12 RSVD_31_12	RO	Reserved	0x0

30.3.7 TIM Global Status Register (TIM_GLBSR)

30.3.7.1 Offset

Register	Offset
TIM_GLBSR	0x14

30.3.7.2 Function

This register indicates the global status.

30.3.7.3 Fields

Field	Access	Description	Reset Value
0 CH0F	R/W0C	Channel 0 flag CH0F bit is the mirror of TIM_CMC0[CHF] bit. This flag is cleared by writing 0 to TIM_CMC0[CHF] after reading TIM_CMC0[CHF] while the TIM_CMC0[CHF] bit is set. Writing a 1 to TIM_CMC0[CHF] has no effect. 0b: No channel event occurred in channel 0. 1b: A channel event occurred in channel 0.	0x0
1 CH1F	R/W0C	Channel 1 flag CH1F bit is the mirror of TIM_CMC1[CHF] bit. This flag is cleared by writing 0 to TIM_CMC1[CHF] after reading TIM_CMC1[CHF] while TIM_CMC1[CHF] bit is set. Writing a 1 to TIM_CMC1[CHF] has no effect. 0b: No channel event occurred in channel 1. 1b: A channel event occurred in channel 1.	0x0
2 CH2F	R/W0C	Channel 2 flag CH2F bit is the mirror of TIM_CMC2[CHF] bit. This flag is cleared by writing 0 to TIM_CMC2[CHF] after reading TIM_CMC2[CHF] while TIM_CMC2[CHF] bit is set. Writing a 1 to TIM_CMC2[CHF] has no effect. 0b: No channel event occurred in channel 2. 1b: A channel event occurred in channel 2.	0x0
3 CH3F	R/W0C	Channel 3 flag CH3F bit is the mirror of TIM_CMC3[CHF] bit. This flag is cleared by writing 0 to TIM_CMC3[CHF] after reading TIM_CMC3[CHF] while TIM_CMC3[CHF] bit is set. Writing a 1 to TIM_CMC3[CHF] has no effect. 0b: No channel event occurred in channel 3. 1b: A channel event occurred in channel 3.	0x0
4 CH4F	R/W0C	Channel 4 flag CH4F bit is the mirror of TIM_CMC4[CHF] bit. This flag is cleared by writing 0 to TIM_CMC4[CHF] after reading TIM_CMC4[CHF] while TIM_CMC4[CHF] bit is set. Writing a 1 to TIM_CMC4[CHF] has no effect. 0b: No channel event occurred in channel 4. 1b: A channel event occurred in channel 4.	0x0
5 CH5F	R/W0C	Channel 5 flag CH5F bit is the mirror of TIM_CMC5[CHF] bit. This flag is	0x0

Field	Access	Description	Reset Value
		cleared by writing 0 to TIM_CMC5[CHF] after reading TIM_CMC5[CHF] while TIM_CMC5[CHF] bit is set. Writing a 1 to TIM_CMC5[CHF] has no effect. 0b: No channel event occurred in channel 5. 1b: A channel event occurred in channel 5.	
6 CH6F	R/W0C	Channel 6 flag CH6F bit is the mirror of TIM_CMC6[CHF] bit. This flag is cleared by writing 0 to TIM_CMC6[CHF] after reading TIM_CMC6[CHF] while TIM_CMC6[CHF] bit is set. Writing a 1 to TIM_CMC6[CHF] has no effect. 0b: No channel event occurred in channel 6. 1b: A channel event occurred in channel 6.	0x0
7 CH7F	R/W0C	Channel 7 flag CH7F bit is the mirror of TIM_CMC7[CHF] bit. This flag is cleared by writing 0 to TIM_CMC7[CHF] after reading TIM_CMC7[CHF] while TIM_CMC7[CHF] bit is set. Writing a 1 to TIM_CMC7[CHF] has no effect. 0b: No channel event occurred in channel 7. 1b: A channel event occurred in channel 7.	0x0
8 RLDF	R/W0C	Reload flag Set by hardware, RLDF is asserted when TIM counter reaches the reload point configured by RELOAD register. RLDF is cleared by software writing a 0 to this bit after reading RLDF flag while RLDF is set. Writing a 1 to RLDF has no effect. 0b: The counter did not reach the reload point. 1b: The counter reached the reload point.	0x0
9 TOF	R/W0C	Timer overflow flag Set by hardware, TOF is asserted when TIM counter reaches the MOD value and continues to increment. TOF is cleared by software writing a 0 to this bit after reading TOF flag while TOF is set. Writing a 1 to TOF has no effect. 0b: A timer overflow did not occur. 1b: A timer overflow occurred.	0x0
10 WPDIS	RW	Write protection disable This bit indicates whether the write-protected bits can be written or not. The WPDIS bit is a negation of the WPEN bit. WPDIS is cleared when 1 is written to WPEN, WPDIS is set when WPEN bit is read as 1 and then 1 is written to WPDIS. Writing 0 to WPDIS has no effect. 0b: A write protection is enabled. 1b: A write protection is disabled.	0x1
31:11 RSVD_31_11	RO	Reserved	0x0

30.3.8 TIM Channel n Mode Configuration Register (TIM_CMCn)

30.3.8.1 Offset

Register	Offset
TIM_CMCn	0x18 + 0x4*n (n = 0~7)

30.3.8.2 Function

TIM_CMCn register configures channel n work mode, enables channel n interrupt and DMA request, and determines whether the counter is reset or not when the capture event occurs.

30.3.8.3 Fields

Field	Access	Description	Reset Value
1:0 ELS	RW	Edge or level select CMS and ELS fields determine the channel work mode. See the channel mode chapter for details.	0x0
3:2 CMS	RW	Channel mode select CMS and ELS fields determine the channel work mode. See the channel mode chapter for details.	0x0
4 CHIE	RW	Channel interrupt enable 0b: Disable this channel interrupt. 1b: Enable this channel interrupt.	0x0
5 CHF	R/W0C	Channel flag Set by hardware when an event occurs in this channel. CHF is cleared by writing a 0 to the CHF bit after reading CHF while it is set. Writing a 1 to CHF has no effect. 0b: No Channel event occurred in this channel. 1b: A channel event occurred in this channel.	0x0
6 DMA	RW	DMA enable 0b: DMA transfer for this channel is disabled. 1b: DMA transfer for this channel is enabled. NOTE DMA request is only available if channel interrupt is also enabled.	0x0
7 ICRST	RW	Input capture reset TIM counter reset is driven by the selected event of the channel n in the input capture mode. This field is write protected. It can be written only when WPDIS equals 1. 0b: TIM counter is not reset when the selected channel input event is detected. 1b: TIM counter is reset when the selected channel input event is detected.	0x0
31:8 RSVD_31_8	RO	Reserved	0x0

30.3.9 TIM Channel n Capture/Compare Value Register (TIM_CCVn)

30.3.9.1 Offset

Register	Offset
TIM_CCVn	0x38+0x4*n (n=0~7)

30.3.9.2 Function

TIM_CCVn register value has different meanings in different channel modes. When a channel is selected as an input channel, including input capture mode and dual-edge capture mode, the TIM_CCVn register value represents the channel n capture value, which

can NOT be written by software. Only when a capture event occurs will the current counter value be automatically captured and loaded into the TIM_CCvN register.

When a channel is selected as an output channel, the TIM_CCvN register value represents the channel n compare value, which is programmed by software. When a counter matched with the TIM_CCvN value, a channel n match event occurs.

30.3.9.3 Fields

Field	Access	Description	Reset Value
15:0 CCV	RW	This value is the TIM counter capture value in input mode, or the counter compares the value in output mode for this channel.	0x0
31:16 RSVD_31_16	RO	Reserved	0x0

30.3.10 TIM Output Control Register (TIM_OUTCR)

30.3.10.1 Offset

Register	Offset
TIM_OUTCR	0x58

30.3.10.2 Function

This is the output control register.

30.3.10.3 Fields

Field	Access	Description	Reset Value
0 INIT0	RW	Channel 0 initialization value Select the value that was forced into channel 0 output when initialization occurred. 0b: Channel 0 output initial value is inactive state. 1b: Channel 0 output initial value is active state.	0x0
1 INIT1	RW	Channel 1 initialization value Select the value that was forced into channel 1 output when initialization occurred. 0b: Channel 1 output initial value is inactive state. 1b: Channel 1 output initial value is active state.	0x0
2 INIT2	RW	Channel 2 initialization value Select the value that was forced into channel 2 output when initialization occurred. 0b: Channel 2 output initial value is inactive state. 1b: Channel 2 output initial value is active state.	0x0
3 INIT3	RW	Channel 3 initialization value Select the value that was forced into channel 3 output when initialization occurred. 0b: channel 3 output initial value is inactive state. 1b: channel 3 output initial value is active state.	0x0
4 INIT4	RW	Channel 4 initialization value Select the value that was forced into channel 4 output when initialization occurred. 0b: Channel 4 output initial value is inactive state. 1b: Channel 4 output initial value is active state.	0x0

Field	Access	Description	Reset Value
5 INIT5	RW	Channel 5 initialization value Select the value that was forced into channel 5 output when initialization occurred. 0b: Channel 5 output initial value is inactive state. 1b: Channel 5 output initial value is active state.	0x0
6 INIT6	RW	Channel 6 initialization value Select the value that was forced into channel 6 output when initialization occurred. 0b: Channel 6 output initial value is inactive state. 1b: Channel 6 output initial value is active state.	0x0
7 INIT7	RW	Channel 7 initialization value Select the value that was forced into channel 7 output when initialization occurred. 0b: Channel 7 output initial value is inactive state. 1b: Channel 7 output initial value is active state.	0x0
8 POL0	RW	Channel 0 output polarity This bit defines the channel 0 active state in the output mode. This field is write protected. It can only be written when the write protection is disabled. 0b: Channel 0 output polarity is active high. 1b: Channel 0 output polarity is active low.	0x0
9 POL1	RW	Channel 1 output polarity This bit defines the channel 1 active state in the output mode. This field is write protected. It can only be written when the write protection is disabled. 0b: Channel 1 output polarity is active high. 1b: Channel 1 output polarity is active low.	0x0
10 POL2	RW	Channel 2 output polarity This bit defines the channel 2 active state in the output mode. This field is write protected. It can only be written when the write protection is disabled. 0b: Channel 2 output polarity is active high. 1b: Channel 2 output polarity is active low.	0x0
11 POL3	RW	Channel 3 output polarity This bit defines the channel 3 active states in the output mode. This field is write protected. It can only be written when the write protection is disabled. 0b: Channel 3 output polarity is active high. 1b: Channel 3 output polarity is active low.	0x0
12 POL4	RW	Channel 4 output polarity This bit defines the channel 4 active state in output mode. This field is write protected. It can only be written when the write protection is disabled. 0b: Channel 4 output polarity is active high. 1b: Channel 4 output polarity is active low.	0x0
13 POL5	RW	Channel 5 output polarity This bit defines the channel 5 active state in output mode. This field is write protected. It can only be written when the write protection is disabled. 0b: Channel 5 output polarity is active high. 1b: Channel 5 output polarity is active low.	0x0
14 POL6	RW	Channel 6 output polarity This bit defines the channel 6 active state in output mode. This field is write protected. It can only be written when the write protection is disabled. 0b: Channel 6 output polarity is active high.	0x0

Field	Access	Description	Reset Value
		1b: Channel 6 output polarity is active low.	
15 POL7	RW	Channel 7 output polarity This bit defines the channel 7 active state in output mode. This field is write protected. It can only be written when the write protection is disabled. 0b: Channel 7 output polarity is active high. 1b: Channel 7 output polarity is active low.	0x0
16 TRIGE0	RW	Channel 0 match trigger enable Enable the generation of match trigger when the counter reaches the CCV0 value. 0b: Disable channel 0 match trigger generation. 1b: Enable channel 0 match trigger generation.	0x0
17 TRIGE1	RW	Channel 1 match trigger enable Enable the generation of match trigger when the counter reaches the CCV1 value. 0b: Disable channel 1 match trigger generation. 1b: Enable channel 1 match trigger generation.	0x0
18 TRIGE2	RW	Channel 2 match trigger enable Enable the generation of match trigger when the counter reaches the CCV2 value. 0b: Disable channel 2 match trigger generation. 1b: Enable channel 2 match trigger generation.	0x0
19 TRIGE3	RW	Channel 3 match trigger enable Enable the generation of match trigger when the counter reaches the CCV3 value. 0b: Disable channel 3 match trigger generation. 1b: Enable channel 3 match trigger generation.	0x0
20 TRIGE4	RW	Channel 4 match trigger enable Enable the generation of match trigger when the counter reaches the CCV4 value. 0b: Disable channel 4 match trigger generation. 1b: Enable channel 4 match trigger generation.	0x0
21 TRIGE5	RW	Channel 5 match trigger enable Enable the generation of match trigger when the counter reaches the CCV5 value. 0b: Disable channel 5 match trigger generation. 1b: Enable channel 5 match trigger generation.	0x0
22 TRIGE6	RW	Channel 6 match trigger enable Enable the generation of match trigger when the counter reaches the CCV6 value. 0b: Disable channel 6 match trigger generation. 1b: Enable channel 6 match trigger generation.	0x0
23 TRIGE7	RW	Channel 7 match trigger enable Enable the generation of match trigger when the counter reaches the CCV7 value. 0b: Disable channel 7 match trigger generation. 1b: Enable channel 7 match trigger generation.	0x0
24 INITRIGE	RW	Initialization trigger enable Enable the generation of initial trigger when the counter reaches CNTINIT value. 0b: Disable initialization trigger generation. 1b: Enable initialization trigger generation. Note: Initialization trigger frequency is equal to reload frequency which is configured in LDFREQ field.	0x0
25 TRIGF	R/W0C	Channel output trigger flag When any channel match trig is generated, this flag is set	0x0

Field	Access	Description	Reset Value
		by hardware. TRIGF is cleared when writing 0 to TRIGF after reading TRIGF while TRIGF is 1. 0b: No channel match trigger is generated. 1b: At least one channel match trigger is generated.	
31:26 RSVD_31_26	RO	Reserved	0x0

30.3.11 TIM Channel Output Software Control Register (TIM_OUTSWCR)

30.3.11.1 Offset

Register	Offset
TIM_OUTSWCR	0x5C

30.3.11.2 Function

This is the software control register.

30.3.11.3 Fields

Field	Access	Description	Reset Value
0 CH0OC	RW	Channel 0 output software control enable 0b: Disable channel 0 output software control. 1b: Enable channel 0 output software control.	0x0
1 CH1OC	RW	Channel 1 output software control enable 0b: Disable channel 1 output software control. 1b: Enable channel 1 output software control.	0x0
2 CH2OC	RW	Channel 2 output software control enable 0b: Disable channel 2 output software control. 1b: Enable channel 2 output software control.	0x0
3 CH3OC	RW	Channel 3 output software control enable 0b: Disable channel 3 output software control. 1b: Enable channel 3 output software control.	0x0
4 CH4OC	RW	Channel 4 output software control enable 0b: Disable channel 4 output software control. 1b: Enable channel 4 output software control.	0x0
5 CH5OC	RW	Channel 5 output software control enable 0b: Disable channel 5 output software control. 1b: Enable channel 5 output software control.	0x0
6 CH6OC	RW	Channel 6 output software control enable 0b: Disable channel 6 output software control. 1b: Enable channel 6 output software control.	0x0
7 CH7OC	RW	Channel 7 output software control enable 0b: Disable channel 7 output software control. 1b: Enable channel 7 output software control.	0x0
8 CH0OCV	RW	Channel 0 output software control value 0b: Channel 0 output is forced to inactive state by the software control. 1b: Channel 0 output is forced to active state by the software control. Note: in independent mode, channel0's inactive/active state is decided by POL0, for example, if POL0 is active	0x0

Field	Access	Description	Reset Value
		high and CH0OCV is active state, Channel 0 output is high level. In combined mode, channel0's and channel1's inactive/active state is decided by Table 30-9.	
9 CH1OCV	RW	Channel 1 output software control value 0b: Channel 1 output is forced to inactive state by the software control. 1b: Channel 1 output is forced to active state by the software control. Note: in independent mode, channel1's inactive/active state is decided by POL1, for example, if POL1 is active high and CH1OCV is active state, Channel 1 output is high level. In combined mode, channel0's and channel1's inactive/active state is decided by Table 30-9.	0x0
10 CH2OCV	RW	Channel 2 output software control value 0b: Channel 2 output is forced to inactive state by the software control. 1b: Channel 2 output is forced to active state by the software control. Note: in independent mode, channel2's inactive/active state is decided by POL2, for example, if POL2 is active high and CH2OCV is active state, Channel 2 output is high level. In combined mode, channel2's and channel3's inactive/active state is decided by Table 30-9.	0x0
11 CH3OCV	RW	Channel 3 output software control value 0b: Channel 3 output is forced to inactive state by the software control. 1b: Channel 3 output is forced to active state by the software control. Note: in independent mode, channel3's inactive/active state is decided by POL3, for example, if POL3 is active high and CH3OCV is active state, Channel 3 output is high level. In combined mode, channel2's and channel 3's inactive/active state is decided by Table 30-9.	0x0
12 CH4OCV	RW	Channel 4 output software control value 0b: Channel 4 output is forced to inactive state by the software control. 1b: Channel 4 output is forced to active state by the software control. Note: in independent mode, channel4's inactive/active state is decided by POL4, for example, if POL4 is active high and CH4OCV is active state, Channel 4 output is high level. In combined mode, channel4's and channel 5's inactive/active state is decided by Table 30-9.	0x0
13 CH5OCV	RW	Channel 5 output software control value 0b: Channel 5 output is forced to inactive state by the software control. 1b: Channel 5 output is forced to active state by the software control. Note: in independent mode, channel5's inactive/active state is decided by POL5, for example, if POL5 is active high and CH5OCV is active state, Channel 5 output is high level. In combined mode, channel4's and channel 5's inactive/active state is decided by Table 30-9.	0x0
14 CH6OCV	RW	Channel 6 output software control value 0b: Channel 6 output is forced to inactive state by the software control.	0x0

Field	Access	Description	Reset Value
		1b: Channel 6 output is forced to active state by the software control. Note: in independent mode, channel6's inactive/active state is decided by POL6, for example, if POL6 is active high and CH6OCV is active state, Channel 6 output is high level. In combined mode, channel6's and channel7's inactive/active state is decided by Table 30-9.	
15 CH7OCV	RW	Channel 7 output software control value 0b: Channel 7 output is forced to inactive state by the software control. 1b: Channel 7 output is forced to active state by the software control. Note: in independent mode, channel7's inactive/active state is decided by POL7, for example, if POL7 is active high and CH7OCV is active state, Channel 7 output is high level. In combined mode, channel6's and channel7's inactive/active state is decided by Table 30-9.	0x0
31:16 RSVD_31_16	RO	Reserved	0x0

30.3.12 TIM Pairs of Channel Control Register (TIM_PCR)

30.3.12.1 Offset

Register	Offset
TIM_PCR	0x60

30.3.12.2 Function

This is the pair of channels control register.

30.3.12.3 Fields

Field	Access	Description	Reset Value
0 FCTLEN0	RW	Fault control enabled for pairs of channel 0 Enable the fault control in the channels (n) and (n+1), where n=0. This field is write protected. It can be written only when the write protection is disabled. 0b: The fault control in this pair of channels is disabled. 1b: The fault control in this pair of channels is enabled.	0x0
1 DTEN0	RW	Deadtime insertion enable for pairs of channel 0 Enable the deadtime insertion in the channels (n) and (n+1), where n=0. This field is write protected. It can be written only when the write protection is disabled. 0b: The deadtime insertion in this pair of channels is disabled. 1b: The deadtime insertion in this pair of channels is enabled.	0x0
2 COMB0	RW	Combine enable for pairs of channel 0 Enable the combination of channel(n) and channel(n+1), where n=0. When this pair of channels is combined, their channel outputs are complementary. This field is write protected. It can be written only when write protection is disabled.	0x0

Field	Access	Description	Reset Value
		0b: The channel(n) and channel(n+1) are independent. 1b: The channel(n) and channel(n+1) are combined, channel(n) and channel(n+1) outputs are complementary.	
3 SYNCEN0	RW	Synchronization enable for pairs of channel 0 Enable the synchronization of CCV(n) and CCV(n+1) register value, where n=0. 0b: Synchronization in this pair of channels is disabled. 1b: Synchronization in this pair of channels is enabled.	0x0
4 DECAPEN0	RW	Dual edge capture enable for pairs of channel 0 Enable the dual-edge capture mode in the channels (n) and (n+1), where n=0. This field is write protected. It can be written only when write protection is disabled. 0b: The dual-edge capture mode in this pair of channels is disabled. 1b: The dual-edge capture mode in this pair of channels is enabled.	0x0
5 DECAPO	RW	Dual edge capture mode for pairs of channel 0 Activate the dual-edge capture of the channel(n) input event, where n=0. This field only makes sense when DECAPEN0 was enabled. DECAP bit is cleared automatically by hardware if dual-edge one pulse capture mode is selected and this input pulse is captured. 0b: The dual-edge captures are inactive. 1b: The dual-edge captures are active.	0x0
7:6 RSVD_7_6	RO	Reserved	0x0
8 FCTLEN1	RW	Fault control enabled for pairs of channel 1 Enable the fault control in the channels (n) and (n+1), where n=2. This field is write protected. It can be written only when the write protection is disabled. 0b: The fault control in this pair of channels is disabled. 1b: The fault control in this pair of channels is enabled.	0x0
9 DTEN1	RW	Deadtime insertion enable for pairs of channel 1 Enable the deadtime insertion in the channels (n) and (n+1), where n=2. This field is write protected. It can be written only when the write protection is disabled. 0b: The deadtime insertion in this pair of channels is disabled. 1b: The deadtime insertion in this pair of channels is enabled.	0x0
10 COMB1	RW	Combine enable for pairs of channel 1 Enable the combination of channel(n) and channel(n+1), where n=2. When this pair of channels is combined, outputs of their channels are complementary. This field is write protected. It can be written only when the write protection is disabled. 0b: The channel(n) and channel(n+1) are independent. 1b: The channel(n) and channel(n+1) are combined, channel(n) and channel(n+1) outputs are complementary.	0x0
11 SYNCEN1	RW	Synchronization enable for pairs of channel 1 Enable the synchronization of CCV(n) and CCV(n+1) register value, where n=2. 0b: Synchronization in this pair of channels is disabled.	0x0

Field	Access	Description	Reset Value
		1b: Synchronization in this pair of channels is enabled.	
12 DECAPEN1	RW	Dual edge capture enable for pairs of channel 1 Enable the dual-edge capture mode in the channels (n) and (n+1), where n=2. This field is write protected. It can be written only when write protection is disabled. 0b: The dual-edge capture mode in this pair of channels is disabled. 1b: The dual-edge capture mode in this pair of channels is enabled.	0x0
13 DECAP1	RW	Dual edge capture mode for pairs of channel 1 Activate the dual-edge capture of the channel (n) input event, where n=2. This field only makes sense when DECAPEN1 was enabled. DECAP bit is cleared automatically by hardware if the dual-edge one pulse capture mode is selected and this input pulse is captured. 0b: The dual-edge captures are inactive. 1b: The dual-edge captures are active.	0x0
15:14 RSVD_15_14	RO	Reserved	0x0
16 FCTLEN2	RW	Fault control enabled for pairs of channel 2 Enable the fault control in the channels (n) and (n+1), where n=4. This field is write protected. It can be written only when the write protection is disabled. 0b: The fault control in this pair of channels is disabled. 1b: The fault control in this pair of channels is enabled.	0x0
17 DTEN2	RW	Deadtime insertion enable for pairs of channel 2 Enable the deadtime insertion in the channels (n) and (n+1), where n=4. This field is write protected. It can be written only when write protection is disabled. 0b: The deadtime insertion in this pair of channels is disabled. 1b: The deadtime insertion in this pair of channels is enabled.	0x0
18 COMB2	RW	Combine enable for pairs of channel 2 Enable the combination of channel(n) and channel(n+1), where n=4. When this pair of channels is combined, their channels outputs are complementary. This field is write protected. It can be written only when write protection is disabled. 0b: The channel(n) and channel(n+1) are independent. 1b: The channel(n) and channel(n+1) are combined, channel(n) and channel(n+1) outputs are complementary.	0x0
19 SYNCEN2	RW	Synchronization enable for pairs of channel 2 Enable the synchronization of CCV(n) and CCV(n+1) register value, where n=4. 0b: Synchronization in this pair of channels is disabled. 1b: Synchronization in this pair of channels is enabled.	0x0
20 DECAPEN2	RW	Dual edge capture enable for pairs of channel 2 Enable the dual-edge capture mode in the channels (n) and (n+1), where n=4. This field is write protected. It can be written only when write protection is disabled. 0b: The dual-edge capture mode in this pair of channels is disabled. 1b: The dual-edge capture mode in this pair of channels	0x0

Field	Access	Description	Reset Value
		is enabled.	
21 DECAP2	RW	Dual edge capture mode for pairs of channel 2 Activate the dual-edge capture of the channel (n) input event, where n=4. This field only makes sense when DECAPEN2 was enabled. DECAP bit is cleared automatically by hardware if the dual-edge one pulse capture mode is selected and this input pulse is captured. 0b: The dual-edge captures are inactive. 1b: The dual-edge captures are active.	0x0
23:22 RSVD_23_22	RO	Reserved	0x0
24 FCTLEN3	RW	Fault control enabled for pairs of channel 3 Enable the fault control in the channels (n) and (n+1), where n=6. This field is write protected. It can be written only when write protection is disabled. 0b: The fault control in this pair of channels is disabled. 1b: The fault control in this pair of channels is enabled	0x0
25 DTEN3	RW	Deadtime insertion enable for pairs of channel 3 Enable the deadtime insertion in the channels (n) and (n+1), where n=6. This field is write protected. It can be written only when write protection is disabled. 0b: The deadtime insertion in this pair of channels is disabled. 1b: The deadtime insertion in this pair of channels is enabled.	0x0
26 COMB3	RW	Combine enable for pairs of channel 3 Enable the combination of channel(n) and channel(n+1), where n=6. When this pair of channels is combined, their channels output is complementary. This field is write protected. It can be written only when the write protection is disabled. 0b: The channel(n) and channel(n+1) are independent. 1b: The channel(n) and channel(n+1) are combined. Channel(n) and channel(n+1) outputs are complementary.	0x0
27 SYNCEN3	RW	Synchronization enable for pairs of channel 3 Enable the synchronization of CCV(n) and CCV(n+1) register value, where n=6. 0b: Synchronization in this pair of channels is disabled. 1b: Synchronization in this pair of channels is enabled.	0x0
28 DECAPEN3	RW	Dual edge capture enable for pairs of channel 3 Enable the dual-edge capture mode in the channels (n) and (n+1), where n=6. This field is write protected. It can be written only when write protection is disabled. 0b: The dual-edge capture mode in this pair of channels is disabled. 1b: The dual-edge capture mode in this pair of channels is enabled.	0x0
29 DECAP3	RW	Dual edge capture mode for pairs of channel 3 Activate the dual edge capture of the channel (n) input event, where n=6. This field only makes sense when DECAPEN3 was enabled. DECAP bit is cleared automatically by hardware if the dual-edge one pulse capture mode is selected and this input pulse is captured.	0x0

Field	Access	Description	Reset Value
		0b: The dual-edge captures are inactive. 1b: The dual-edge captures are active.	
31:30 RSVD_31_30	RO	Reserved	0x0

30.3.13 TIM Input Filter Register (TIM_FILTER)

30.3.13.1 Offset

Register	Offset
TIM_FILTER	0x64

30.3.13.2 Function

This register is about the input filter configuration.

30.3.13.3 Fields

Field	Access	Description	Reset Value
3:0 FILVAL0	RW	Channel 0 input filter value Configure the channel 0 input filter value when channel 0 works as an input channel. This channel input filter is disabled when this field value is zero.	0x0
7:4 FILVAL1	RW	Channel 1 input filter value Configure the channel 1 input filter value when channel 1 works as an input channel. This channel input filter is disabled when this field value is zero.	0x0
11:8 FILVAL2	RW	Channel 2 input filter value Configure the channel 2 input filter value when channel 2 works as an input channel. This channel input filter is disabled when this field value is zero.	0x0
15:12 FILVAL3	RW	Channel 3 input filter value Configure the channel 3 input filter value when channel 3 works as an input channel. This channel input filter is disabled when this field value is zero.	0x0
17:16 DTPS	RW	Deadtime prescaler value Select the divide value of system clock. This divided clock is used for deadtime counter. This field is write protected. It can only be written when write protection is disabled. 0xb: Divided the system clock by 1. 10b: Divided the system clock by 4. 11b: Divided the system clock by 16.	0x0
23:18 DTVAL	RW	Deadtime insertion value Configure the deadtime insertion value for the deadtime counter. This field is write protected. It can only be written when write protection is disabled. The deadtime insertion value equals (DTPS*DTVAL). DTVAL indicates the number of deadtime counts inserted. It ranges from 0 to 63.	0x0

Field	Access	Description	Reset Value
31:24 RSVD_31_24	RO	Reserved	0x0

30.3.14 TIM Fault Control Register (TIM_FLTCR)

30.3.14.1 Offset

Register	Offset
TIM_FLTCR	0x68

30.3.14.2 Function

This register is about the fault control configuration.

30.3.14.3 Fields

Field	Access	Description	Reset Value
0 FAULTEN0	RW	Fault input enable 0 Enable the fault input channel 0. This field is write protected. It can only be written when the write protection is disabled. 0b: The fault input channel 0 is disabled. 1b: The fault input channel 0 is enabled.	0x0
1 FIFEN0	RW	Fault input filter enable 0 Enable the filter for fault input channel 0. This field is write protected. It can only be written when the write protection is disabled. 0b: The filter for fault input channel 0 is disabled. 1b: The filter for fault input channel 0 is enabled.	0x0
2 FLTPOLO	RW	Fault input channel 0 polarity This field is write protected. It can only be written when the write protection is disabled. 0b: The fault input channel 0 polarity is active high, a logic one at fault input channel indicates a fault event. 1b: The fault input channel 0 polarity is active low, a logic zero at fault input channel indicates a fault event.	0x0
3 FAULTEN1	RW	Fault input enable 1 Enable the fault input channel 1. This field is write protected. It can only be written when the write protection is disabled. 0b: The fault input channel 1 is disabled. 1b: The fault input channel 1 is enabled.	0x0
4 FIFEN1	RW	Fault input filter enable 1 Enable the filter for fault input channel 1. This field is write protected. It can only be written when the write protection is disabled. 0b: The filter for fault input channel 1 is disabled. 1b: The filter for fault input channel 1 is enabled.	0x0
5 FLTPOL1	RW	Fault input channel 1 polarity This field is write protected. It can only be written when the write protection is disabled. 0b: The fault input channel 1 polarity is active high, a logic one at fault input channel indicates a fault event. 1b: The fault input channel 1 polarity is active low. A logic	0x0

Field	Access	Description	Reset Value
		zero at fault input channel indicates a fault event.	
6 FLTIE	RW	Fault interrupt enable Enable the interrupts generation when a fault is detected, and the fault control is enabled. 0b: A fault detection interrupt was disabled. 1b: A fault detection interrupt was enabled.	0x0
7 FLTCEN	RW	Fault control enable This field is write protected. It can only be written when the write protection is disabled. 0b: A fault control is disabled. 1b: A fault control is enabled.	0x0
8 FLTCM	RW	Fault clear mode This bit defines when fault input is deasserted, the fault condition is manually cleared or automatically cleared. When the fault flag is set and fault input is deasserted, for automatically clear mode, the fault condition is automatically cleared and channel output is not disabled after fault input is deasserted; for manually clear mode, the user must clear the fault flag by software, otherwise the fault condition is not cleared and channel output is still disabled by previous fault condition. This field is write protected, it can only be written when write protection is disabled. 0b: Manually fault clear mode. 1b: Automatically fault clear mode.	0x0
10:9 RSVD_10_9	RO	Reserved	0x0
14:11 FIFVAL	RW	Fault input filter value Configure the filter value for the fault inputs. The fault input filter is disabled when this field value is zero. Refer to the section fault input control for more details.	0x0
15 FSTATE	RW	Fault output status This bit allows TIM output to be tri-stated when a fault event is detected. This field is write protected. It can only be written when the write protection is disabled. 0b: When the fault event is on-going, TIM output will be inactive state. 1b: When the fault event is on-going, TIM output will be tri-stated.	0x0
31:16 RSVD_31_16	RO	Reserved	0x0

30.3.15 TIM Fault Status Register (TIM_FLTSR)

30.3.15.1 Offset

Register	Offset
TIM_FLTSR	0x6C

30.3.15.2 Function

This register is about the fault input status.

30.3.15.3 Fields

Field	Access	Description	Reset Value
0 FAULTF0	W0C	Fault detection flag of fault channel 0 When the corresponding fault input is enabled and a fault condition is detected at the fault input, the corresponding fault flag was set. FAULTF0 was cleared by writing a 0 to FAULTF0 while there is no existing fault condition at the corresponding fault inputs. Writing a 1 to FAULTF0 has no effect. 0b: No fault event was detected at fault input channel 0. 1b: A fault event was detected at fault input channel 0.	0x0
1 FAULTF1	W0C	Fault detection flag of fault channel 1 When the corresponding fault input is enabled and a fault condition is detected at the fault input, the corresponding fault flag was set. FAULTF1 was cleared by writing a 0 to FAULTF1 while there is no existing fault condition at the corresponding fault inputs. Writing a 1 to FAULTF1 has no effect. 0b: No fault event was detected at fault input channel 1. 1b: A fault event was detected at fault input channel 1.	0x0
3:2 RSVD_3_2	RO	Reserved	0x0
4 FAULTF	W0C	Fault detection flag Represents the logic OR of the fault input channels fault detection flag. FAULTF was cleared by writing a 0 to FAULTF while there is no existing fault condition at the enabled fault inputs. Writing a 1 to FAULTF has no effect. 0b: No fault event was detected. 1b: At least one fault event was detected.	0x0
5 FAULTIN	RO	Fault inputs value Represents the logic OR of the enabled fault inputs after their filter (if their filter is enabled) when the fault control is enabled. 0b: The logic OR of the enabled fault inputs is 0. 1b: The logic OR of the enabled fault inputs is 1.	0x0
6 WPEN	RW	Write protection enable The WPEN bit is the negation of the WPDIS bit. WPEN is set when 1 is written to it. WPEN is cleared when WPEN bit is read as a 1 and then 1 is written to WPDIS. Writing 0 to WPEN has no effect. 0b: The write protection is disabled. Write protected bits can be written. 1b: The write protection is enabled. Write protected bits cannot be written.	0x0
31:7 RSVD_31_7	RO	Reserved	0x0

30.3.16 TIM Synchronization Register (TIM_SYNC)

30.3.16.1 Offset

Register	Offset
TIM_SYNC	0x70

30.3.16.2 Function

This register is about synchronization.

30.3.16.3 Fields

Field	Access	Description	Reset Value
0 SWTRIG	RW	PWM synchronization software trigger Selects the software trigger as the PWM synchronization trigger. The software trigger happens when a 1 is written to SWTRIG bit. 0b: A software trigger is not selected. 1b: A software trigger is selected. NOTE SWTRIG bit is cleared by two ways: a) Software write 0 to SWTRIG bit. b) Set SWRSTCNT bit when the software trigger event occurs.	0x0
7:1 RSVD_7_1	RO	Reserved	0x0
8 SYNCINIT	RW	CNTINIT register synchronization Selects when the CNTINIT register value is updated with its buffer value. 0b: The CNTINIT register is updated with its buffer value on all rising edges of the system clock. 1b: The CNTINIT register is updated with its buffer value only by the PWM synchronization.	0x0
9 SYNCOSWC	RW	OUTSWCR register synchronization Selects when the OUTSWCR register value is updated with its buffer value. 0b: The OUTSWCR register is updated with its buffer value on all rising edges of the system clock. 1b: The OUTSWCR register is updated with its buffer values only by PWM synchronization.	0x0
17:10 RSVD_17_10	RO	Reserved	0x0
18 SWWRBUF	RW	Software trigger activate write buffer synchronization Software trigger activates MOD, CNTINIT, CCVn, OUTSWCR and HCV registers synchronization with their written buffer value. 0b: The software trigger does not activate MOD, CNTINIT, CCVn, OUTSWCR and HCV registers synchronization with their written buffer value. 1b: The software trigger activates MOD, CNTINIT, CCVn, OUTSWCR and HCV registers synchronization with their written buffer value.	0x0
19 SWRSTCNT	RW	Software trigger activate CNT register synchronization NOTE When the SWRSTCNT bit is set, TIM_SYNC[SWTRIG] bit will be automatically cleared after writing 1 to SWTRIG. 0b: The software trigger does not activate the CNT register synchronization. 1b: The software trigger activates the CNT register synchronization.	0x0
31:20 RSVD_31_20	RO	Reserved	0x0

30.3.17 TIM Reload Register (TIM_RELOAD)

30.3.17.1 Offset

Register	Offset
TIM_RELOAD	0x74

30.3.17.2 Function

This register controls the reload mechanisms, and enables the reload of the MOD, CNTINIT, CCV, HCV registers with their write buffer value when a reload event happens. A full cycle reload event occurs when the counter reaches the MOD value or matches the CCV(n) value. A half-cycle reload event occurs when the counter reaches the HCV value.

30.3.17.3 Fields

Field	Access	Description	Reset Value
0 CH0SEL	RW	Channel 0 select When the counter matches the CCV0 value, and this bit is enabled, a reload event occurs. 0b: Channel 0 match is not an opportunity of reloading. 1b: Channel 0 match is an opportunity of reloading.	0x0
1 CH1SEL	RW	Channel 1 select When the counter matches the CCV1 value, and this bit is enabled, a reload event occurs. 0b: Channel 1 match is not an opportunity of reloading. 1b: Channel 1 match is an opportunity of reloading.	0x0
2 CH2SEL	RW	Channel 2 select When the counter matches the CCV2 value, and this bit is enabled, a reload event occurs. 0b: Channel 2 match is not an opportunity of reloading. 1b: Channel 2 match is an opportunity of reloading.	0x0
3 CH3SEL	RW	Channel 3 select When the counter matches the CCV3 value, and this bit is enabled, a reload event occurs. 0b: Channel 3 match is not an opportunity of reloading. 1b: Channel 3 match is an opportunity of reloading.	0x0
4 CH4SEL	RW	Channel 4 select When the counter matches the CCV4 value, and this bit is enabled, a reload event occurs. 0b: Channel 4 match is not an opportunity of reloading. 1b: Channel 4 match is an opportunity of reloading.	0x0
5 CH5SEL	RW	Channel 5 select When the counter matches the CCV5 value, and this bit is enabled, a reload event occurs. 0b: Channel 5 match is not an opportunity of reloading. 1b: Channel 5 match is an opportunity of reloading.	0x0
6 CH6SEL	RW	Channel 6 select When the counter matches the CCV6 value, and this bit is enabled, a reload event occurs. 0b: Channel 6 match is not an opportunity of reloading. 1b: Channel 6 match is an opportunity of reloading.	0x0
7 CH7SEL	RW	Channel 7 select When the counter matches the CCV7 value, and this bit is enabled, a reload event occurs.	0x0

Field	Access	Description	Reset Value
		0b: Channel 7 match is not an opportunity of reloading. 1b: Channel 7 match is an opportunity of reloading.	
8 LOADEN	RW	Reload enable Enables the loading of MOD, CNTINIT and CCV register with their write buffer value when a reload event happens. This bit is automatically cleared after the reload is completed. 0b: Disable MOD, CNTINIT, and CCV register loads the write buffer value. 1b: Enable MOD, CNTINIT, and CCV register loads the write buffer value.	0x0
10:9 RSVD_10_9	RO	Reserved	0x0
11 HCSEL	RW	Half cycle reload select HCSEL defines whether this half-cycle match is regarded as a reload opportunity when the TIM counter matches the HCV value. 0b: Half cycle match is not a reload opportunity. 1b: Half cycle match is a reload opportunity.	0x0
15:12 RSVD_15_12	RO	Reserved	0x0
20:16 LDFREQ	RW	Reload frequency This field decides the PWM reload frequency. LDFREQ = n indicates RLDF bit is set every (n+1) reload opportunities. n ranges from 0 to 31.	0x0
31:21 RSVD_31_21	RO	Reserved	0x0

30.3.18 TIM Half Cycle Reload Value Register (TIM_HCV)

30.3.18.1 Offset

Register	Offset
TIM_HCV	0x78

30.3.18.2 Function

This register stores half cycle value.

30.3.18.3 Fields

Field	Access	Description	Reset Value
15:0 HCV	RW	Half cycle reload value This field contains the match value for the TIM half cycle reload feature. After the TIM counter reaches this value, a reload opportunity is generated if TIM_RELOAD[HCSEL] is set to 1. Writing to the HCV register latches the value into a buffer.	0x0
31:16 RSVD_31_16	RO	Reserved	0x0

30.4 Function description

This section provides the functional information of the TIM module.

NOTE: system clock represents core clock in this chapter.

30.4.1 Time base unit

The main block of TIM is a 16-bit free-running counter. The time base unit provides this counter clock. The time base unit selects one clock source which is divided by prescaler. The time base unit also determines the counter behavior in the debug mode.

30.4.1.1 Clock source selection

You can select the clock source for the counter clock from the following:

- Core clock
- Function clock
- External clock from pad

To select the counter clock source, configure the TIMEBASE[CKSRC] register. The CKSRC field can be written or read at any time.

- CKSRC=2'b00, no clock source is selected so counter is disabled. Writing 2'b00 to CKSRC will disable counter but not affect counter register value.
- CKSRC=2'b01, core clock source is selected.
- CKSRC=2'b10, function clock source is selected.
- CKSRC=2'b11, external clock source is selected.

The function clock source and external clock from the pad are asynchronous clock source. So it must pass through a synchronizer clocked by the core clock to assure that the counter transitions are properly aligned to the core clock. Therefore, to meet Nyquist criteria and consider the jitter, when the function clock or external clock is selected as counter clock, the frequency of clock source must not exceed 1/4 of the core clock frequency.

NOTE

In this SoC chip, there are three external clock sources from pads: TCLK0, TCLK1, and TCLK2. Any external clock source can be selected as TIM external clock source by configuring SCM_MISCCTL2 register.

30.4.1.2 Prescaler description

The prescaler implements a 3-bit prescaler counter to divide the selected counter clock by factor 1/2/4/8/16/32/64/128. This divided factor is selected by the bit field PSDIV[4:2].

The following figure gives an example of the counter and prescaler counter behavior.

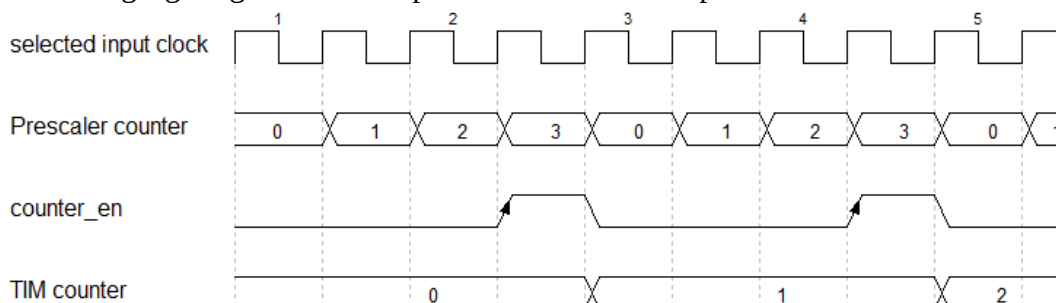


Figure 30-2 TIM counter prescaler with PSDIV=3'b010

30.4.1.3 Debug mode

When the chip enters into the debug mode, the counter behavior is determined by the DBGCM

fields based on the following table.

Table 30-3 TIM behavior in debug mode

DBGM	counter	CHF	Channel output
00b	stopped	can be set	Function mode
01b	stopped	is not set	Channels outputs are forced to safe value
10b	stopped	is not set	Channels outputs are frozen
11b	running	can be set	Function mode

30.4.2 Counter mode

The TIM 16-bit free-running counter can count with two modes: up counting mode and up-down counting mode. The counting mode is selected by GLBCR[CNTMODE] bit.

- CNTMODE = 0b, up counting mode
- CNTMODE = 1b, up-down counting mode

30.4.2.1 Up counting mode

In the up counting mode, the counter increments from CNTINIT value to MOD value, then restarts from CNTINIT and generates counter overflow event. When the counter is free-running, updating CNTINIT or MOD registers will load the updated value to write buffer. See the section of registers synchronization for details about register update.

In the up counting mode, the TIM period is $(MOD - CNTINIT + 1)$ multiplied by counter clock period. The following figure is the counter behavior in the up counting mode.

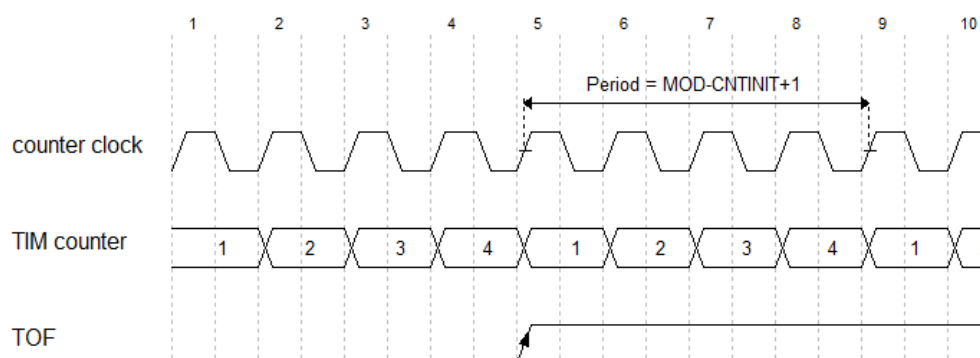


Figure 30-3 Counter behavior in up counting mode (MOD=0x4, CNTINIT=0x1)

30.4.2.2 Up-down counting mode

In the up-down counting mode, the counter increments from CNTINIT value to MOD value, then decrements until CNTINIT value is reached to restart counting. Counter overflow event is generated when counter decrements from MOD to (MOD-1). When the counter is free-running, updating CNTINIT or MOD registers will load the updated value to write buffer. See the section of registers synchronization for details about register update.

In an up-down counting mode, the TIM period is $2 * (MOD - CNTINIT)$ multiplied by counter clock period. The following figure shows the counter behavior in the up-down counting mode.

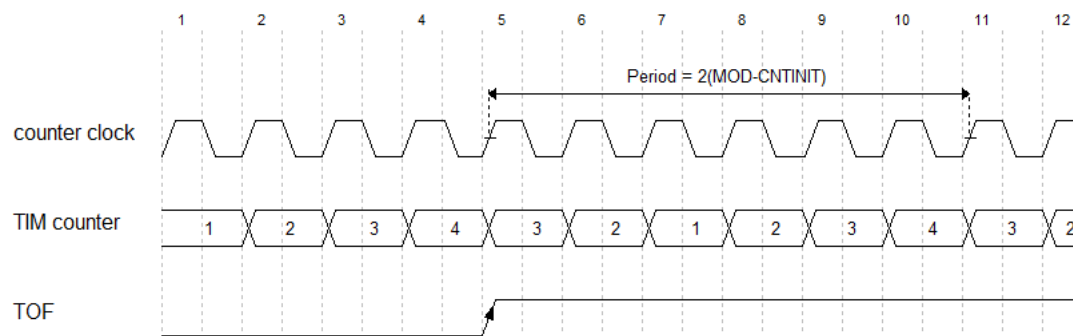


Figure 30-4 Counter behavior in up-down counting mode (MOD=0x4, CNTINIT=0x1)

30.4.3 Channels mode

30.4.3.1 Channels work mode

The TIM channels can work as input channels or output channels.

- If channels work as input channels, the following two channel modes can be selected:
 - Input capture mode
 - Dual-edge capture mode
- If channels work as output channels, the following four channel modes can be selected:
 - Output compare mode
 - Edge-aligned PWM output mode
 - Center-aligned PWM output mode
 - Complementary PWM output mode

The following table shows the selection of different channel modes.

Table 30-4 Channel mode selection (DECAPEN=0)

COMB	CNTMODE	CMS[1:0]	ELS[1:0]	Channel Mode	Function
0	0	00	00	Input capture	Channel input capture is disabled
			01		Capture only on rising edge
			10		Capture only on falling edge
			11		Capture both on rising and falling edges
		01	00	Output compare	Channel output does not change on match
			01		Toggle output on match
			10		Force output to inactive state on match
			11		Force output to active state on match
	1	1x	x1	Edge-aligned PWM	Low true pulse (force output to active state on match)
			10		High true pulse (force output to inactive state on match)
		xx	x1	Center-aligned PWM	Low true pulse (force output to active state on match)
			10		High true pulse (force output to inactive state on match)

COMB	CNTMODE	CMS[1:0]	ELS[1:0]	Channel Mode	Function
1	0	xx	x1	Complementary PWM	Low true pulse (force channel(n) to inactive state on match CCV(n) and force channel(n) to active state on match CCV(n+1))
			10		High true pulse (force channel(n) to active state on match CCV(n) and force channel(n) to inactive state on match CCV(n+1))

30.4.3.2 Channels output initialization

When the channel works as an output channel, channel output is initialized by writing 1'b1 to GLBCR[INIT] bit. Then channel(n) initial value INIT(n) is forced to channel(n) output.

- When channel(n) and channel(n+1) work in independent mode, deadtime insertion is not effective, that is, COMB=0 and DTEN=0, initialization directly forces INIT(n) value to channel(n) output and forces INIT(n+1) to channel(n+1) output, and channel(n) and channel(n+1) outputs are independent.
- When channel(n) and channel(n+1) work in complementary mode, that is, COMB is set to 1, or deadtime insertion is enabled, that is, DTEN is set to 1, initialization directly forces INIT(n) value to channel(n) output, but channel(n+1) value is complementary with channel(n) value. See details in the following table.

Table 30-5 Channel initialization when COMB =1 or DTEN =1

INIT(n)	INIT(n+1)	Channel(n) output	Channel(n+1) output
0	x	inactive state	active state
1	x	active state	inactive state

NOTE

In output mode, if TIM_TIMEBASE[CKSRC] is set to 0 from non-zero, TIM channel's internal level keeps its state, and if TIM_GLBCR[CHOEn] (n=0, ..., 7) is also set to 0, channel's internal level will not be output to the pin. So during initialization or re-initialization, TIM_GLBCR[CHOEn] is set again, channel's internal previous level will be output to the pin immediately. To make sure that TIM initial channel output is an expected level, it is recommended to reconfigure TIM_OUTCR[INITn] (n=0, ..., 7) and write 1 to TIM_GLBCR [INIT] to take effect.

30.4.3.3 Channels output polarity

The POL(n) bit controls the channel(n) output polarity:

- POL = 0, the channel output polarity is high, so logic one is an active state and the logic zero is an inactive state.
- POL = 1, the channel output polarity is low, so logic one is an inactive state and logic zero is an active state.

In the description of the following chapter, if there is no special clarification, by default, the channel polarity is set to high, so logic one is channel active state, and logic zero is channel inactive state. On the other hand, if channel polarity is set as low, then the channel output behavior is reverse, where logic one is channel inactive state and logic zero is channel active state.

30.4.4 Input capture mode

The input capture mode is selected with following configuration.

- DECAPEN = 0
- COMB = 0
- CNTMODE = 0
- CMS = 00
- ELS $\neq$ 00

When a selected edge occurs on this channel input pin:

1. The current value of TIM counter is captured into the channel n CCV register.
2. Channel n flag CHF is set.
3. Channel n interrupt is generated if channel n CHIE is enabled and DMA is disabled.
4. Channel n DMA request is generated if channel n CHIE and DMA are enabled.

In the following input capture figure, the CHn input signal will pass through two flip-flops to be synchronized by system clock to meet Nyquist criteria for signal sampling. Also, considering the jitter, the frequency of the channel input signal cannot exceed 1/4 of the system clock.

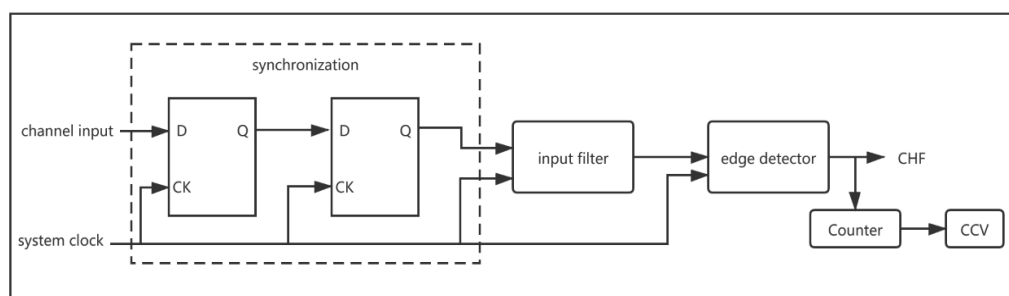


Figure 30-5 Input capture diagram

Considering the input synchronization and edge detect logic, if the input filter is not enabled, that is, FIFVALn = 0, when a valid channel sensitive edge comes, the channel flag will set after delaying three system clock rising edges, two clock cycles for synchronization and one clock cycle for edge detector.

In input capture mode, if TIM_CMCn[ICRST] is set to 1, the counter value will be captured to CCVn register, and the counter will reset when an input capture event occurs. Writing to CCVn register is forbidden.

When a channel is configured as the input capture mode, this channel pin is an edge-sensitive input. And ELS field will determine which edge is sensitive, rising edge, falling edge, or both.

- ELS=01, rising edge sensitive
- ELS=10, falling edge sensitive
- ELS=11, both rising and falling edge sensitive

The following three figures describe the TIM behavior in rising, falling, and both edge capture mode when the core clock is selected as a counter clock.

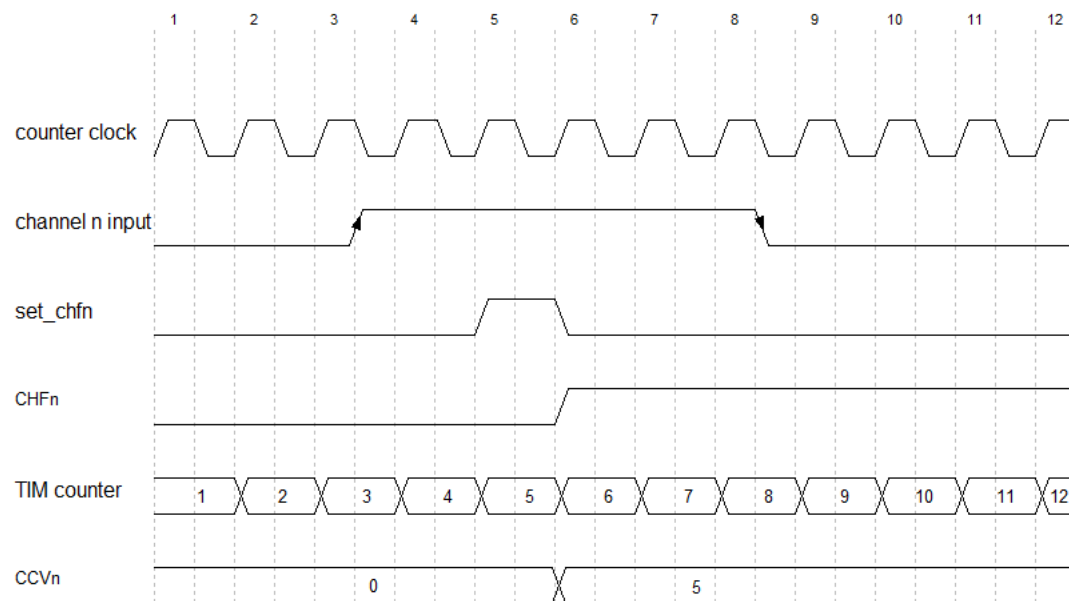


Figure 30-6 Capture input pulse in rising edge

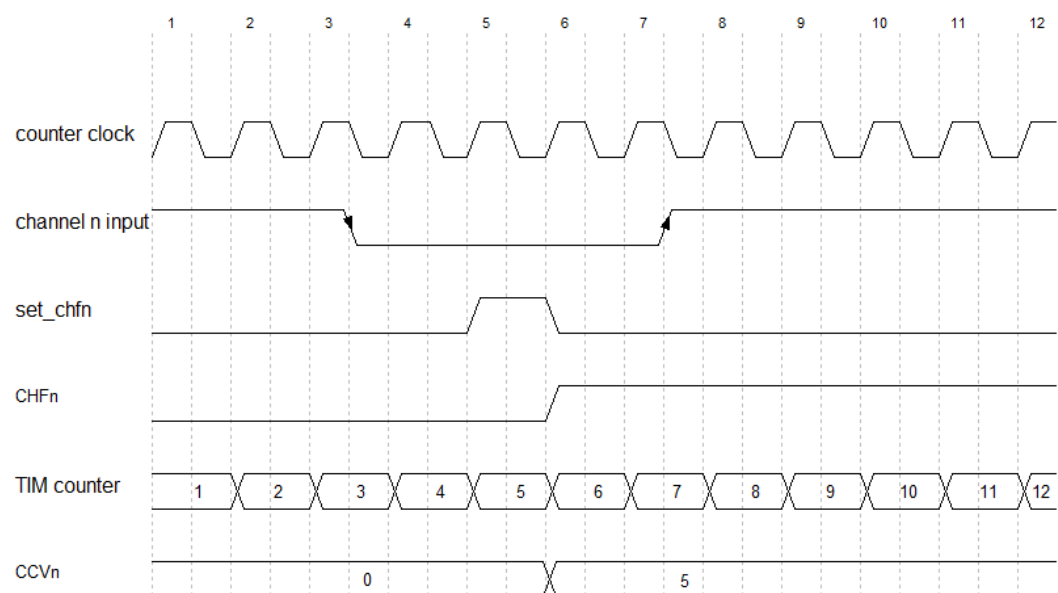


Figure 30-7 Capture input pulse in falling edge

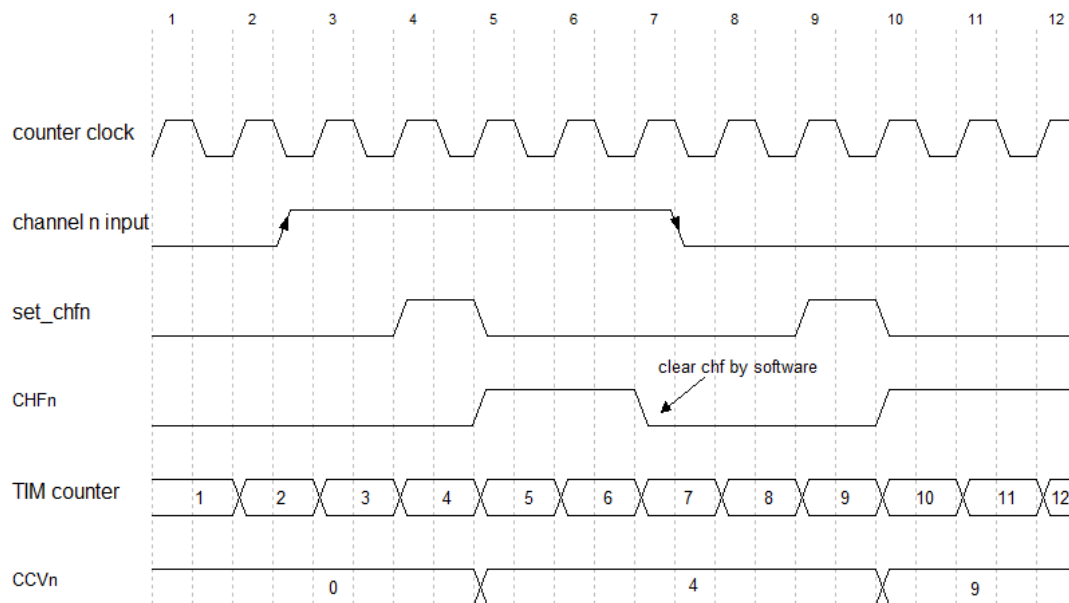


Figure 30-8 Capture input pulse in rising and falling edges

30.4.5 Dual-edge capture mode

The dual-edge capture mode is selected when

- DECAPEN = 1
- DECAP = 1

Before dual-edge capture, channel(n) and channel(n+1) CHF bit must be cleared and DECAP bit should be set to enable capture. The related DECAPEN bit selects channel(n) and channel(n+1) that work in pair, only channel(n) input signal is captured, channel(n+1) input signal is ignored, where $n = 0, 2, 4, 6$.

For one pair of channels, both even channel(n) and odd channel(n+1) ELS bits will select the sensitive edge that is captured. The table below shows the sensitive edge selection.

Table 30-6 Sensitive edge selection for dual-edge capture mode

ELS[1:0]	Channel Mode	Function
00	none	channel disabled
01	dual-edge capture	detect on rising edge
10		detect on falling edge
11		detect on rising and falling edges

For even channel n input signal, when the edge selected by ELS(n) is detected, channel(n) flag is set and the TIM counter value is captured into CCV(n) register. When the edge selected by ELS(n+1) is detected, channel(n+1) flag is set and the TIM counter value is captured into CCV(n+1) register. Therefore, if ELS(n) and ELS(n+1) select the same sensitive edge, it can be used for signal period measurement. On the other hand, if ELS(n) and ELS(n+1) select different sensitive edges, they can be used for signal pulse width measurement.

The channel(n) CMS[0] bit defines the dual-edge capture mode as one pulse capture or continuous pulse capture. The channel(n+1) CMS[0] bit does not make sense.

30.4.5.1 One pulse capture mode

When channel(n) CMS[0] = 0, it is one pulse capture mode. In this capture mode, only two

edges of channel(n) input are captured. ELS(n) selects the first captured edge and ELS(n+1) selects the second captured edge.

When the first edge selected by ELS(n) is detected in one pulse capture mode, channel(n) CHF is set. The counter value is captured into CCV(n) register when the second edge selected by ELS(n+1) is detected, channel(n+1) CHF is set, and the counter value is captured into CCV(n+1) register. At the same time, the DECAP bit is automatically cleared by hardware. Therefore, while the DECAP bit is set and the channel(n+1) flag is not set, these suggest the single capture is in process. When the DECAP bit is cleared and the channel(n+1) flag is set, these suggest that the single capture is done and the captured values in CCV(n) and CCV(n+1) registers are ready for reading.

See the details about the pulse width measurement and signal period measurement in one pulse capture mode in the following two figures.

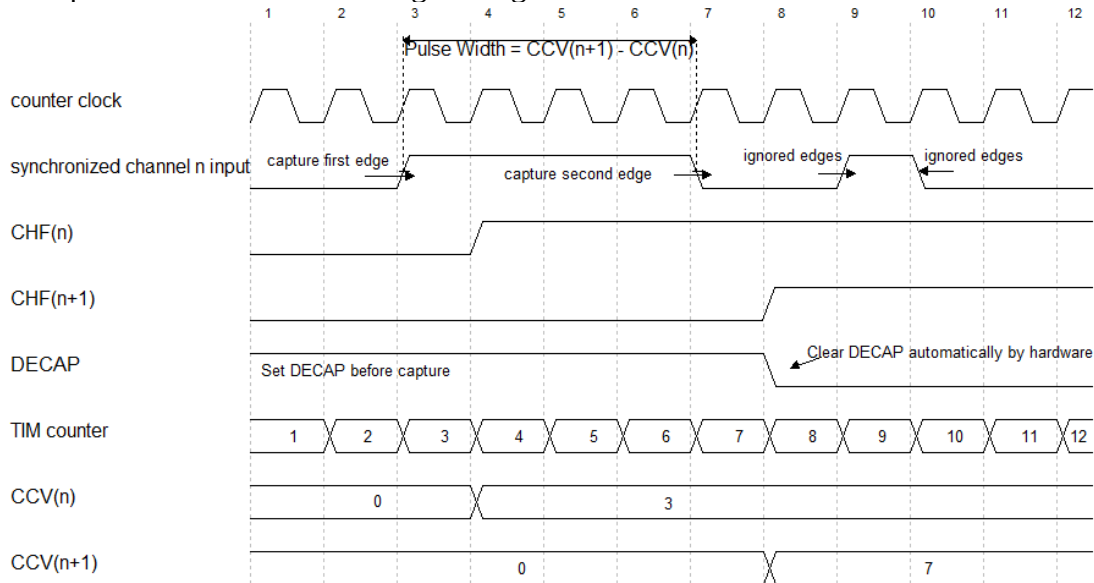


Figure 30-9 Pulse width measurement in one pulse capture mode (ELS(n)=01, ELS(n+1)=10)

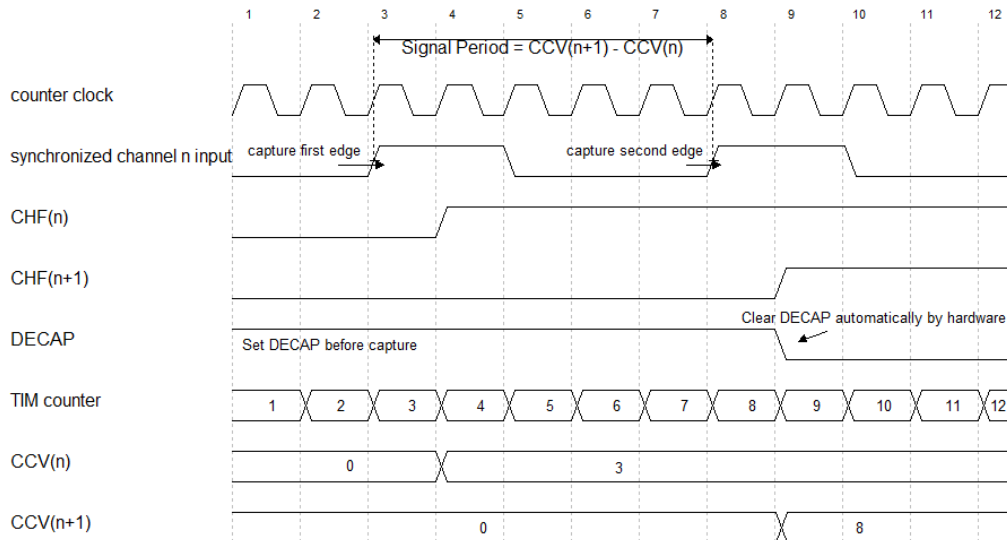


Figure 30-10 Signal period measurement in one pulse capture mode (ELS(n)=ELS(n+1)=01)

30.4.5.2 Continuous pulse capture mode

When channel(n) CMS[0] = 1, it is continuous pulse capture mode. In this capture mode, the edges of channel(n) input are captured continuously. ELS(n) selects the penult captured edge and ELS(n+1) selects the last captured edge.

In continuous pulse capture mode, when the first edge selected by ELS(n) is detected, channel(n) CHF is set and the counter value is captured into CCV(n) register. When the second edge selected by ELS(n+1) is detected, channel(n+1) CHF is set and the counter value is captured into CCV(n+1) register. But DECAP bit will not be cleared by hardware at this point. TIM will continue to capture the next edge. When channel(n+1) CHF is set, the captured values in CCV(n) and CCV(n+1) registers are ready for reading. When the next edge is detected, the newly captured value will override the CCV(n) and CCV(n+1) registers. Therefore, CCV(n) and CCV(n+1) register value is always the last captured value. See the details about pulse width measurement and signal period measurement in continuous pulse capture mode in the following two figures.

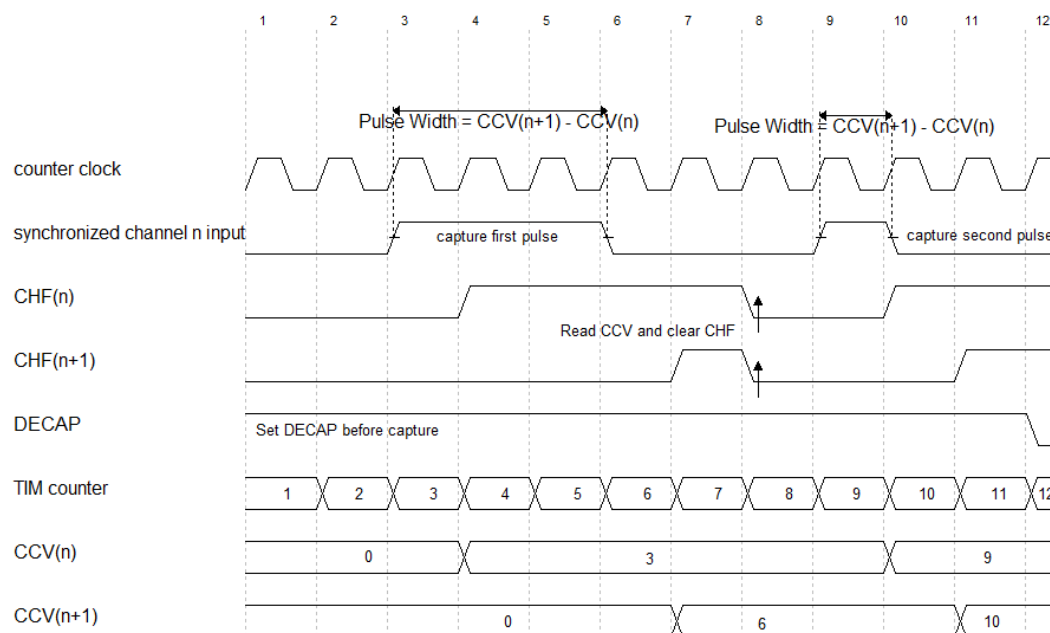


Figure 30-11 Pulse width measurement in continuous capture mode (ELS(n)=01, ELS(n+1)=10)

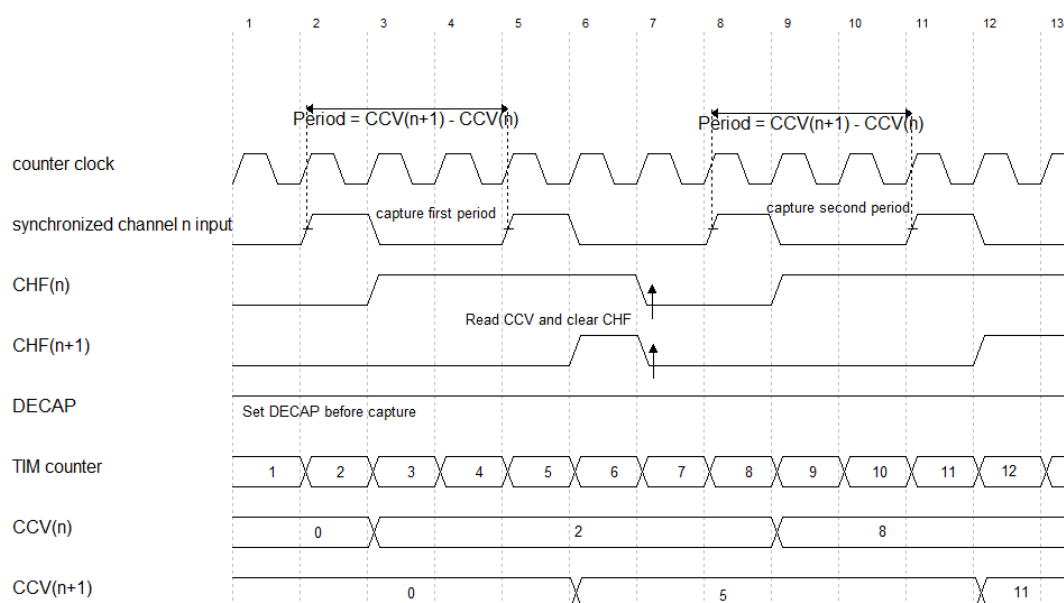


Figure 30-12 Signal period measurement in continuous capture mode (ELS(n)=ELS(n+1)=01)

In dual-edge capture mode, the CCV(n) register must be read prior to CCV(n+1) register to get proper values under the following conditions:

- Channel(n+1) CHF is set
- Captured values in CCV(n) and CCV(n+1) are ready for reading

NOTE

In input capture or dual-edge capture mode, If counter start-stop is needed, TIM capture enable is to configure TIM_CMCn[ELS] first and to start counter later; TIM capture disable is to clear TIM_CMCn[ELS] first and to stop counter later.

30.4.6 Input pulse filter for capture mode

For input capture mode and dual-edge capture mode, the user can select an input pulse filter for channel 0/1/2/3 that can be selected to filter glitch on input signals.

As displayed in the input capture diagram, an input pulse filter is implemented after synchronization. It is used to filter glitches on an input signal. When the configured FILVALn value is not zero, the channel n input filter is enabled. When the input filter is only available for channel 0/1/2/3, channel 4/5/6/7 are not supported.

The input filter implements a 4-bit up counting filter counter which is clocked by system clock divided by 4. When there is a state change in the channel input signal, the filter counter is reset and restarts to count.

- If the input signal state is stable long enough, the filter counter will continue to increment until reaching FILVALn. Then the input signal will be regarded as a valid pulse edge for the edge detector.
- If the input signal state changes before the filter counter reaches to FILVALn, the filter counter will reset to count, and the input signal will be regarded as a glitch and not passed to edge detector. As filter counter is clocked by 1/4 system clocks, any input pulse longer than $4 * \text{FILVALn} * (\text{system clocks})$ can pass through the filter, any input pulse shorter than $(4 * \text{FILVALn} - 1) * (\text{system clocks})$ will be filtered as a glitch, and input pulse between $4 * \text{FILVALn} * (\text{system clocks})$ and $(4 * \text{FILVALn} - 1) * (\text{system clocks})$ may pass through the filter.

The following figure shows the input filter output for a valid and invalid input pulse.

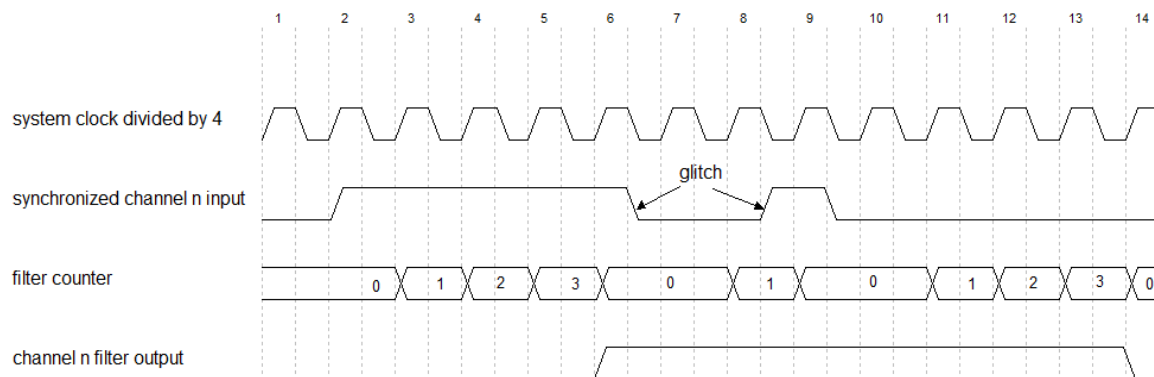


Figure 30-13 Input filter for input pulse (FILVALn=4)

30.4.7 Output compare mode

The output compare mode is selected by setting following configurations.

- DECAPEN = 0
- COMB = 0
- CNTMODE = 0
- CMS = 01

In an output compare mode, when the counter matches the value in CCVn register, channel n

output compare event occurs:

1. Channel n flag is set.
2. Channel n interrupt is generated if channel n CHIE is enabled and DMA is disabled.
3. Channel n DMA request is generated if channel n CHIE and DMA are enabled.
4. Channel n output can be set, cleared, or toggled based on ELS and POL configuration.

When a channel n output compare event occurs, channel n output is enabled if CHOEn is set.

- ELS = 00, channel output does not change on match.
- ELS = 01, channel output toggles on match.
- ELS = 10, channel output clears on match if channel polarity POLn = 0.
- ELS = 11, channel output sets on match if channel polarity POLn = 0.

The following three figures show the TIM behavior of output signal set, clear and toggle on match when channel n polarity is active high. In these figures, select the core clock as counter clock.

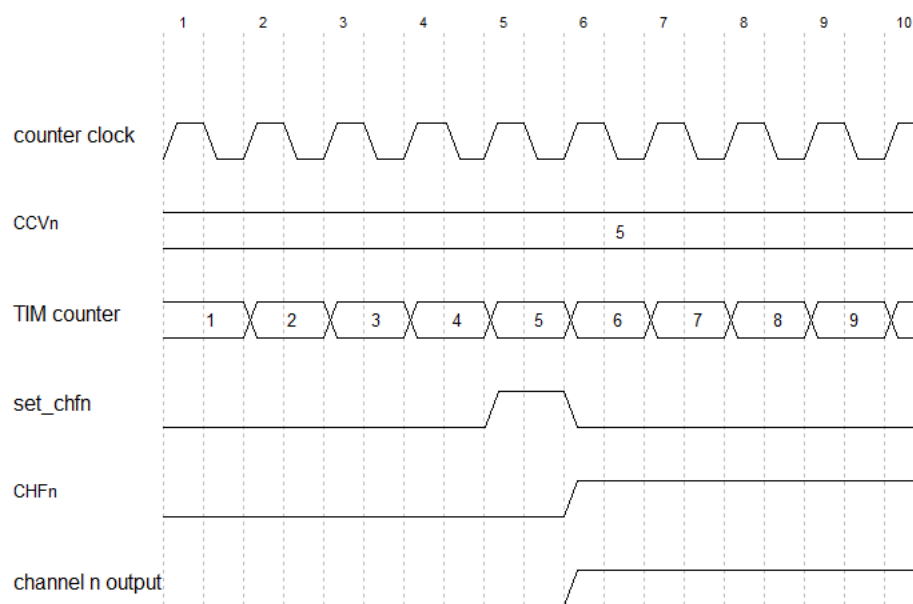


Figure 30-14 Output set on match(ELS=11)

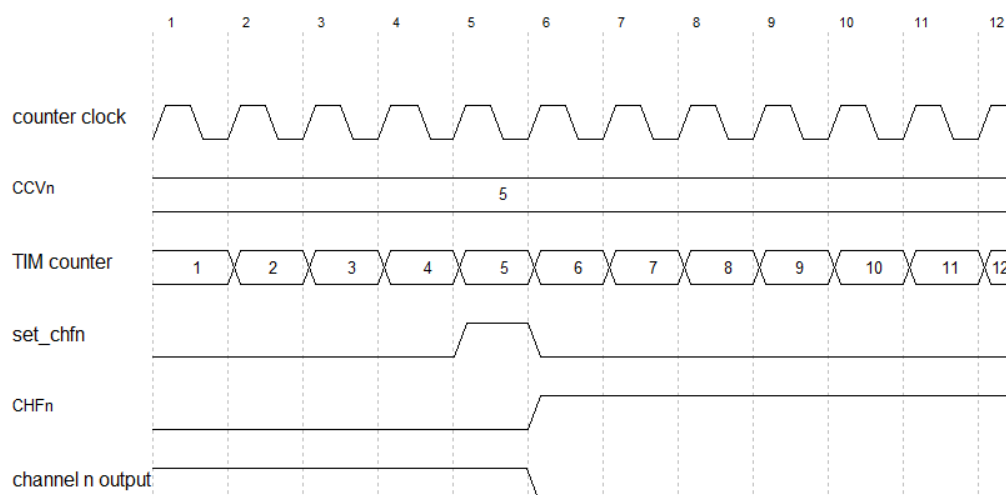
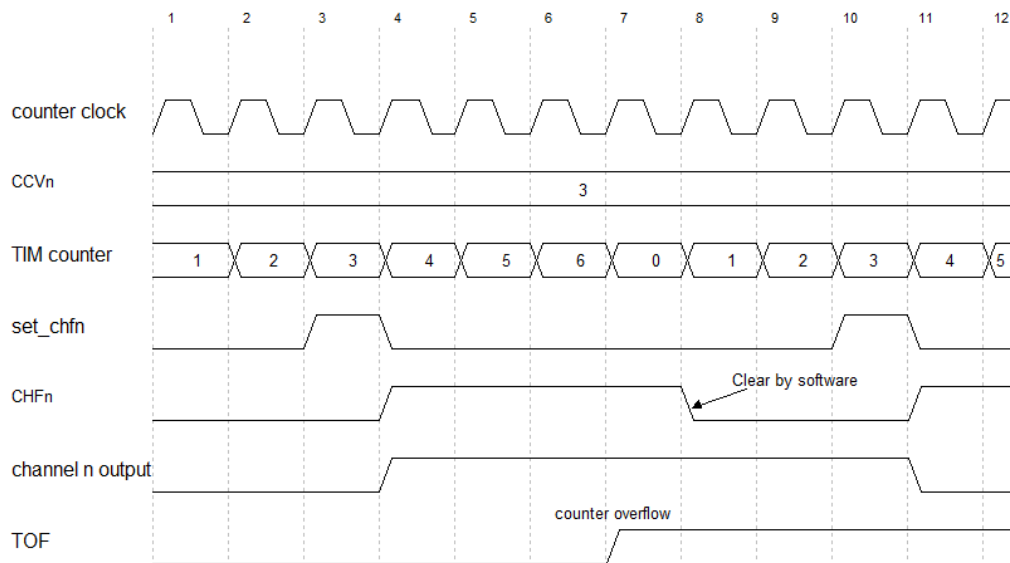


Figure 30-15 Output clear on match(ELS=10)**Figure 30-16 Output toggle on match(ELS=01)**

30.4.8 PWM output mode

Pulse Width Modulation (PWM) mode allows generating an output signal with configurable frequency and duty cycle. The PWM output mode can be selected independently on each channel or complementary on one pair of channels.

30.4.8.1 Center-aligned PWM output mode

The center-aligned PWM(CPWM) output mode is selected when:

- DECAPEN = 0
- COMB = 0
- CNTMODE = 1
- ELS $\neq$ 00

In CPWM output mode, the counter counts in up-down counting mode. The channel output sets or clears when the counter matches the CCV(n) value, so all channel PWM output pulse is center-aligned with CNTINIT value.

CPWM pulse can be selected as high true pulse or low true pulse according to ELS fields.

When ELS = 10, CPWM output signal is high true pulse. The counter increments from CNTINIT value when the counter reaches the CCV(n) value and counter is counting up, channel(n) output is forced to inactive state. When the counter reaches MOD value, the counter starts to decrement. When the counter reaches CCV(n) value again and counts down, channel(n) output is forced to active state. Therefore, the CPWM high true pulse width = $2 \times (\text{CCV}(n) - \text{CNTINIT})$ counter clock cycles, CPWM signal period = $2 \times (\text{MOD} - \text{CNTINIT})$ counter clock cycles.

No matter counter is counting up or counting down, when the counter matches with CCV(n) value, channel(n) flag is set and channel(n) interrupt is generated if CHIE(n) is enabled.

The following figure shows more details about CPWM high true pulse with POLn = 0.

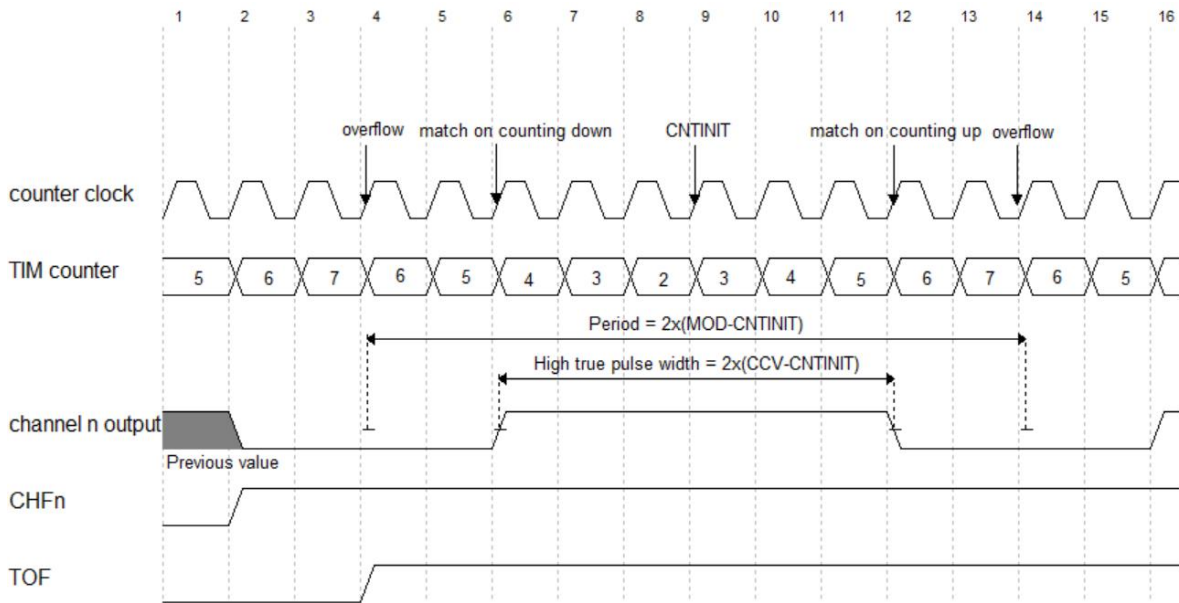


Figure 30-17 CPWM high true pulse with ELS=10 (MOD=7, CNTINIT=2, CCV(n)=5)

NOTE

When ELS=10:

- $CCV(n) = CNTINIT = 0$ or $CCV(n)[15] = 1$ will generate 0% duty cycle CPWM output.
- $CCV(n)[15] = 0$ and $CCV(n) \geq MOD$ will generate 100% duty cycle CPWM output.

When ELS = x1, CPWM output signal is low true pulse. The counter increments from the CNTINIT value. When the counter reaches the CCV(n) value, and the counter is counting up, channel(n) output is forced to an active state. When the counter reaches MOD value, the counter starts to decrement. When the counter reaches CCV(n) value again, and the counter is counting down, channel(n) output is forced to inactive state. Therefore, the CPWM low true pulse width = $2 \times (CCV(n) - CNTINIT)$ counter clock cycles, CPWM signal period = $2 \times (MOD - CNTINIT)$ counter clock cycles.

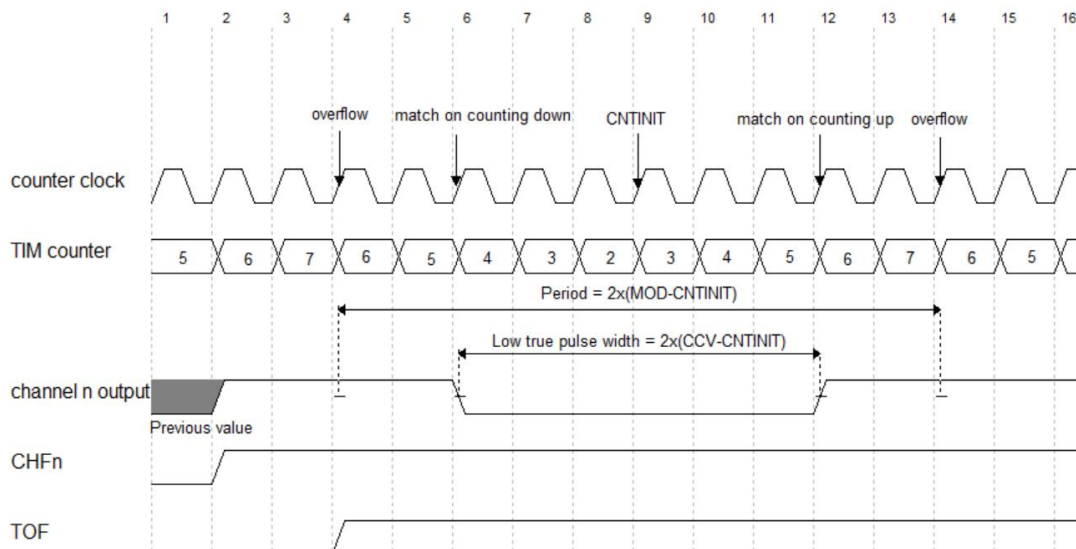


Figure 30-18 CPWM low true pulse with ELS=x1 (MOD=7, CNTINIT=2, CCV(n)=5)

NOTE

When ELS=x1:

$CCV(n) = CNTINIT = 0$ or $CCV(n)[15] = 1$ will generate 100% duty cycle CPWM output.

$CCV(n)[15]=0$ and $CCV(n) \geq MOD$ will generate 0% duty cycle CPWM output.

30.4.8.2 Edge-aligned PWM output mode

The edge-aligned PWM (EPWM) output mode is selected when:

- $DECAPEN = 0$
- $COMB = 0$
- $CNTMODE = 0$
- $CMS = 1x$
- $ELS \neq 00$

In EPWM output mode, the counter is counting in up counting mode. The channel output is set or clear when counter reaches the $CCV(n)$ value and $CNTINIT$ values, so all channel PWM output is edge-aligned with $CNTINIT$ value.

EPWM pulse can be selected as a high true pulse or a low true pulse according to ELS fields. When ELS equals 10, the EPWM output signal is a high true pulse. The counter increments from $CNTINIT$ value. When the counter reaches the $CCV(n)$ value, channel(n) output is forced to an inactive state. When the counter reaches MOD value, counter restarts to increment from $CNTINIT$, channel(n) output is forced to an active state when the counter reaches $CNTINIT$ value. Therefore, the EPWM high true pulse width = $(CCV(n) - CNTINIT)$ counter clock cycles, EPWM signal period = $(MOD - CNTINIT + 1)$ counter clock cycles.

When counter matches the $CCV(n)$ value, channel flag CHF is set, and channel interrupt is generated if $CHIE$ is enabled.

The following figure shows the details of the EPWM high true pulse with $POLn = 0$.

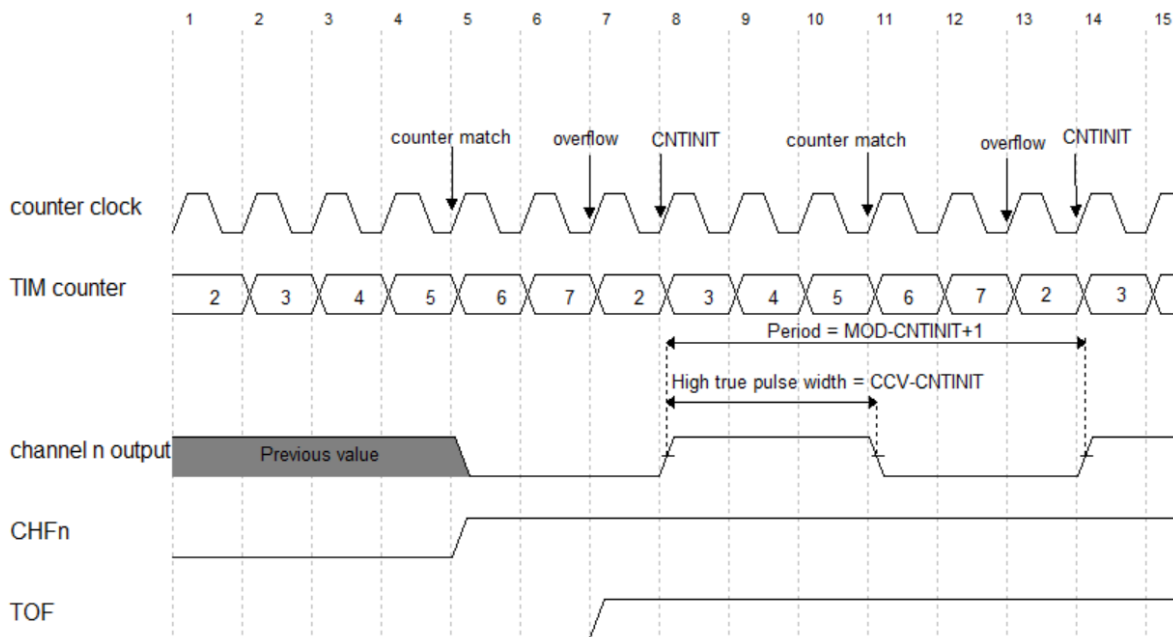


Figure 30-19 EPWM high true pulse with $ELS=10$ ($MOD=7$, $CNTINIT=2$, $CCV(n)=5$)

NOTE

When $ELS=10$:

- $CCV(n) = CNTINIT = 0$ will generate 0% duty cycle EPWM output.
- $CCV(n) > MOD$ will generate 100% duty cycle EPWM output.

When $ELS = x1$, the EPWM output signal is low true pulse. Counter increments from $CNTINIT$ value. When the counter reaches the $CCV(n)$ value, channel(n) output is forced to an active state. When the counter reaches MOD value, counter restarts to increment from $CNTINIT$, channel(n) output is forced to an inactive state at the point of counter reaching $CNTINIT$ value. Therefore, the EPWM low true pulse width = $(CCV(n) - CNTINIT)$ counter

clock cycles, EPWM signal period = $(MOD - CNTINIT + 1)$ counter clock cycles.

When counter matches the $CCV(n)$ value, channel flag CHF is set and channel interrupt is generated if $CHIE$ is enabled. The following figure shows more details about the EPWM low true pulse with $POLn = 0$.

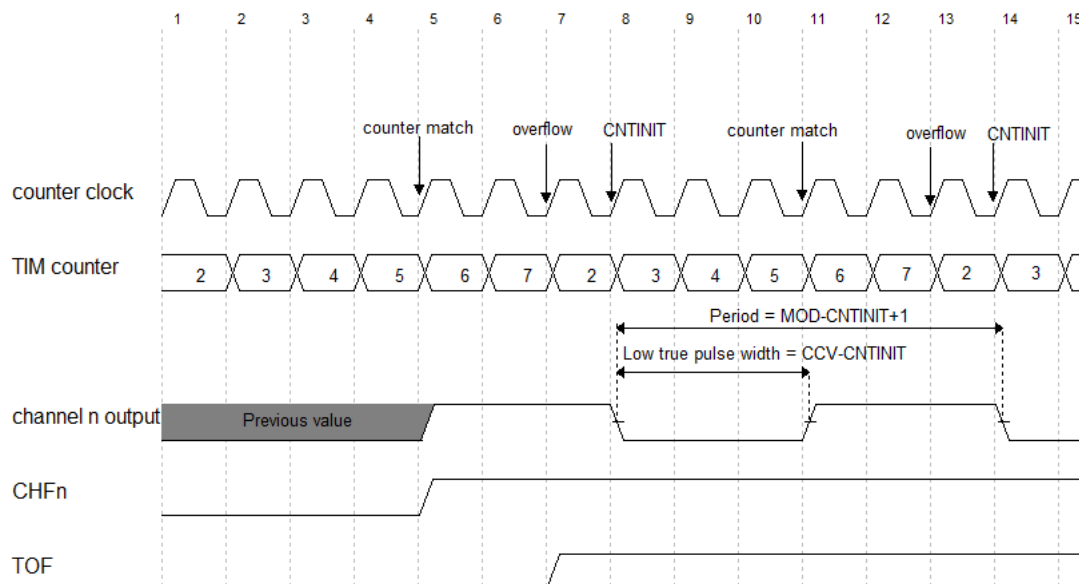


Figure 30-20 EPWM low true pulse with $ELS=x1$ ($MOD=7$, $CNTINIT=2$, $CCV(n)=5$)

NOTE

When $ELS=x1$:

- $CCV(n) = CNTINIT = 0$ will generate 100% duty cycle EPWM output.
- $CCV(n) > MOD$ will generate 0% duty cycle EPWM output.

30.4.8.3 Complementary PWM output mode

In PWM output mode, channels can be combined as pairs to work in complementary mode. In complementary mode, an even channel(n) and adjacent odd channel(n+1) are combined to generate a pair of complementary PWM signals.

The complementary mode is selected when:

- $DECAPEN = 0$
- $COMB = 1$
- $CNTMODE = 0$

In complementary mode, counter must be selected to up counting mode. Channel(n+1) output is inverse of channel(n) output when $POL(n)$ is equal to $POL(n+1)$.

In this mode, even channel(n) ELS bits, $CCV(n)$ value and odd channel(n+1) $CCV(n+1)$ value are used to generate complementary PWM output. The odd channel(n+1) ELS bits do not make sense.

See the following table for details.

Table 30-7 Channel output on complementary mode

ELS(n)[1:0]	Counter Match CCV(n)		Counter Match CCV(n+1)	
	channel(n) output	channel(n+1) output	channel(n) output	channel(n+1) output
10	force to active	force to inactive	force to inactive	force to active
x1	force to inactive	force to active	force to active	force to inactive

If channel(n) ELS = 10, when the counter matches the CCV(n) value, channel(n) output is forced to active state and channel(n) flag is set. When the counter matches the CCV(n+1) value, channel(n) output is forced to inactive state and channel(n+1) flag is set. Channel(n+1) output is always inverse of channel(n) output when POL(n) is equal to POL(n+1). Therefore, for channel(n) PWM output, the high true pulse width = $|CCV(n+1) - CCV(n)|$ counter clock cycles, the period = $(MOD - CNTINIT + 1)$ counter clock cycles. The following figure shows more details about $POL(n) = POL(n+1) = 0$.

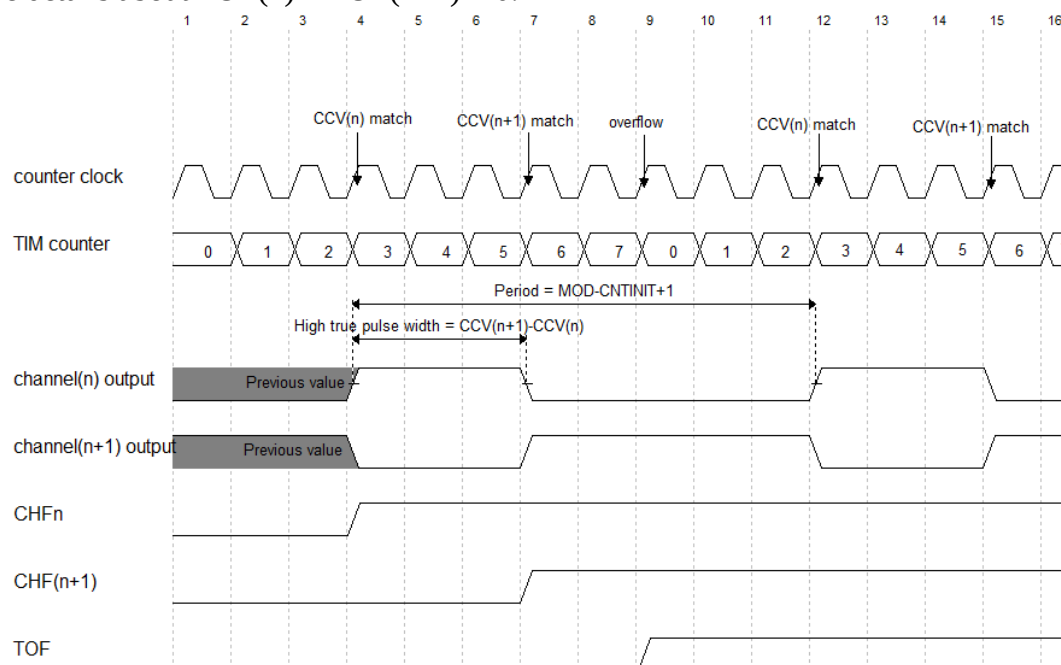


Figure 30-21 Complementary output when ELS = 10 (MOD=7, CCV(n)=2, CCV(n+1)=5)

If channel(n) ELS = x1, when the counter matches the CCV(n) value, channel(n) output is forced to inactive state and channel(n) flag is set. When the counter matches the CCV(n+1) value, channel(n) output is forced to active state and channel(n+1) flag is set. Channel(n+1) output is always inverse of channel(n) output when POL(n) is equal to POL(n). Therefore, for channel(n) PWM output, the low true pulse width = $|CCV(n+1) - CCV(n)|$ counter clock cycles, the period = $(MOD - CNTINIT + 1)$ counter clock cycles. See the following figure for more details.

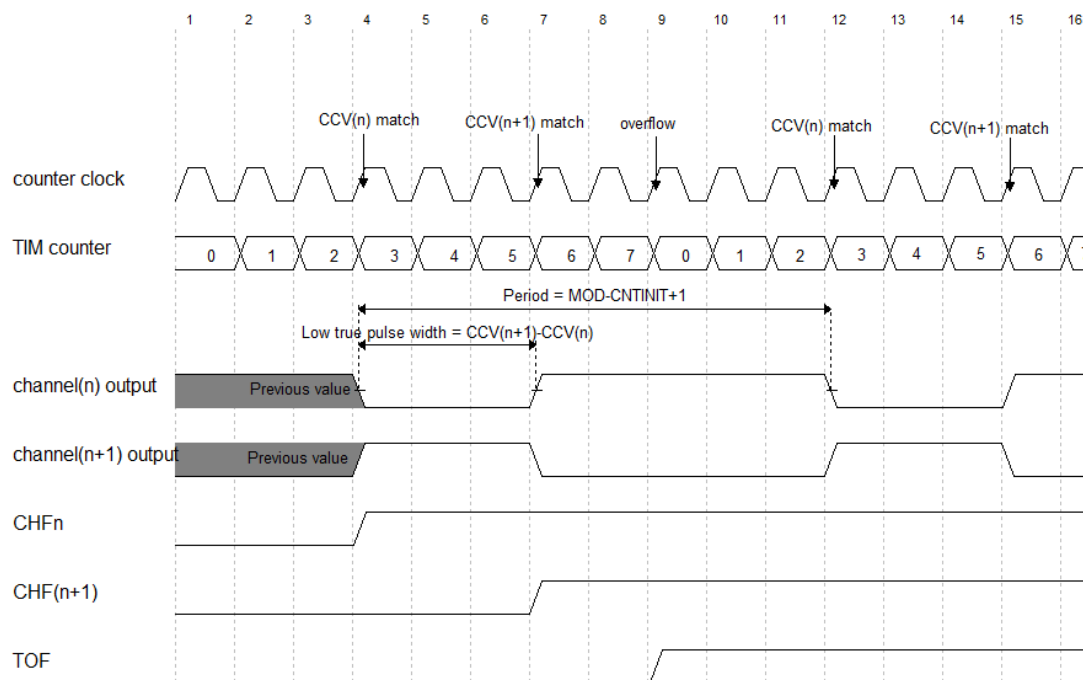


Figure 30-22 Complementary output when ELS = x1 (MOD=7, CCV(n)=2, CCV(n+1)=5)

NOTE

In the complementary mode, to generate one pair of expected complementary PWM output, it is recommended to configure $CNTINIT < CCV(n) < CCV(n+1) < MOD$, $POL(n) = POL(n+1)$.

30.4.8.4 Deadtime insertion for complementary PWM mode

For a complementary PWM output application in motor control, when the counter matches CCV(n) or CCV(n+1) value, it is allowed to keep channel(n) and channel(n+1) output in an inactive state for a short time before they are forced to active state. This inactive state is defined by channel polarity POL bit. This delay time is called deadtime delay region. During the deadtime delay region, channel(n) and channel(n+1) outputs are both inactive. The deadtime delay ensures that no two complementary output signals drive the active state at the same time.

The deadtime insertion is enabled when DTEN = 1 and DTVAL is not zero. A 6-bit deadtime counter is implemented to realize this delay. The system clock clocks this after the divide. DTPS defines the divided value. When the deadtime counter reaches the value of DTVAL, the delay ends. Therefore, the deadtime delay period is $DTVAL * DTPS$ system clock cycles. For example, DTVAL=3, DTPS=2, which represents the system clock is divided by 4, so deadtime delay is $3 * 4 = 12$ system clock cycles. See the following table for more details.

Table 30-8 Deadtime delay in complementary mode with $POL(n)=POL(n+1)=0$

ELS(n)[1:0]	Counter Match CCV(n)		Counter Match CCV(n+1)	
	channel(n) output	channel(n+1) output	channel(n) output	channel(n+1) output
10	delay before forced to 1	forced to 0 immediately	forced to 0 immediately	delay before forced to 1
x1	forced to 0 immediately	delay before forced to 1	delay before forced to 1	forced to 0 immediately

From the table, it can be seen that when $POL(n) = POL(n+1) = 0$, channel(n) and channel(n+1) will delay when switching to active state logic one, while they will not delay when switching to an inactive state logic zero. If $POL(n) = POL(n+1) = 1$, channel(n) and channel(n+1) will delay when switching to logic zero and not delay when switching to logic

1. See the following figure for more details about the deadtime delay. Core clock is selected as timer counter clock in the figure.

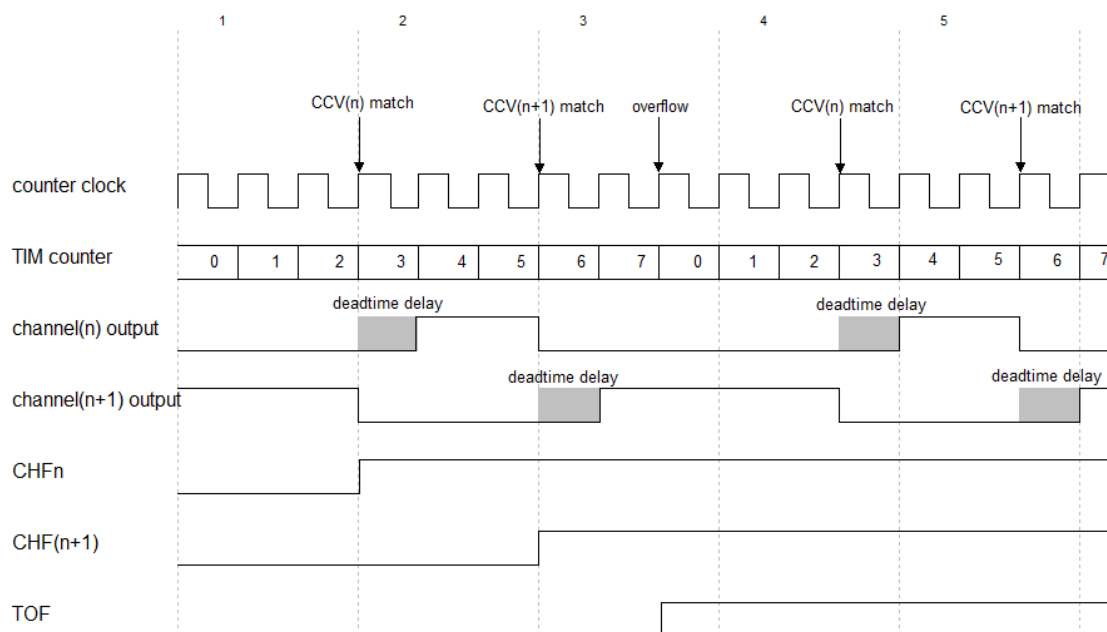


Figure 30-23 Deadtime delay with $ELS(n) = 10$ and $POL(n) = POL(n+1) = 0$

30.4.9 Generation of match trigger and initial trigger

When the counter matches the value in CCV(n) register, the TIM is allowed to generate a match trigger if channel(n) TRIGE is enabled. This match trigger is one system period width pulse and is used to trig other modules, such as ADC and CMP modules.

When the counter reaches a reload point, half or full cycle reload, the TIM is allowed to generate an initial trigger if INITRIGE is enabled. This initial trigger is also one system period width pulse and is used to trig other modules.

The following figure shows more details about the match trigger and initial trigger generation case. Core clock is selected as timer counter clock in the figure.

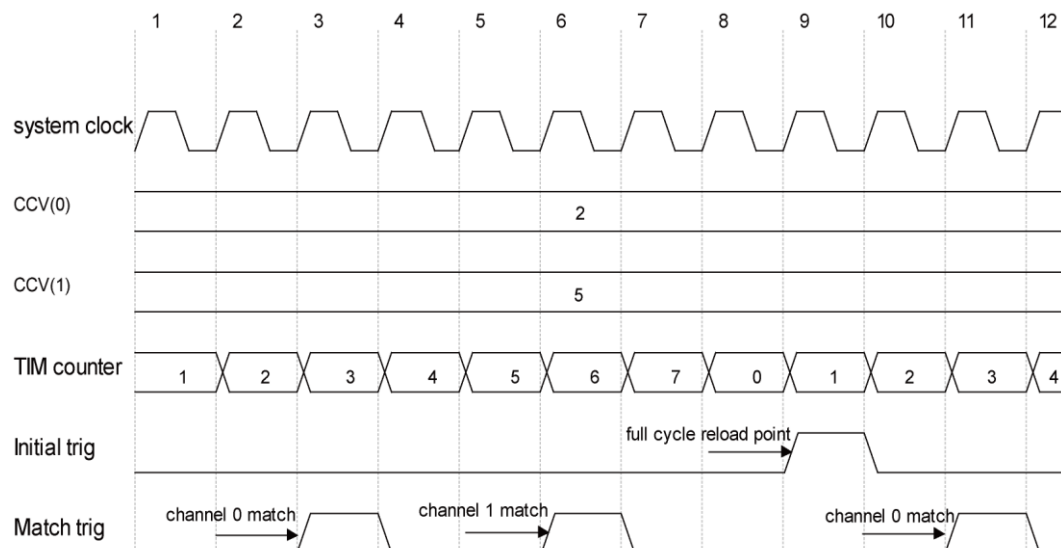


Figure 30-24 Match trigger and initial trigger generation

30.4.10 Channel output software control

The channel output can be controlled by software when:

- $DECAPEN = 0$
- $CH(n)OC = 1$

The $CH(n)OC$ bit enables the channel output software control. When this bit is enabled, channel(n) software control value $CH(n)OCV$ is forced to channel(n) output.

- When channel(n) and channel(n+1) work in independent mode ($COMB = 0$), software output control directly forces $CH(n)OCV$ value to channel(n) output and forces $CH(n+1)OCV$ to channel(n+1) output. The outputs for channel(n) and channel(n+1) are independent.
- When channel(n) and channel(n+1) work in complementary mode ($COMB = 1$), software output control directly forces $CH(n)OCV$ value to channel(n) output. Channel(n+1) value depends on $CH(n)OCV$ and $CH(n+1)OCV$ values. See the following table for more details.

Table 30-9 Channel software output control when $COMB = 1$

$CH(n)OCV$	$CH(n+1)OCV$	Channel(n) output	Channel(n+1) output
0	0	inactive state	inactive state
0	1	inactive state	active state
1	x	active state	forced to inactive state

NOTE

For complementary mode, $CH(n)OC$ and $CH(n+1)OC$ bits should be the same and $COMB$ bit should NOT be modified after software output control is enabled to avoid getting the unexpected result. Besides, $CH(n)OCV$ and $CH(n+1)OCV$ will not be active state at the same time, if $CH(n)OCV$ is set to 1, no matter what $CH(n+1)OCV$ has been set to, $CH(n+1)OCV$ will be forced to output inactive state.

30.4.11 Fault input control

For motor control application, TIM implements two fault input channels from pins. When an active status is detected in fault input pin, fault flag is set. The fault interrupt can be generated if $FLTIE$ is enabled, the channel outputs are forced to their safe values, channels output safe

value depends on channel polarity POLn. See the following fault input control block diagram for more details.

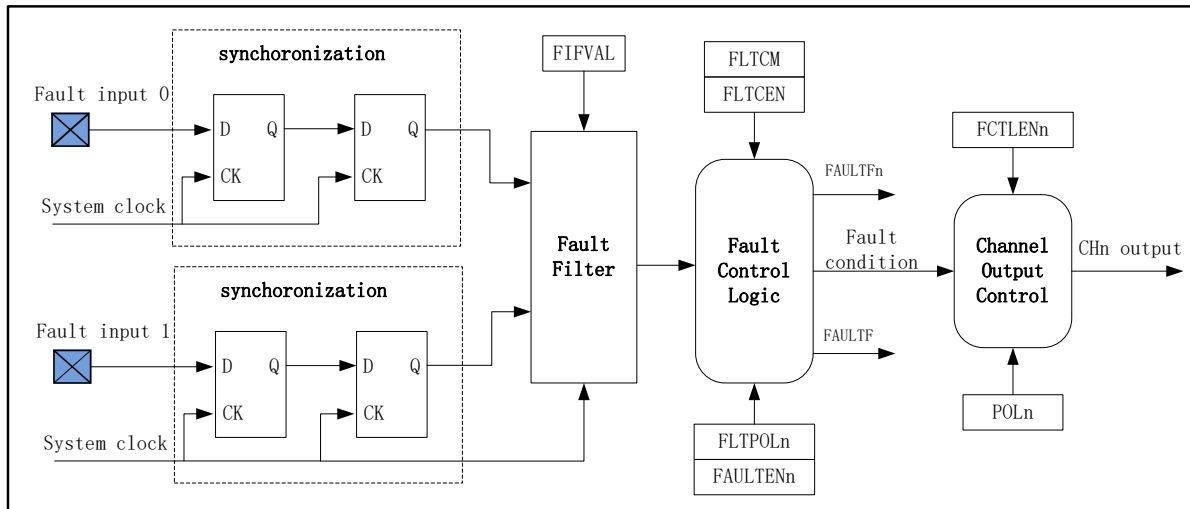


Figure 30-25 Fault input control diagram

For fault input channel(n), the fault input control is enabled when FLTCEN and FAULTENn are both enabled. From the block figure, it can be seen that the fault input n will pass through two flip-flops to be synchronized by system clock, then enter into fault input filter.

Only when FIFENn is enabled and FIFVAL is not zero is this fault input filter enabled. Similar with input pulse filter in capture mode, a 4-bit counter is implemented in fault filter, this counter will increment when fault input is stable and reset when fault input state changes. Therefore, any fault input pulse shorter than FIFVAL*(system clock cycle) will be regarded as a glitch and not pass to next stage. If fault filter is not enabled, the synchronized fault input will directly pass to fault polarity control.

The filtered fault pulse passes to polarity control. The fault input polarity can be configured by FLTPOL0 and FLTPOL1, the active state in filtered fault input will be detected as a fault event. Then related fault flag is set and fault interrupt will be generated if FLTIE is enabled.

When one of the fault inputs is detected, the fault condition is asserted. If the output channel fault control FCTLENn is enabled, then the pairs of channel(n) and channel(n+1) output will be forced to inactive status. Channel(n) output inactive status is defined by POL(n) and channel(n+1) output inactive status is defined by POL(n+1). A fault event occurring in any fault input can force output of all 8 channels to inactive status if all the FCTLENn bits are enabled.

When all fault inputs are deasserted, the fault condition depends on fault clear mode FLTCM. When FLTCM = 1, automatical fault clear mode is selected. When all fault inputs are deasserted, the fault condition deasserts soon and the channels output will resume to generate new PWM output when counter overflows and restarts to count. See the following figure for details about EPWM output with automatical fault clear mode.

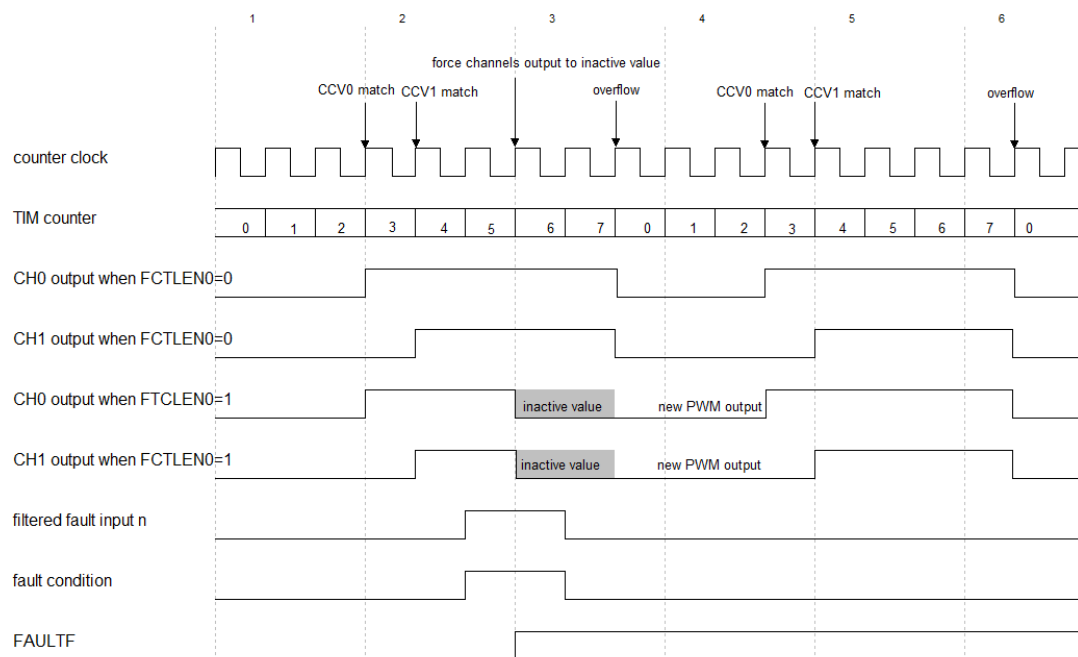


Figure 30-26 EPWM output for automatic fault clear mode, POL0=POL1=0, FSTATE=0

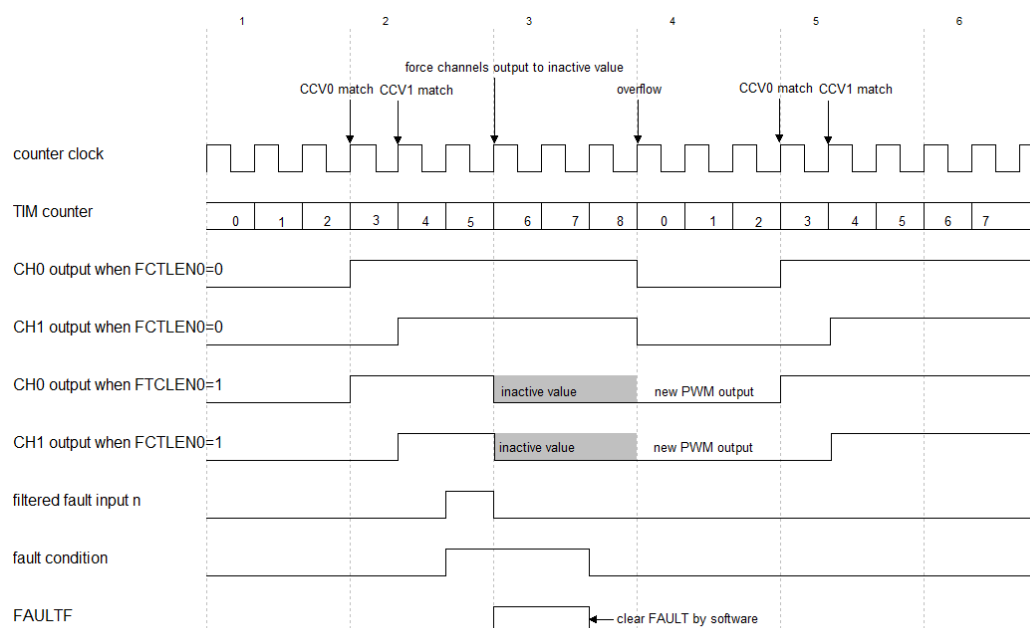


Figure 30-27 EPWM output when fault control is enabled, POL0=POL1=0, FSTATE=0

- FCTLEN0 controls the outputs of channel 0 and channel 1.
- FCTLEN1 controls the outputs of channel 2 and channel 3.
- FCTLEN2 controls the outputs of channel 4 and channel 5.
- FCTLEN3 controls the outputs of channel 6 and channel 7.

When FLTCM = 0, manual fault clear mode is selected. When all fault inputs are deasserted, the fault condition deasserts when fault flag is cleared by software manually. After fault flag is cleared, output of the channels will resume to generate new PWM output when counter overflows and restarts to count. See the following figure for details about EPWM output with manual fault clear mode.

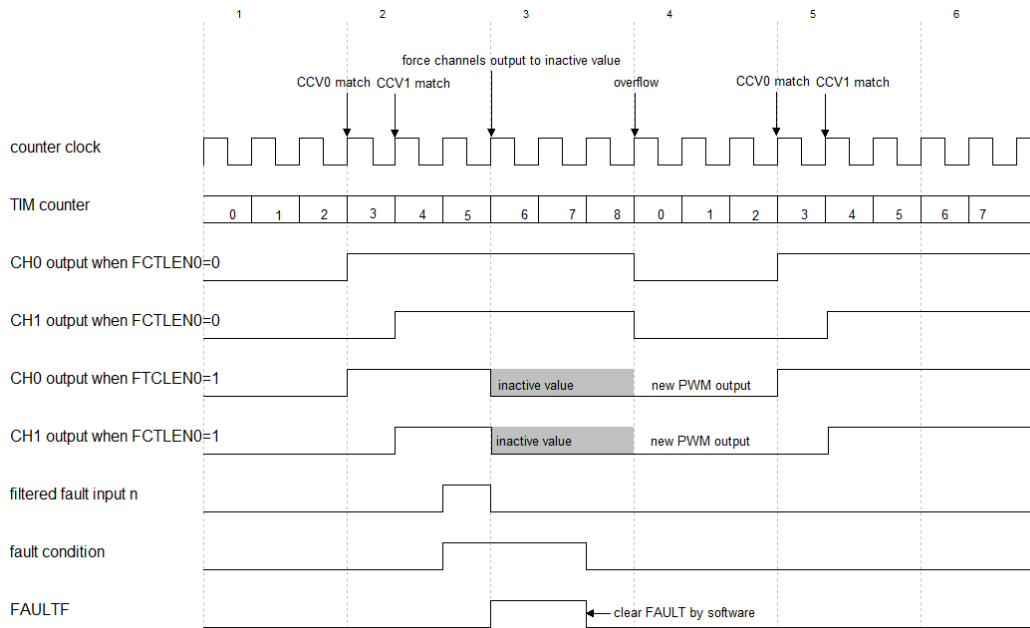


Figure 30-28 EPWM output for manual fault clear mode, POL0=POL1=0, FSTATE=0

30.4.12 Registers synchronization

The TIM registers CNTINIT, MOD, CCV(n), OUTSWCR and HCV include a write buffer structure.

When the 16-bit TIM counter is off, that is, CKSRC[1:0] = 2'b00, when the software writes a value to these registers, it will directly update these registers' value.

But when the counter is counting, that is, CKSRC[1:0] ≠ 2'b00, writing a value to these registers will not update these registers' value immediately. However, it will write values to their write buffers instead. And these registers are allowed to be synchronized to their write buffer value by different synchronization strategies.

30.4.12.1 CNTINIT register synchronization

The CNTINIT register is allowed to update from its write buffer by three strategies according to different configurations:

1. Synchronization at the next system clock after the CNTINIT was written when:
 - SYNCINIT = 0
2. Synchronization by PWM synchronization software trigger when:
 - SYNCINIT = 1
 - SWRSTCNT = 1
 - SWWRBUF = 1
3. Synchronization by PWM reload when:
 - SYNCINIT = 1
 - LOADEN = 1

The CNTINIT register is configured as synchronization at the next system clock. After CNTINIT register is written, the write value is loaded into its write buffer and CNTINIT register will update from its write buffer on the next rising edge of system clock.

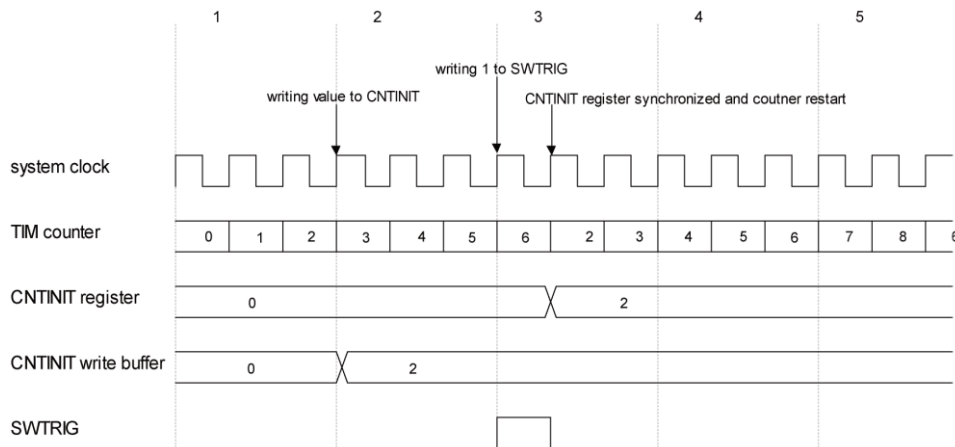


Figure 30-29 CNTINIT register synchronization by software trigger

1. When the CNTINIT register is configured as synchronization by PWM synchronization software trigger, and after CNTINIT register is written, the write value is loaded into its write buffer.
2. Then writing a 1 to SWTRIG bit will generate one system clock cycle pulse software trigger to trigger CNTINIT register synchronization.
3. SWTRIG bit will be automatically cleared by hardware on the next rising edge of the system clock.
4. After SWTRIG is asserted, CNTINIT register will update from its write buffer and the counter restarts to count. See the details in the figure above.

When the CNTINIT register is configured as synchronization by PWM reload, after the CNTINIT register is written, the write value is loaded into its write buffer, and the CNTINIT register will update from its write buffer at the following reload opportunities:

- Full cycle reload opportunity, that is, when TIM counter reaches the value of MOD.
- Half cycle reload opportunity, that is, when HCSEL is set to 1 and TIM counter reaches the value of HCV.
- Channel match reload opportunity, that is, when CHnSEL is set to 1 and TIM counter reaches the value of CCV(n).

In the PWM reload, registers complete the reload from their write buffers when reload flag RLDF is set. However, the RLDF is not set at every reload opportunity. It depends on the configured reload frequency LDFREQ[4:0] fields.

- LDFREQ = n indicates RLDF will be set when every (n+1) reload opportunities occur.
- LDFREQ = 2, then RLDF is set when every 3 reload opportunities occur.
- If LDFREQ is default value zero, then RLDF is set at every reload opportunity.

When RLDF is set, reload interrupt can be generated if the reload interrupt enable RLDIE is enabled. In addition, when CKSRC[1:0] is selected for the first time, the RLDF is also set. However, we still do not regard this as a reload opportunity, because before CKSRC[1:0] is selected, counter is off. Thus, any writing to these registers will directly update the register value instead of loading the writing value into their write buffers.

30.4.12.2 MOD and HCV registers synchronization

MOD and HCV registers are allowed to update from their write buffers by two strategies according to different configurations:

1. Synchronization by PWM synchronization software trigger when:
 - SWRSTCNT = 1

- SWWRBUF = 1
2. Synchronization by PWM reload when:
 - LOADEN = 1

The PWM synchronization software trigger and PWM reload mechanisms of the MOD and HCV registers are the same as those of the CNTINIT register.

30.4.12.3 CCV(n) register synchronization

CCV(n) register is allowed to update from its write buffer by two strategies according to different configurations:

1. Synchronization by PWM synchronization software trigger when:
 - SYNCENn = 1
 - SWRSTCNT = 1
 - SWWRBUF = 1
2. Synchronization by PWM reload when:
 - SYNCENn = 1
 - LOADEN = 1

The PWM synchronization software trigger and PWM reload mechanism of CCV(n) register are also the same with those of the CNTINIT register. The SYNCENn (where n=0,1,2,3) bit enables one pair of channel CCV registers synchronization.

- SYNCEN0 enables CCV0 and CCV1 registers synchronization.
- SYNCEN1 enables CCV2 and CCV3 registers synchronization.
- SYNCEN2 enables CCV4 and CCV5 registers synchronization.
- SYNCEN3 enables CCV6 and CCV7 registers synchronization.
-

Note: due to PWM reload mechanism, two cases that may lead to unpredictable result are listed as below:

- In the current EPWM period, CCV match reload, full cycle reload and half cycle reload are enabled, the current CCV is larger than HCV and the new CCV is not larger than HCV. EPWM abnormal duty ratio occurs when CCV is modified into new CCV during counting from CNTINIT to HCV.
- In the current CPWM period, before the first time matching CCV, CCV is modified into new value, after that, no reload event happens before the first time matching CCV. If the CCV reload even occurs between the first and second time matching CCV, CPWM unexpected duty ratio occurs.

30.4.12.4 OUTSWCR register synchronization

OUTSWCR register is allowed to update from its write buffer by two strategies according to different configurations:

1. Synchronization at the next system clock after OUTSWCR was written when:
 - SYNCOSWC = 0
2. Synchronization by PWM synchronization software trigger when:
 - SYNCOSWC = 1
 - SWRSTCNT = 1
 - SWWRBUF = 1

OUTSWCR register synchronization at the next system clock is the same as the CNTINIT register. Thus, after the OUTSWCR register is written, the write value is loaded into its write buffer, and the OUTSWCR register will update from its write buffer on the next rising edge of system clock.

PWM synchronization software trigger mechanism of OUTSWCR register is also the same as that of the CNTINIT register.

Synchronization by PWM reload is not supported for the OUTSWCR register.

30.4.13 Interrupt and DMA signals

According to the following table, the interrupt and DMA request lines of the TIM module are mapped to the internal flags.

Table 30-10 Assignment of Interrupt Request and DMA Request Lines

IRQ/DMA	Flag	Interrupt Description
IRQ		
TIM channel[7:0] interrupt	CHF	TIM channel interrupt
TIM fault interrupt	FAULTF	TIM fault interrupt
TIM overflow interrupt	TOF	TIM counter overflow interrupt
TIM reload interrupt	RLDF	TIM reload interrupt
DMA		
TIM DMA request	CHF	DMA transfer request

30.4.13.1 DMA

Every TIM channel supports generating a DMA request to activate DMA transfer. Channel n DMA request is enabled when channel n CMC[DMA] and CMC[CHIE] are both enabled. After channel n channel flag CHF_n is asserted, the channel n DMA request is generated.

When the channel n DMA request is asserted, channel n flag CHF_n is cleared automatically after this channel DMA transfer is done, then DMA request is deasserted.

30.4.13.2 Interrupt

TIM can generate the following four different interrupts:

- Timer overflow interrupt. This interrupt is generated when TOIE is enabled, and TOF is asserted.
- Channel interrupt. This interrupt is generated when CHIE_n is enabled, but DMA_n is disabled, and CHF_n is asserted.
- Fault interrupt. This interrupt is generated when FLTIE is enabled, and FAULTF is asserted.
- Reload interrupt. This interrupt is generated when RLDIE is enabled, and RLDF is asserted.

Chapter 31

System Timer (STIM)

31.1 Introduction

This chapter defines a System Timer (STIM).

- System Timer (STIM) implements 4-channel 32-bit counter. For every counter channel, four clock sources can be selected as the STIM counter clock source.
- This STIM counter clock source will clock the prescaler and glitch filter.
- The clock divided by prescaler will clock the counter.
- Each channel counter can work in the free counting mode or pulse counting mode.
- Match trigger will be generated when the counter matches a configurable value.

31.1.1 Features

The STIM supports the following key features:

4 independent channels to service:

- Each channel implements a 32-bit counter.
- Four clock sources can be selected for each channel. The clock sources are specified by the SoC integration guide.
- Four input pulses can be selected for each channel. Three pulses come from pads, and one pulse comes from Trigger Mux Unit (TMU).
- Each channel counter can work as a free counting mode or pulse counting mode.
- In a free counting mode, counter increments on the rising edge of the divided clock source.
- In a pulse counter mode, counter increments on the rising or falling edge of the input pulse. A filter can be enabled to filter the glitch of the input pulse.
- Each channel can generate a match trigger when the counter value matches.
- Each channel can generate an interrupt and DMA request.
- Some fields are write protected. They can only be written when the channel is disabled.
- STIM can also work in low power mode if the selected clock source is not gated off in low power mode.

31.1.2 Block diagram

The following figure is a block diagram of the System Timer (STIM) module.

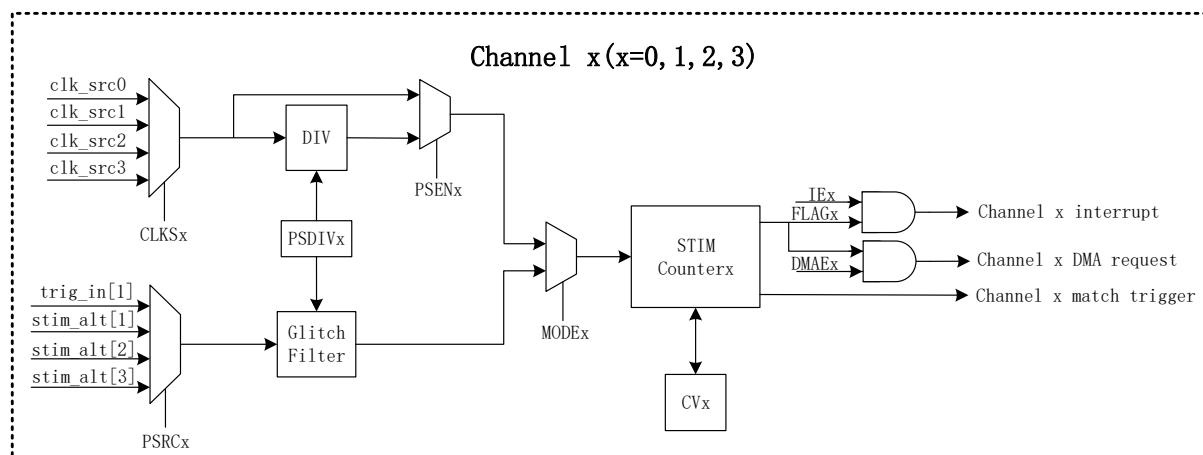


Figure 31-1 STIM block diagram

31.1.3 Instantiation and channel information

The following table lists the STIM instantiation and channel information for each chip in the product series.

Configuration	K144MC	K146MC	K148MC
Instance(s)	STIM	STIM	STIM
Channel(s)	4	4	4

31.2 Signal description

The STIM module has one I/O signal connected to the external chip pins and four channels.

31.3 Memory map and register definition

This section provides the memory map and register definitions of the STIM module.

31.3.1 Memory map

STIM base address: 0x40040000

Table 31-1 STIM Memory map

Offset	Register	Width (In bits)	Reset Value
0x04 + 4*n (n=0~3)	STIM Channel n Counter Register (STIM_CNTn)	32	0x00000000
0x14 + 4*n (n=0~3)	STIM Channel n Compare Value Register (STIM_CVn)	32	0x00000000
0x24 + 4*n (n=0~3)	STIM Channel n Status and Control Register (STIM_SCn)	32	0x00000000

31.3.2 STIM Channel n Counter Register (STIM_CNTn)

31.3.2.1 Offset

Register	Offset
STIM_CNTn	0x04 + 4*n, (n=0~3)

31.3.2.2 Function

STIM implements 4 independent channels. Every channel contains one independent counter, STIM_CNTn (n=0, 1, 2, 3) register represents the channel n counter value.

31.3.2.3 Fields

Field	Access	Description	Reset Value
31:0 CNT	RO	This field is the 32-bit counter n value. Reading this field returns current counter value.	0x0

31.3.3 STIM Channel n Compare Value Register (STIM_CVn)

31.3.3.1 Offset

Register	Offset
STIM_CVn	0x14 + 4*n, (n=0~3)

31.3.3.2 Function

STIM_CVn (n=0, 1, 2, 3) register represents the channel n compare value. When channel n counter matches with the STIM_CVn value, channel n FLAG is set.

When channel n is enabled, updating the STIM_CVn register value by software will reset the STIM_CNTn value and counter n restarts to count from 0.

31.3.3.3 Fields

Field	Access	Description	Reset Value
31:0 CV	RW	When the channel n counter value matches with its compare value, channel n FLAG is set.	0x0

31.3.4 STIM Channel n Status and Control Register (STIM_SCn)

31.3.4.1 Offset

Register	Offset
STIM_SCn	0x24 + 4*n,(n=0~3)

31.3.4.2 Function

Every channel of STIM is independently controlled, it is determined by STIM_SCn (n=0, 1, 2, 3).

STIM_SCn[11:0] fields are write protected. When channel n is enabled, writing to these fields is ignored. They can only be written when channel n is disabled.

31.3.4.3 Fields

Field	Access	Description	Reset Value
1:0 CLKS	RW	Clock source select This field selects the clock source as the counter clock source. 00b: Select function clock of STIM as the clock source. 01b: Select bus clock as the clock source. 10b: Select OSC-40M clock as the clock source. 11b: Select LPO-32K clock as the clock source.	0x0
5:2 PSDIV	RW	In the free counting mode, PSDIV is used as the clock divider factor. In the pulse counting mode, PSDIV is used as the glitch filter value. The input pulse width less than (2 ^{PSDIV} -1) STIM counter clock source period will be filtered as glitch. NOTE In the pulse counting mode, if PSEN is enabled, PSDIV can NOT be configured as 0, otherwise it will get the unpredictable result. 0000b: Divide the STIM counter clock source by 2. The	0x0

Field	Access	Description	Reset Value
		glitch filter does not support this configuration. 0001b: Divide the STIM counter clock source by 4. The pulse width less than 1 STIM system clock period will be filtered as glitch. 0010b: Divide the STIM counter clock source by 8. The pulse width less than 3 STIM system clock periods will be filtered as glitch. 0011b: Divide the STIM counter clock source by 16. The pulse width less than 7 STIM system clock periods will be filtered as glitch. 0100b: Divide the STIM counter clock source by 32. The pulse width less than 15 STIM system clock periods will be filtered as glitch. 0101b: Divide the STIM counter clock source by 64. The pulse width less than 31 STIM system clock periods will be filtered as glitch. 0110b: Divide the STIM counter clock source by 128. The pulse width less than 63 STIM system clock periods will be filtered as glitch. 0111b: Divide the STIM counter clock source by 256. The pulse width less than 127 STIM system clock periods will be filtered as glitch. 1000b: Divide the STIM counter clock source by 512. The pulse width less than 255 STIM system clock periods will be filtered as glitch. 1001b: Divide the STIM counter clock source by 1024. The pulse width less than 511 STIM system clock periods will be filtered as glitch. 1010b: Divide the STIM counter clock source by 2048. The pulse width less than 1023 STIM system clock periods will be filtered as glitch. 1011b: Divide the STIM counter clock source by 4096. The pulse width less than 2047 STIM system clock periods will be filtered as glitch. 1100b: Divide the STIM counter clock source by 8192. The pulse width less than 4095 STIM system clock periods will be filtered as glitch. 1101b: Divide the STIM counter clock source by 16384. The pulse width less than 8191 STIM system clock periods will be filtered as glitch. 1110b: Divide the STIM counter clock source by 32768. The pulse width less than 16383 STIM system clock periods will be filtered as glitch. 1111b: Divide the STIM counter clock source by 65536. The pulse width less than 32767 STIM system clock period will be filtered as glitch.	
6 PSEN	RW	Prescaler/Glitch filter enable In a free counting mode, PSEN is used to enable the prescaler. In a pulse counting mode, PSEN is used to enable the glitch filter. 0b: The prescaler or glitch filter was disabled. A counter clock source will not be divided, and the input pulse glitch cannot be filtered. 1b: The prescaler or glitch filter was enabled. A counter clock source will be divided, and the input pulse glitch	0x0

Field	Access	Description	Reset Value
		can be filtered.	
7 POL	RW	Input pin polarity This bit defines the input pin polarity in pulse counting mode. 0b: The input pin polarity is active high. The counter will increment on the rising edge of the input pulse. 1b: The input pin polarity is active low. The counter will increment on the falling edge of the input pulse.	0x0
9:8 PSRC	RW	Input pin source select This field defines which input pin source was selected as input pulse for pulse counting mode. The source of input pin 0 comes from TMU, and the source of input pin 1/2/3 is from pads. 00b: The input pin 0 is selected. 01b: The input pin 1 is selected. 10b: The input pin 2 is selected. 11b: The input pin 3 is selected.	0x0
10 MODE	RW	Timer mode select 0b: Stim channel n counter operates in free counting mode. 1b: Stim channel n counter operates in pulse counting mode.	0x0
11 RSTDIS	RW	Counter reset disable on match This bit defines whether the counter resets when the counter matches the compare value. 0b: Stim channel n counter resets and increases from 0 when counter matches. 1b: Stim channel n counter continues to increase when counter matches.	0x0
12 DMAE	RW	DMA enable This bit defines whether DMA request can be generated when the FLAG is set. 0b: Stim channel n DMA request is disabled. 1b: Stim channel n DMA request is enabled.	0x0
13 IE	RW	Interrupt enable This bit defines whether interrupt request can be generated when FLAG is set. 0b: Stim channel n interrupt request is disabled. 1b: Stim channel n interrupt request is enabled.	0x0
14 EN	RW	Stim enable 0b: Stim channel n is disabled. 1b: Stim channel n is enabled.	0x0
15 FLAG	W1C	Timer flag This flag is automatically set when the channel n counter matches the compare value. Writing 1 to this bit clears this flag. Writing 0 has no effect. This flag can also be cleared by DMA transfer done. 0b: Stim channel n counter did not match the compare value. 1b: Stim channel n counter matched the compare value.	0x0
31:16 RSVD_31_16	RO	Reserved	0x0

31.4 Function description

This section provides the functional information of the STIM module.

31.4.1 Counter clock source

For each STIM channel, one of four clock sources can be selected as STIM counter clock source. STIM_SCn[CLKS] fields define which clock is selected.

In this SoC, the four clock sources are defined as follows:

- STIM_SCn[CLKS]=00, select STIM function clock as STIM counter clock source.
- STIM_SCn[CLKS]=01, select BUS clock as STIM counter clock source.
- STIM_SCn[CLKS]=10, select OSC-40M clock as STIM counter clock source.
- STIM_SCn[CLKS]=11, select LPO-32K clock as STIM counter clock source.

STIM can still work in low power mode if the selected STIM counter clock source is not gated off in low power mode.

31.4.2 Counter mode

The STIM can work in two modes: free counting mode and pulse counting mode. The mode is selected by STIM_SCn[MODE] bit.

- When STIM_SCn[MODE] is 0, the free counting mode is selected.
 - In a free counting mode, if STIM_SCn[PSEN] is disabled, the prescaler is the bypass. Counter will increment on every rising edge of the STIM counter clock source.
 - In a free counting mode, if STIM_SCn[PSEN] is enabled, the prescaler is enabled. The STIM counter clock source will be divided by the prescaler, the divided clock is used to clock the counter. And divider factor is determined by STIM_SCn[PSDIV]. For example, when STIM_SCn[PSDIV] is 0001, the STIM counter clock source is divided by 4. When STIM_SCn[PSDIV] is 0010, the STIM counter clock source is divided by 8.
- When STIM_SCn[MODE] is 1, the pulse counting mode is selected.
 - In a pulse counting mode, if STIM_SCn[PSEN] is disabled, the glitch filter is the bypass. The counter will increment on every rising edge of input pulse when the pulse polarity is active high or increment on every falling edge of input pulse when pulse polarity is active low.
 - In pulse counting mode, if STIM_SCn[PSEN] is enabled, the glitch filter is enabled. The STIM counter clock source is used to sample the input pulse and STIM_SCn[PSDIV] fields determine the input pulse width should be stable in how many STIM counter clock source cycles. When PSDIV=n, that means input pulse width that is longer than (2^{n+1}) STIM system clock period will pass through the filter, the input pulse width that is less than (2^{n-1}) STIM counter clock source period will be filtered as glitch, the input pulse width between (2^{n+1}) and (2^{n-1}) STIM counter clock source clock period may pass through the filter or not.

NOTE

In the pulse counting mode, if the glitch filter is enabled, STIM_SCn[PSDIV] fields are not allowed to be configured as 0.

31.4.3 Counter pulse source

In pulse counter mode, for each channel, the user can select one of the four input pulse sources. Input pulse source 0 is a trigger pulse from SoC Trigger Mux Unit (TMU), input pulse 1/2/3 comes from pads. STIM_SCn[PSRC] fields define which pulse source is selected.

31.4.4 Counter match

For each channel, no matter it is in the free counting mode or pulse counting mode, the counter match occurs when the counter value reaches the STIM_CV value.

STIM_SCn[FLAG] is set on counter match, and interrupt or DMA request can be generated if interrupt enable or DMA enable is enabled.

When counter match occurred, the counter can continue to increment or reset to initial value 0, which is determined by STIM_SCn[RSTDIS] bit.

When the counter match happened, a match trigger pulse is generated. The trigger pulse width is the divided counter clock source period.

31.5 Initialization application information

Before a channel works, the counter should be initialized with the following steps:

1. Write to STIM_CV register to configure the compare value.
2. Configure STIM_SCn[RSTDIS] bit to select whether the counter resets when counter matches the compare value.
3. Configure STIM_SCn[CLKS] fields to select the counter clock source.
4. Configure STIM_SCn[MODE] bit to select the counter mode.
5. Configure input pulse source and polarity if pulse counting mode is selected.
6. Configure STIM_SCn[PSEN] bit to enable/disable the prescaler in free counting mode or the glitch filter in pulse counting mode.
7. Configure STIM_SCn[PSDIV] bit to be used as the clock divider factor in free counting mode or the glitch filter value in pulse counting mode when the STIM_SCn[PSEN] is enabled.
8. Enable STIM_SCn[IE] and STIM_SCn[DMAE] bit if we want to generate interrupt and DMA request.
9. Set STIM_SCn[EN] bit to 1 to enable counter.

Chapter 32

Real Time Counter (RTC)

32.1 Introduction

The chapter defines Real Time Counter (RTC).

32.1.1 Features

The RTC supports the following key features:

- 32-bit seconds counter to generate alarm match and seconds interrupt.
- Optional counter clock source from LPO32K.
- Register written lock protection feature. It requires unlock key as writing 0x9867 to RTC_LOCKR[31:16]. Default RTC registers could not be written after chip POR reset.
- Configurable 1Hz, 2Hz, 4Hz, 8Hz, 16Hz, 32Hz, 64Hz, 128Hz crystal square wave RTC clock out to PTC5/PTD6/PTD13. Also support configurable seconds interrupt.
- It can generate an asynchronous interrupt to wakeup chip from low power mode, and also support periodic asynchronous wakeup interrupt under STANDBY mode.

32.1.2 Block diagram

The following figure is a block diagram of the Real Time Counter (RTC) module.

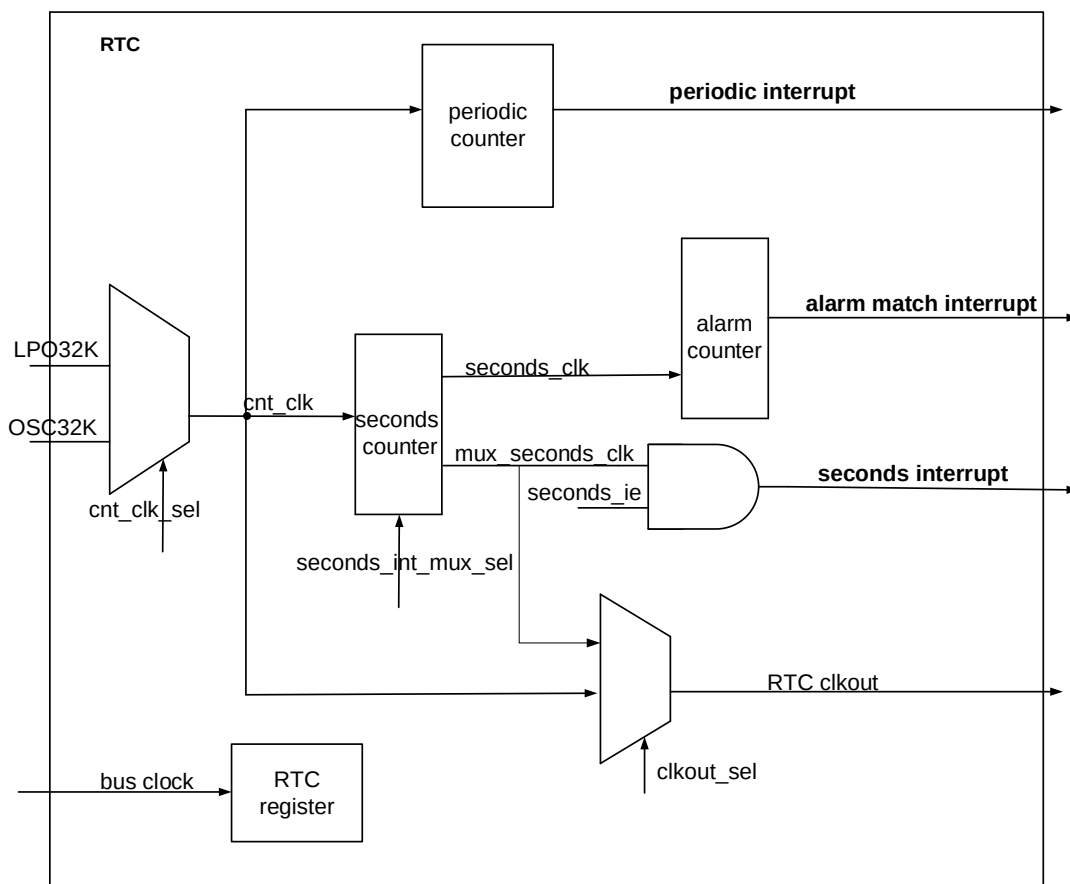


Figure 32-1 RTC block diagram

32.1.3 Instantiation information

The following table lists the RTC instantiation information for each chip in the product series.

Configuration	K144MC	K146MC	K148MC
Instance(s)	RTC	RTC	RTC

32.2 Signal description

Table 32-1 RTC signal descriptions

Signal	Description	I/O
RTC_CLKOUT	Configurable prescaler square-wave output	Output

32.3 Memory map and register definition

This section provides the memory map and register definitions of the RTC module.

In non-supervisor mode, write access is allowed if RTC_CSR[SUP_EN] is 1, and write access is forbidden if RTC_CSR[SUP_EN] is 0.

32.3.1 Memory map

RTC base address: 0x40079000

Table 32-2 RTC memory map

Offset	Register	Width (In bits)	Reset Value
0x00	RTC Control and Status Register (RTC_CSR)	32	0x00000002
0x04	RTC Interrupt Enable Register (RTC_IER)	32	0x00000000
0x08	RTC Alarm Counter Register (RTC_ALARM_CNTR)	32	0x00000000
0x0C	RTC One Second Counter Register (RTC_ONE_SECOND_CNTR)	32	0x00000000
0x10	RTC Match Counter Register (RTC_MATCH_CNTR)	32	0xFFFFFFFF
0x14	RTC Compensation Register (RTC_COMPENSATIONR)	32	0x00000000
0x18	RTC Lock Register (RTC_LOCKR)	32	0x80000000
0x20	RTC Periodic Timer Counter (RTC_PERIODIC_TMR_CNT)	32	0x00000000
0x24	RTC Periodic Timer Match Counter (RTC_PERIODIC_TMR_MATCH_CNT)	32	0xFFFFFFFF

32.3.2 RTC Control and Status Register (RTC_CSR)

32.3.2.1 Offset

Register	Offset

Register	Offset
RTC_CSR	0x00

32.3.2.2 Function

This register provides configuration of RTC and status check.

32.3.2.3 Fields

Field	Access	Description	Reset Value
0 CRYSTAL_EN	RW	32k crystal enable bit. 0b: Crystal is disabled. 1b: Crystal is enabled.	0x0
1 CNT_CLK_SEL	RW	RTC one second counter clock source select bit. It could only be valid written when RTC_CSR[COUNT_EN] = 0. 0b: Selects RTC OSC32K. 1b: Selects LPO32K.	0x1
2 CLKOUT_SEL	RW	RTC output clock selection. 0b: Selects internal seconds counter output clock. It could be impacted via SECOND_INT_MUX_SEL options. 1b: Selects RTC OSC32K or LPO32K to RTC_CLKOUT. It depends on CNT_CLK_SEL value.	0x0
3 CLKOUT_PIN_EN	RW	RTC output clock to pin output enable. 0b: Does not allow RTC_CLKOUT to pin. 1b: Allows RTC_CLKOUT to pin observe.	0x0
4 SUP_EN	RW	Supervisor mode enable bit. 0b: Non-supervisor mode write access to RTC is not allowed and generates bus error. 1b: Non-supervisor mode write access to RTC is allowed.	0x0
5 SW_RST	RW	Software reset It is suggested to write 1 to assert RTC soft reset, then write 0 to release the reset, which requires 4 counter clock cycles to complete, after which the register configuration takes effect. 0b: Reset release. 1b: Reset all RTC registers except SW_RST bit and RTC_LOCKR register.	0x0
6 COUNT_EN	RW	RTC timer counter enable. When disabled, ONE_SECOND_CNT and ALARM_CNT can be written but not increment. When enabled, ONE_SECOND_CNT and ALARM_CNT could not be written but only increment. 0b: RTC timer counter is disabled. 1b: RTC timer counter is enabled.	0x0
7 OVF	RO	RTC overflow flag. Only assert to high when 32bit ALARM_CNT overflows. ALARM_CNT will be read as zero once overflow occurs and will be cleared via writing to ALARM_CNT or	0x0

Field	Access	Description	Reset Value
		RTC_ONE_SECOND_CNTR when the RTC counter is disabled. It could also be reset via RTC soft reset and chip POR reset. 0b: No RTC overflow occurs. 1b: The RTC 32bit alarm counter overflow occurs.	
8 ALARM_MATCHF	RO	RTC alarm match flag. Only assert to high when ALARM_CNT equals ALARM_MATCH value. It is generated via the second clock. And it will be cleared via writing to RTC_MATCH_CNTR register or RTC soft reset or chip POR reset. 0b: No RTC alarm match event occurs. 1b: The RTC alarm match event occurs.	0x0
9 OSC_RDY	RO	OSC32K ready. 0b: OSC32K is not stable to use. 1b: OSC32K startup has finished and been stable to use.	0x0
10 EXTAL_32K_EN	RW	32K external square wave clock enable bit. 1b: Enable extal 32k. 0b: Disable extal 32k.	0x0
11 SECONDS_INT_FLAG	W1C	Seconds interrupt flag. It will be asserted to 1 at rising edge of 1hz clock and will be cleared via w1c to this bit. When disabling the RTC_IER[SECONDS_IE] = 0, this seconds_int_flag will be held as low. POR reset or RTC software reset also can clear this bit.	0x0
12 PERIODIC_CNT_EN	RW	RTC periodic counter enable. Only when this bit and RTC_CSR[COUNT_EN] are both set to 1 can periodic counter start increasing.	0x0
13 PERIODIC_MATCH_FLG	W1C	RTC periodic counter match flag When RTC periodic counter increases up to the periodic match value, then this periodic_match_flag asserts to high. The periodic counter increases based on 32K counter clock when both RTC_CSR[PERIODIC_CNT_EN] and RTC_CSR[COUNT_EN] are set to 1. It's used for STANDBY mode only, like periodically generating interrupt every 1ms, 10ms, or 100ms, etc. It can be cleared via w1c option or writing RTC_PERIODIC_TMR_MATCH_CNT register. RTC soft reset or chip POR reset also can clear this flag.	0x0
31:14 RSVD_31_14	RO	Reserved	0x0

32.3.3 RTC Interrupt Enable Register (RTC_IER)

32.3.3.1 Offset

Register	Offset

Register	Offset
RTC_IER	0x04

32.3.3.2 Function

This register can enable or disable interrupts and select second interrupt frequency.

32.3.3.3 Fields

Field	Access	Description	Reset Value
2:0 SECOND_INT_MUX_SEL	RW	Seconds interrupt mux selection It selects different frequency to generate the seconds interrupt. It can only be written when SECONDS_IE=0. It will also impact the RTC clock out to pin selections. 111b: 128Hz 110b: 2Hz 101b: 4Hz 100b: 8Hz 011b: 16Hz 010b: 32Hz 001b: 64Hz 000b: 1Hz	0x0
3 OVFLOW_IE	RW	RTC overflow interrupt enable 1b: Enable overflow interrupt. 0b: Disable overflow interrupt.	0x0
4 ALARM_IE	RW	RTC timer alarm match interrupt enable 1b: Enable alarm match interrupt. 0b: Disable alarm match interrupt.	0x0
5 SECONDS_IE	RW	Seconds interrupt enable bit The seconds interrupt is an edge-sensitive interrupt and generated once a second. There is no any second flag. 1b: Enable seconds interrupt. 0b: Disable seconds interrupt.	0x0
6 PERIODIC_IE	RW	Periodic interrupt enable bit If RTC_CSR[PERIODIC_MATCH_FLG] asserts to 1 when PERIODIC_IE=1, then related periodic interrupt will occur.	0x0
31:7 RSVD_31_7	RO	Reserved	0x0

32.3.4 RTC Alarm Counter Register (RTC_ALARM_CNTR)

32.3.4.1 Offset

Register	Offset
RTC_ALARM_CNTR	0x08

32.3.4.2 Function

This register specifies alarm counter.

32.3.4.3 Fields

Field	Access	Description	Reset Value
31:0 ALARM_CNT	RW	Alarm counter value When RTC timer is enabled, it is read-only. And it increments once a second if the RTC_CSR[OVF] is not set. It will be read as 0 when OVF is set. When timer is disabled, it can be written or read. Writing any value to RTC_ALARM_CNTR when the time counter is disabled will clear the RTC_CSR[OVF]. It is not recommended to write 0x0 to this register when timer is disabled, considering it will be read as 0 when RTC_CSR[OVF] is set.	0x0

32.3.5 RTC One Second Counter Register (RTC_ONE_SECOND_CNTR)

32.3.5.1 Offset

Register	Offset
RTC_ONE_SECOND_CNTR	0x0c

32.3.5.2 Function

This register is one second counter.

32.3.5.3 Fields

Field	Access	Description	Reset Value
15:0 ONE_SECOND_CNT	RW	One seconds counter value When RTC timer is enabled, it is read-only, and increments once a 32.768 kHz cycle. And it will be read as 0 when RTC_CSR[OVF] is set. RTC_ALARM_CNTR will increase when bit 14 of RTC_ONE_SECOND_CNTR changes from logic one to logic zero.	0x0
31:16 RSVD_31_16	RO	Reserved	0x0

32.3.6 RTC Match Counter Register (RTC_MATCH_CNTR)

32.3.6.1 Offset

Register	Offset
RTC_MATCH_CNTR	0x10

32.3.6.2 Function

This register can be configured match value for alarm interrupt.

32.3.6.3 Fields

Field	Access	Description	Reset Value
31:0 MATCH_CNT	RW	When the RTC timer counter is enabled, RTC_CSR[ALARMF] will be set when RTC_ALARM_CNTR equals RTC_MATCH_CNTR. Any write to RTC_MATCH_CNTR will clear RTC_CSR[ALARMF].	0xFFFFFFFF

32.3.7 RTC Compensation Register (RTC_COMPENSATIONR)

32.3.7.1 Offset

Register	Offset
RTC_COMPENSATIONR	0x14

32.3.7.2 Function

This register provides compensation configuration.

32.3.7.3 Fields

Field	Access	Description	Reset Value
6:0 COMP_CNT	RW	Compensation counter value. The written COMP_CNT will take effect at the end of DLY. If COMP_DIRECTION=1, the second frequency needs to go down. 00h: One second counter overflows every 32896 clock cycles. 01h: One second counter overflows every 32895 clock cycles. 7Eh: One second counter overflows every 32770 clock cycles. 7Fh: One second counter overflows every 32769 clock cycles. If COMP_DIRECTION=0, the second frequency needs to up. 00h: One second counter overflows every 32768 clock cycles. 01h: One second counter overflows every 32767 clock cycles. 7Fh: One second counter overflows every 32641 clock cycles.	0x0
7 COMP_DIRECTION	RW	Frequency compensation direction 1b: When the compensation direction goes down, the second frequency goes slower. 0b: When the compensation direction goes up, the second frequency goes faster.	0x0
15:8 DLY	RW	Compensation delay value It can be configured from 1 to 255 to control how frequently RTC adjusts its number of 32.768 kHz in each second. The written COMP_CNT will	0x0

Field	Access	Description	Reset Value
		take effect when DLY decreases to 0.	
23:16 CCOMP_CNT	RO	Current compensation counter value It will be updated once a second. If CDLY=0, it will be loaded from COMP_CNT, otherwise it will be loaded 0.	0x0
31:24 CDLY	RO	Current compensation delay value If CDLY=0, it will be loaded from DLY, and if CDLY != 0, it will decrease once a second.	0x0

32.3.8 RTC Lock Register (RTC_LOCKR)

32.3.8.1 Offset

Register	Offset
RTC_LOCKR	0x18

32.3.8.2 Function

This is RTC lock register.

32.3.8.3 Fields

Field	Access	Description	Reset Value
30:0 RSVD_30_0	RO	Reserved	0x0
31 LOCK	RW	RTC_LOCKR lock bit It will block write option of all other RTC registers when LOCK=1. It can be only cleared via writing 0x9867 to RTC_LOCKR[31:16]. After being cleared, it could be asserted to high via POR/LVD reset. 0b: Allow RTC_LOCKR, and the normal written option. 1b: RTC_LOCKR is locked, and the written option is ignored.	0x1

32.3.9 RTC Periodic Timer Counter (RTC_PERIODIC_TMR_CNT)

32.3.9.1 Offset

Register	Offset
RTC_PERIODIC_TMR_CNT	0x20

32.3.9.2 Function

This is the periodic timer counter.

32.3.9.3 Fields

Field	Access	Description	Reset Value
31:0 PERIODIC_TMR_CNT	RW	Periodic timer counter value. It is read-only when RTC_CSR[COUNT_EN] and RTC_CSR[PERIODIC_CNT_EN] are both enabled. It can also be written when RTC_CSR[COUNT_EN]=0. The written value should be $\leq$ periodic_match_cnt-2.	0x0

32.3.10 RTC Periodic Timer Match Counter (RTC_PERIODIC_TMR_MATCH_CNT)

32.3.10.1 Offset

Register	Offset
RTC_PERIODIC_TMR_MATCH_CNT	0x24

32.3.10.2 Function

This is the periodic match counter.

32.3.10.3 Fields

Field	Access	Description	Reset Value
31:0 PERIODIC_MATCH_CNT	RW	Periodic match counter value. The written match value can be from 0x02 to 0xffffffff.	0xFFFFFFFF

32.4 Function description

This section provides the functional information of the RTC module.

RTC is put on an always-on power domain, and it can work on all low power modes. The power-on reset can initialize all RTC registers to default reset state. RTC soft reset can also initialize RTC registers except its LOCK register.

32.4.1 Oscillator control

The OSC32K (32.768kHz) oscillator is disabled after system reset and must enable OSC32K before enabling RTC_CSR[COUNT_EN]. After enabling OSC32K, it will assert RTC_CSR[OSC_RDY] flag to indicate OSC32K is stable after its startup time (1s).

When setting RTC_CSR[EXTAL_32K_EN] = 1, crystal oscillator will be bypassed, and RTC_CSR[OSC_RDY] flag will be asserted high after 2 synchronous external 32k cycles.

32.4.2 Timer counter

RTC includes a 32-bit counter that increases once every 32.768 kHz cycle. There is also select bit to choose LPO32K as counter clock source.

RTC_ALARM_CNTR and RTC_ONE_SECOND_CNTR registers could only be written when RTC counter is disabled. When RTC 32bit alarm counter overflows, overflow flag can be cleared via writing any value to RTC_ALARM_CNTR or RTC_ONE_SECOND_CNTR when counter is disabled.

When RTC counter is enabled, above two counter registers are only readable.

32.4.3 Alarm match

When RTC_CSR[ALARM_MATCHF] asserts to high, and RTC_CSR[ALARM_IE] is set to 1, RTC will generate an alarm match interrupt. And this flag can be cleared via writing any value to RTC_MATCH_CNTR register. If RTC_ALARM_CNTR increases up to RTC_MATCH_CNTR values, RTC will assert its RTC_CSR[ALARM_MATCHF] flag.

32.4.4 Periodic interrupt

Under STANDBY mode, RTC also has high resolution asynchronous wakeup interrupt to support periodic interrupt. The periodic interrupt is used for periodic wakeup under STANDBY mode.

32.4.5 Register lock

RTC_LOCKR register is used to block writing to RTC registers until next unlock sequence. Default LOCK bit is high.

Write to RTC registers when RTC_LOCKR[LOCK]=1 will be ignored and will not cause a bus error.

After chip POR reset, unlock sequence is required via writing 0x9867 to RTC_LOCKR[31:16]. After polling RTC_LOCKR[LOCK]=0, the user can begin configuring RTC registers.

If RTC_LOCKR[LOCK]=0, the user can re-lock the written option of RTC registers via write RTC_LOCKR[31]=1. It can also again block any written option to RTC registers.

32.4.6 Compensation

RTC compensation logic updates the number of 32.768 kHz clock cycles needed for RTC_ONE_SECOND_CNT overflow. It adjusts the number of cycles between -127 to 128.

RTC_ONE_SECOND_CNT will add or subtract a configurable number when its value is equal to 0x3FFF and then update seconds counter when the compensation interval decreases to 0. The compensation interval values can be configured from once one second to once 256 seconds.

If the RTC_COMPENSATIONR[COMP_DIRECTION] =0, it will decrease the RTC_ONE_SECOND_CNT and compensation towards faster frequency. Otherwise, it will increase RTC_ONE_SECOND_CNT and compensation towards slower frequency.

Updating RTC_COMPENSATIONR registers will not take effect until next time seconds counter increases and the compensation delay interval has expired.

Compensation will be disabled via writing 0x00000000 to RTC_COMPENSATIONR.

32.5 Initialization application information

32.5.1 Alarm match interrupt setup via OSC32K

1. Unlock RTC_LOCKR register via writing 0x9867 to RTC_LOCKR[31:16].
2. Poll RTC_LOCKR[LOCK]=0.
3. Write match value like 0x10 to RTC_MATCH_CNTR.
4. Write 0x10 to RTC_IER, enable alarm match interrupt enable bit.
5. Write 0x00000001 to RTC_CSR, enable OSC32K and poll RTC_CSR[OSC_RDY]=1.
6. Write 0x00000041 to RTC_CSR, enable RTC counter.
7. Poll RTC_CSR[ALARM_MATCHF]=1, then check alarm match interrupt.

32.5.2 Alarm match interrupt setup via LPO32K

1. Unlock RTC_LOCKR register via writing 0x9867 to RTC_LOCK[31:16].
2. Poll RTC_LOCKR[LOCK]=0.
3. Write match value like 0x10 to RTC_MATCH_CNTR.
4. Set RTC_IER=0x10, enable alarm match interrupt enable bit.
5. Write 0x00000002 to RTC_CSR, select LPO32K as counter clock source.
6. Write 0x00000042 to RTC_CSR, enable RTC counter.
7. Poll RTC_CSR[ALARM_MATCHF]=1, then check alarm match interrupt.

32.5.3 Seconds interrupt setup via OSC32K

1. Unlock RTC_LOCKR register via writing 0x9867 to RTC_LOCKR[31:16].
2. Poll RTC_LOCKR[LOCK]=0.
3. Set RTC_IER=0x20, enable RTC seconds interrupt enable bit.
4. Write 0x00000001 to RTC_CSR, select OSC32K as counter clock source and poll RTC_CSR[OSC_RDY]=1.
5. Write 0x00000041 to RTC_CSR, enable RTC counter.
6. Check RTC seconds interrupt will occur once.

Chapter 33

Trigger Delay Generator (TDG)

33.1 Introduction

This chapter defines Trigger Delay Generator (TDG).

33.1.1 Features

The TDG supports the following key features:

- One external hardware trigger source and one software trigger source
- Up to 6 configurable channels and up to 8 programmable delayed outputs on each channel
- Delayed outputs can be enabled or disabled independently
- Programmable delay interrupt
- One sequence error interrupt

33.1.2 Block diagram

The following figure is a block diagram of the Trigger Delay Generator (TDG) module.

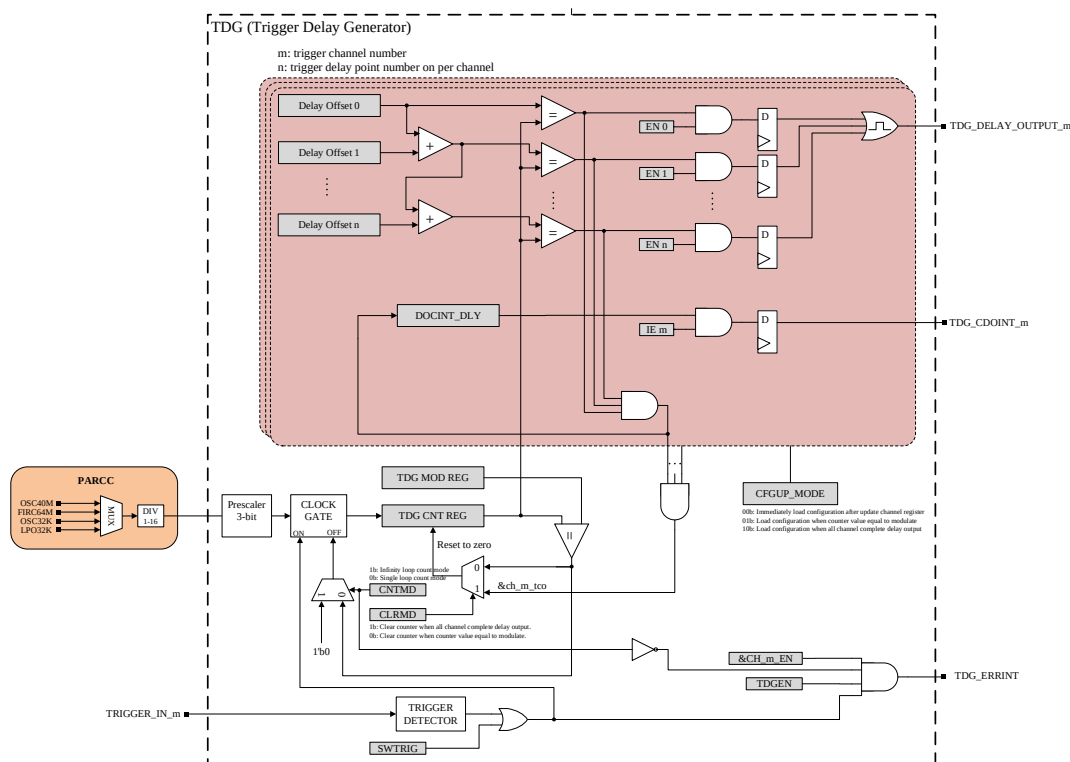


Figure 33-1 TDG block diagram

33.1.3 Instantiation information

The following table lists the TDG instantiation information for each chip in the product series.

Configuration	K144MC	K146MC	K148MC
Instance(s)	TDG0	TDG0	TDG0
	TDG1	TDG1	TDG1

33.2 Memory map and register definition

This section provides the memory map and register definitions of the TDG module.

33.2.1 Memory map

TDG0 base address: 0x4000F000

TDG1 base address: 0x4001B000

Table 33-1 TDG memory map

Offset	Register	Width (In bits)	Reset Value
0x4	TDG Parameter (TDG_PARAM)	32	0x00000068
0x8	TDG Control Register 1 (TDG_CTRL1)	32	0x00000000
0xC	TDG Control Register 2 (TDG_CTRL2)	32	0x00000000
0x10	TDG Status Register (TDG_STAT)	32	0x00000000
0x14	TDG Modulate Register (TDG_MOD)	32	0x0000FFFF
0x18	TDG Counter Register (TDG_CNT)	32	0x00000000
0x1C+n*0x28, n=(0~5)	TDG Channel n Control Register (TDG_CHnCTRL)	32	0x00000000
0x20+n*0x28, n=(0~5)	TDG Channel n Delay Output 0 Offset Register (TDG_CHnDO0OFS)	32	0x0000FFFF
0x24+n*0x28, n=(0~5)	TDG Channel n Delay Output 1 Offset Register (TDG_CHnDO1OFS)	32	0x0000FFFF
0x28+n*0x28, n=(0~5)	TDG Channel n Delay Output 2 Offset Register (TDG_CHnDO2OFS)	32	0x0000FFFF
0x2C+n*0x28, n=(0~5)	TDG Channel n Delay Output 3 Offset Register (TDG_CHnDO3OFS)	32	0x0000FFFF
0x30+n*0x28, n=(0~5)	TDG Channel n Delay Output 4 Offset Register (TDG_CHnDO4OFS)	32	0x0000FFFF
0x34+n*0x28, n=(0~5)	TDG Channel n Delay Output 5 Offset Register (TDG_CHnDO5OFS)	32	0x0000FFFF
0x38+n*0x28, n=(0~5)	TDG Channel n Delay Output 6 Offset Register (TDG_CHnDO6OFS)	32	0x0000FFFF
0x3C+n*0x28, n=(0~5)	TDG Channel n Delay Output 7 Offset Register (TDG_CHnDO7OFS)	32	0x0000FFFF
0x40+n*0x28,	TDG Channel n Complete Delay Output	32	0x00010000

Offset	Register	Width (In bits)	Reset Value
n=(0~5)	Interrupt Delay Register (TDG_CHnCDOINTDLY)		

33.2.2 TDG Parameter (TDG_PARAM)

33.2.2.1 Offset

Register	Offset
TDG_PARAM	0x4

33.2.2.2 Function

This register is for indicating the design feature implementation.

33.2.2.3 Fields

Field	Access	Function	Reset Value
3:0 DOPNUM	RO	Delay output point number Number of delay output point on per channel.	0x8
7:4 CHNUM	RO	Channel number Number of implemented channels.	0x6
31:8 RSVD_31_8	RO	Reserved	0x0

33.2.3 TDG Control Register 1 (TDG_CTRL1)

33.2.3.1 Offset

Register	Offset
TDG_CTRL1	0x8

33.2.3.2 Function

This register is for the TDG control configuration.

33.2.3.3 Fields

Field	Access	Function	Reset Value
0 TDGEN	RW	TDG enabled This bit enables the TDG. 0b: TDG disabled. 1b: TDG enabled.	0x0
3:1 RSVD_3_1	RO	Reserved	0x0
4 TRIGS	RW	Trigger source select This bit selects the trigger source. 0b: External trigger source. 1b: Software trigger source.	0x0
5	RW	Count mode	0x0

Field	Access	Function	Reset Value
CNTMD		This bit selects count mode. 0b: Single loop mode. 1b: Infinity loop mode.	
6 CLRMD	RW	Counter clear mode This bit selects counter clear mode. 0b: Clear counter when counter is equal to modulator. 1b: Clear counter when the channel with the maximum accumulated delay output offset completes its delay output. Note: The accumulated delay output offset is the sum of the channel delay outputs 0-7, regardless of whether the delay outputs are enabled.	0x0
7 RSVD_7	RO	Reserved	0x0
10:8 PRES	RW	Prescaler These bits are used to prescale the count clock source. 000b: DIV1 001b: DIV2 010b: DIV4 011b: DIV8 100b: DIV16 101b: DIV32 110b: DIV64 111b: DIV128	0x0
15:11 RSVD_15_11	RO	Reserved	0x0
16 CFGUP	RW	Configuration update Set this bit to load channel configuration. This bit will be cleared automatically by hardware when the load is complete.	0x0
17 SWTRG	WO	Software trigger Write only. Read always returns 0. Trigger software trigger by writing 1.	0x0
19:18 RSVD_19_18	RO	Reserved	0x0
21:20 UPMD	RW	Channel configuration update mode 00b: Immediately load configuration after update channel register. 01b: Load configuration when counter counts up to be equal to modulator. 10b: Load configuration when all channels complete delay output. 11b: Reserved	0x0
31:22 RSVD_31_22	RO	Reserved	0x0

33.2.4 TDG Control Register 2 (TDG_CTRL2)

33.2.4.1 Offset

Register	Offset
TDG_CTRL2	0xC

33.2.4.2 Function

This register is for the TDG configuration.

33.2.4.3 Fields

Field	Access	Function	Reset Value
0 CDO0IE	RW	Channel 0 complete delay output interrupt enable 0b: Disable 1b: Enable	0x0
1 CDO1IE	RW	Channel 1 complete delay output interrupt enable 0b: Disable 1b: Enable	0x0
2 CDO2IE	RW	Channel 2 complete delay output interrupt enable 0b: Disable 1b: Enable	0x0
3 CDO3IE	RW	Channel 3 complete delay output interrupt enable 0b: Disable 1b: Enable	0x0
4 CDO4IE	RW	Channel 4 complete delay output interrupt enable 0b: Disable 1b: Enable	0x0
5 CDO5IE	RW	Channel 5 complete delay output interrupt enable 0b: Disable 1b: Enable	0x0
7:6 RSVD_7_6	RO	Reserved	0x0
8 ERRIE	RW	Error interrupt enable New trigger arrived causes error interrupt before any channel complete output delay.	0x0
15:9 RSVD_15_9	RO	Reserved	0x0
16 CH0E	RW	Channel 0 enable 0b: Disable 1b: Enable	0x0
17 CH1E	RW	Channel 1 enable 0b: Disable 1b: Enable	0x0
18 CH2E	RW	Channel 2 enable 0b: Disable 1b: Enable	0x0
19 CH3E	RW	Channel 3 enable 0b: Disable 1b: Enable	0x0
20 CH4E	RW	Channel 4 enable 0b: Disable 1b: Enable	0x0
21 CH5E	RW	Channel 5 enable 0b: Disable 1b: Enable	0x0
31:22 RSVD_31_22	RO	Reserved	0x0

33.2.5 TDG Status Register (TDG_STAT)

33.2.5.1 Offset

Register	Offset
TDG_STAT	0x10

33.2.5.2 Function

This register is for the interrupt status indication.

33.2.5.3 Fields

Field	Access	Function	Reset Value
0 CDO0IF	W1C	Channel 0 complete delay output interrupt flag 0b: Channel 0 complete delay output interrupt does not occur. 1b: Channel 0 complete delay output interrupt occurs. Write 1 to clear.	0x0
1 CDO1IF	W1C	Channel 1 complete delay output interrupt flag 0b: Channel 1 complete delay output interrupt does not occur. 1b: Channel 1 complete delay output interrupt occurs. Write 1 to clear.	0x0
2 CDO2IF	W1C	Channel 2 complete delay output interrupt flag 0b: Channel 2 complete delay output interrupt does not occur. 1b: Channel 2 complete delay output interrupt occurs. Write 1 to clear.	0x0
3 CDO3IF	W1C	Channel 3 complete delay output interrupt flag 0b: Channel 3 complete delay output interrupt does not occur. 1b: Channel 3 complete delay output interrupt occurs. Write 1 to clear.	0x0
4 CDO4IF	W1C	Channel 4 complete delay output interrupt flag 0b: Channel 4 complete delay output interrupt does not occur. 1b: Channel 4 complete delay output interrupt occurs. Write 1 to clear.	0x0
5 CDO5IF	W1C	Channel 5 complete delay output interrupt flag 0b: Channel 5 complete delay output interrupt does not occur. 1b: Channel 5 complete delay output interrupt occurs. Write 1 to clear.	0x0
7:6 RSVD_7_6	RO	Reserved	0x0
8 ERRIF	W1C	Error interrupt flag New trigger arrived causes error interrupt before any channel complete output delay. 0b: Error interrupt does not occur. 1b: Error interrupt occurs. Write 1 to clear.	0x0
31:9 RSVD_31_9	RO	Reserved	0x0

33.2.6 TDG Modulate Register (TDG_MOD)

33.2.6.1 Offset

Register	Offset
TDG_MOD	0x14

33.2.6.2 Function

This register is for the modulate value configuration.

33.2.6.3 Fields

Field	Access	Function	Reset Value
15:0 MOD	RW	TDG modulate value. The value must be [0x0001, 0xFFFF].	0xFFFF
31:16 RSVD_31_16	RO	Reserved	0x0

33.2.7 TDG Counter Register (TDG_CNT)

33.2.7.1 Offset

Register	Offset
TDG_CNT	0x18

33.2.7.2 Function

This register is a counter register.

33.2.7.3 Fields

Field	Access	Function	Reset Value
15:0 CNT	RO	TDG counter value. This register provides read to the value of the free-running TDG counter.	0x0
31:16 RSVD_31_16	RO	Reserved	0x0

33.2.8 TDG Channel n Control Register (TDG_CHnCTRL)

33.2.8.1 Offset

Register	Offset
TDG_CHnCTRL	0x1C + n*0x28, (n=0,1,2,3,4,5)

33.2.8.2 Function

This register is for the channel configuration.

33.2.8.3 Fields

Field	Access	Function	Reset Value
7:0 RSVD_7_0	RO	Reserved	0x0
8 DO0E	RW	Delay output 0 enable on channel n 0b: Delay output 0 disable. 1b: Delay output 0 enable.	0x0
9 DO1E	RW	Delay output 1 enable on channel n 0b: Delay output 1 disable. 1b: Delay output 1 enable.	0x0
10 DO2E	RW	Delay output 2 enable on channel n 0b: Delay output 2 disable. 1b: Delay output 2 enable.	0x0
11 DO3E	RW	Delay output 3 enable on channel n 0b: Delay output 3 disable. 1b: Delay output 3 enable.	0x0
12 DO4E	RW	Delay output 4 enable on channel n 0b: Delay output 4 disable. 1b: Delay output 4 enable.	0x0
13 DO5E	RW	Delay output 5 enable on channel n 0b: Delay output 5 disable. 1b: Delay output 5 enable.	0x0
14 DO6E	RW	Delay output 6 enable on channel n 0b: Delay output 6 disable. 1b: Delay output 6 enable.	0x0
15 DO7E	RW	Delay output 7 enable on channel n 0b: Delay output 7 disable. 1b: Delay output 7 enable.	0x0
31:16 RSVD_31_16	RO	Reserved	0x0

33.2.9 TDG Channel n Delay Output 0 Offset Register (TDG_CHnDO0OFS)

33.2.9.1 Offset

Register	Offset
TDG_CHnDO0OFS	0x20 + n*0x28, (n=0,1,2,3,4,5)

33.2.9.2 Function

This register is for the channel delay output 0 offset configuration.

33.2.9.3 Fields

Field	Access	Function	Reset Value
15:0 DO0OFS	RW	TDG Delay Output 0 Offset	0xFFFF
31:16 RSVD_31_16	RO	Reserved	0x0

33.2.10 TDG Channel n Delay Output 1 Offset Register (TDG_CHnDO1OFS)

33.2.10.1 Offset

Register	Offset
TDG_CHnDO1OFS	$0x24 + n \times 0x28$, (n=0,1,2,3,4,5)

33.2.10.2 Function

This register is for the channel delay output 1 offset configuration.

33.2.10.3 Fields

Field	Access	Function	Reset Value
15:0 DO1OFS	RW	TDG Delay Output 1 Offset	0xFFFF
31:16 RSVD_31_16	RO	Reserved	0x0

33.2.11 TDG Channel n Delay Output 2 Offset Register (TDG_CHnDO2OFS)

33.2.11.1 Offset

Register	Offset
TDG_CHnDO2OFS	$0x28 + n \times 0x28$, (n=0,1,2,3,4,5)

33.2.11.2 Function

This register is for the channel delay output 2 offset configuration.

33.2.11.3 Fields

Field	Access	Function	Reset Value
15:0 DO2OFS	RW	TDG Delay Output 2 Offset	0xFFFF
31:16 RSVD_31_16	RO	Reserved	0x0

33.2.12 TDG Channel n Delay Output 3 Offset Register (TDG_CHnDO3OFS)

33.2.12.1 Offset

Register	Offset
TDG_CHnDO3OFS	$0x2C + n \times 0x28$, (n=0,1,2,3,4,5)

33.2.12.2 Function

This register is for the channel delay output 3 offset configuration.

33.2.12.3 Fields

Field	Access	Function	Reset Value
15:0 DO3OFS	RW	TDG Delay Output 3 Offset	0xFFFF
31:16 RSVD_31_16	RO	Reserved	0x0

33.2.13 TDG Channel n Delay Output 4 Offset Register (TDG_CHnDO4OFS)

33.2.13.1 Offset

Register	Offset
TDG_CHnDO4OFS	$0x30 + n \times 0x28$, (n=0,1,2,3,4,5)

33.2.13.2 Function

This register is for the channel delay output 4 offset configuration.

33.2.13.3 Fields

Field	Access	Function	Reset Value
15:0 DO4OFS	RW	TDG Delay Output 4 Offset	0xFFFF
31:16 RSVD_31_16	RO	Reserved	0x0

33.2.14 TDG Channel n Delay Output 5 Offset Register (TDG_CHnDO5OFS)

33.2.14.1 Offset

Register	Offset
TDG_CHnDO5OFS	$0x34 + n \times 0x28$, (n=0,1,2,3,4,5)

33.2.14.2 Function

This register is for the channel delay output 5 offset configuration.

33.2.14.3 Fields

Field	Access	Function	Reset Value
15:0 DO5OFS	RW	TDG Delay Output 5 Offset	0xFFFF

Field	Access	Function	Reset Value
31:16 RSVD_31_16	RO	Reserved	0x0

33.2.15 TDG Channel n Delay Output 6 Offset Register (TDG_CHnDO6OFS)

33.2.15.1 Offset

Register	Offset
TDG_CHnDO6OFS	$0x38 + n \times 0x28$, (n=0,1,2,3,4,5)

33.2.15.2 Function

This register is for the channel delay output 6 offset configuration.

33.2.15.3 Fields

Field	Access	Function	Reset Value
15:0 DO6OFS	RW	TDG Delay Output 6 Offset	0xFFFF
31:16 RSVD_31_16	RO	Reserved	0x0

33.2.16 TDG Channel n Delay Output 7 Offset Register (TDG_CHnDO7OFS)

33.2.16.1 Offset

Register	Offset
TDG_CHnDO7OFS	$0x3C + n \times 0x28$, (n=0,1,2,3,4,5)

33.2.16.2 Function

This register is for the channel delay output 7 offset configuration.

33.2.16.3 Fields

Field	Access	Function	Reset Value
15:0 DO7OFS	RW	TDG Delay Output 7 Offset	0xFFFF
31:16 RSVD_31_16	RO	Reserved	0x0

33.2.17 TDG Channel n Complete Delay Output Interrupt Delay Register (TDG_CHnCDOINTDLY)

33.2.17.1 Offset

Register	Offset
TDG_CHnCDOINTDLY	0x40 + n*0x28, (n=0,1,2,3,4,5)

33.2.17.2 Function

This register is for the channel complete delay output interrupt configuration.

33.2.17.3 Fields

Field	Access	Function	Reset Value
9:0 CDOINTDLY	RW	TDG Complete Delay Output Interrupt Delay TDG Delay Output Complete Interrupt Delay. Writing this field triggers loading event. If LDCO is logic 1, it means loading is completed. The loading of this field is unaffected by writing 1 to TDG_CTRL1[CFGUP] bit.	0x0
15:10 RSVD_15_10	RO	Reserved	0x0
16 LDCO	RO	Complete Delay Output Interrupt Delay Configuration Load Complete This bit is read-only. It indicates if delay output complete interrupt delay configuration loading is completed. This bit will be cleared automatically when write CDOINTDLY, and it will be set automatically when CDOINTDLY shadow buffer gets loaded. 0b: Loading is on-going. 1b: Loading is completed.	0x1
31:17 RSVD_31_17	RO	Reserved	0x0

33.3 Function description

This section provides the functional information of the TDG module.

33.3.1 Input trigger

The TDG has two types of triggers: external hardware trigger and software trigger. The user can select external trigger as input source via TMU module and select software trigger via writing 1 to TDG_CTRL1[SWTRG].

33.3.2 Counter

The TDG counter can count with two modes: single loop mode and infinity loop mode. The counting mode is selected by TDG_CTRL1[CNTMD] bit.

The TDG counter can clear the free-running counter when counter equals modulator or all channels complete delay output by configuring TDG_CTRL1[CLRMD].

If the single loop mode is selected by setting TDG_CTRL1[CNTMD] to 0, the free-running TDG counter will just work for one time. It will stop counting when the counter reaches the modulator or all channels complete delay output, which depends on the clear mode.

If the infinity loop mode is selected by setting TDG_CTRL1[CNTMD] to 1, the TDG counter will reset the counter to zero and restart counting when it reaches the modulator value or all channels complete delay output.

33.3.3 Delayed output

The TDG has 6 channels and each channel has 8 delay output points. The user can configure offset for each channel. When the counter reaches the offset and the corresponding enable bit is written to 1, the TDG will generate a trigger. For a channel, all offset values are cumulative.

NOTE

Please ensure that the cumulative delay output offset does not exceed the overflow limit. If the accumulated value surpasses 0xFFFF, only the lower 16 bits will be considered. It is recommended to set the disabled offset values to 0.

33.3.4 Interrupt

The TDG module has two interrupts: channel interrupt and error interrupt.

Channel interrupt: When all its delay outputs are completed and the enable bit is written to be 1, it will start to count with function clock. When the counter reaches the interrupt delay, it will generate channel interrupt.

Error interrupt: The arrival of new trigger will cause error interrupt before any channel completes output delay.

Chapter 34

Analog-to-digital converter (ADC)

34.1 Introduction

The purpose of this chapter is to define Analog-to-digital converter (ADC).

The ADC is designed for converting the analog signal to digital signal on SOC with the linear successive approximation algorithm and up to 12 bits resolution. The max sample rate is 1.0 Mps. The ADC supports interrupt or DMA request after operation.

34.1.1 Features

The ADC supports the following key features:

- Linear successive approximation algorithm with up to 12-bit resolution
- Resolution can be selected to 8 bits, 10 bits, 12 bits
- Up to 3 pairs of differential and 20 single-ended external analog inputs
- Internal channels for FUSA usage
- Configurable sample time and conversion speed
- Single or continuous conversion mode
- Automatic compare with interrupt for less-than, greater-than or equal-to, within range, or out-of-range, programmable value
- 16 depth FIFO for conversion output
- Hardware trigger and software trigger to trigger the conversion
- Voltage reference can be external or tie to fixed voltage
- Up to 6 input triggers for loop trigger mode and map trigger mode
- Interrupt or DMA request supported
- Async interrupt supported in low power mode
- Hardware average supported

34.1.2 Block diagram

The following figure is a block diagram of the Analog-to-digital converter (ADC) module.

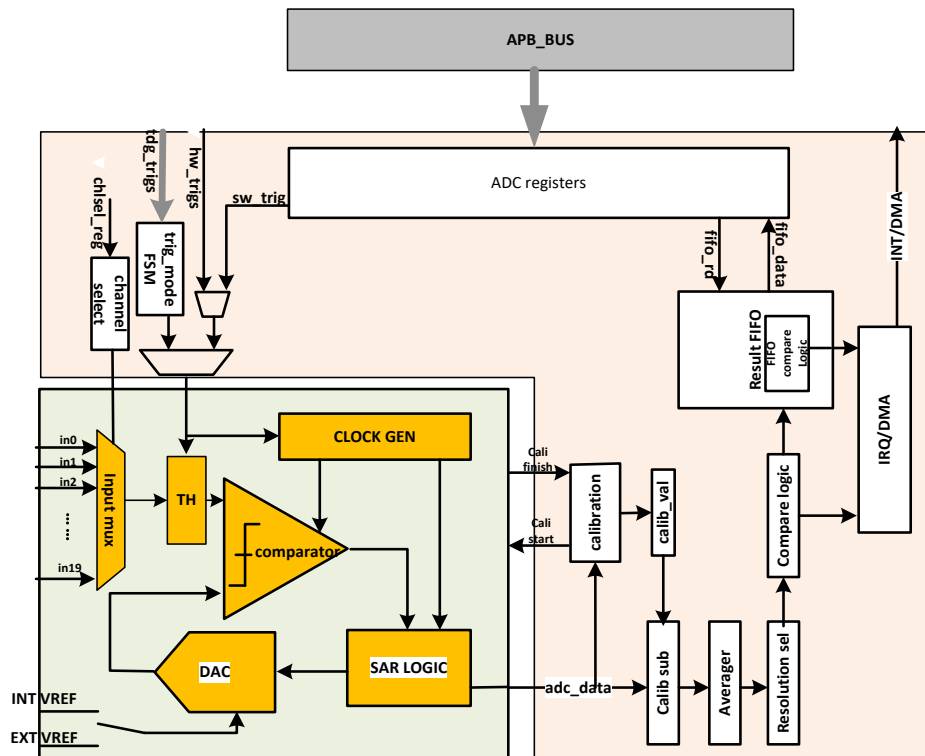


Figure 34-1 ADC block diagram

34.1.3 Instantiation and channel information

The following table lists the ADC instantiation and channel information for each chip in the product series.

Configuration	K144MC	K146MC	K148MC
Instance(s)	ADC0	ADC0	ADC0
	ADC1	ADC1	ADC1
Channel(s)	16	20	20

34.2 Signal description

The ADC module supports up to 20 single-ended inputs.

The ADC also requires 3 pairs of differential inputs.

The ADC does not produce any output signals.

Table 34-1 ADC input signal descriptions

Signal	Description
ADC_Positive_ChannelIn	Single-Ended Analog Channel Inputs Differential Analog Positive Channel Inputs
ADC_Negative_ChannelIn	Differential Analog Negative Channel Inputs
Vref_H	Voltage Reference Select High
Vref_L	Voltage Reference Select Low
VDDA	Analog Power Supply
VSSA	Analog Ground

34.3 Memory map and register definition

This section provides the memory map and register definitions of the ADC module.

34.3.1 Memory map

ADC0 base address: 0x4003B000

ADC1 base address: 0x40027000

Table 34-2 ADC memory map

Offset	Register	Width (In bits)	Reset Value
0x004	ADC Parameter Register (ADC_PARAM)	32	0x00001000
0x008	ADC Control Register (ADC_CTRL)	32	0x0001FB00
0x00C	ADC Status Register (ADC_STAT)	32	0x00000100
0x010	ADC Interrupt Enable Register (ADC_IE)	32	0x00000000
0x014	ADC Configure information (ADC_CFG)	32	0x00000000
0x018	ADC Compare Value Register (ADC_CV)	32	0x00000000
0x020	ADC Data Register (ADC_DATA_RD)	32	0x00000000
0x024	ADC FIFO Configuration Register (ADC_FCTRL)	32	0x01000000
0x028	ADC Software Trigger Register (ADC_SWTRIG)	32	0x00000000
0x030	ADC Command Buff Register0 (ADC_CMD_BUFF0)	32	0x00000000
0x034	ADC Command Buff Register1 (ADC_CMD_BUFF1)	32	0x00000000

34.3.2 ADC Parameter Register (ADC_PARAM)

34.3.2.1 Offset

Register	Offset
ADC_PARAM	0x004

34.3.2.2 Function

This register is used to keep the design parameter.

34.3.2.3 Fields

Field	Access	Description	Reset Value
7:0 RSVD_7_0	RO	Reserved	0x0
15:8 FIFO_SIZE	RO	FIFO Depth	0x10
31:16	RO	Reserved	0x0

Field	Access	Description	Reset Value
RSVD_31_16			

34.3.3 ADC Control Register (ADC_CTRL)

34.3.3.1 Offset

Register	Offset
ADC_CTRL	0x008

34.3.3.2 Function

This register is used to store the control information of ADC.

34.3.3.3 Fields

Field	Access	Description	Reset Value
0 ADC_EN	RW	ADC Enable 0b: ADC is disabled. 1b: ADC is enabled.	0x0
1 RSVD_1	RO	Reserved	0x0
2 RST	RW	Software Reset Reset all ADC internal logic and registers. 0b: Do not reset ADC logic and registers. 1b: Write 1 to reset ADC logic and registers. Note: The readback value of this field is always 0. After writing 1 to this field, at least 2 function clock cycles are required to complete the reset process.	0x0
3 DOZEN	RW	Doze Enable 0b: ADC DOZE is disabled. 1b: ADC DOZE is enabled.	0x0
4 TRIG_MODE_ENABLE	RW	Trig mode enable bit 0b: Normal mode. 1b: Trigger mode.	0x0
5 TRIG_MODE	RW	Trigger mode: 0b: Loop mode, only trigger0 is valid. 1b: Mapping mode, one trigger triggers one CMD buff.	0x0
6 MODE1_RST	RW	Loop reset bit Terminate the loop. This bit will be cleared after returning to the idle state.	0x0
9:7 LMD	RW	The depth of loop mode 000b: Loop mode is disabled 001b: 1 command buffer conversion will be executed 010b: 2 command buffer conversion will be executed 110b: 6 command buffer conversion will be executed 111b: Reserved.	0x6
10 IREF_SEL	RW	IREF source select 0b: External IREF	0x0

Field	Access	Description	Reset Value
		1b: Internal IREF Note: This field must be set to 1.	
21:11 STABLE_T	RW	ADC stable time Note: STABLE_T * FUNCTION_CLK > 500ns	0x3F
31:22 RSVD_31_22	RO	Reserved	0x0

34.3.4 ADC Status Register (ADC_STAT)

34.3.4.1 Offset

Register	Offset
ADC_STAT	0x00C

34.3.4.2 Function

This register is used to store the states bits of ADC.

34.3.4.3 Fields

Field	Access	Description	Reset Value
0 FWM_INT	R/W1C	FIFO Watermark Interrupt	0x0
1 FOF	R/W1C	Result FIFO Overflow Flag Indicates that more data has been written to the Result FIFO than it can hold. The newer data is not stored and the FIFO remains holding the original contents. This field will assert regardless of the value of FOF_IE. However, an interrupt request is issued only if FOF_IE is set. This flag is cleared by writing a 1. 0b: FIFO does not overflow. 1b: FIFO overflows.	0x0
2 TCOMP_INT	R/W1C	Complete Interrupt Write 1 to this bit to clear TCOMP_INT.	0x0
3 CMP_INT	R/W1C	Compare Interrupt Write 1 to this bit to clear CMP_INT.	0x0
4 DMA_REQ	RO	DMA Request 0b: No DMA request. 1b: ADC generates a DMA request. Clear automatically after the DMA transfer is done.	0x0
5 RDY	RO	Result FIFO Ready Flag 0b: Result FIFO data level not above watermark level. 1b: Result FIFO holding data above watermark level.	0x0
6 RSVD_6	RO	Reserved	0x0
7 ADC_ACTIVE	RO	ADC Active This flag shows if the ADC is currently processing a conversion, or has pending triggers to service. When the ADC is IDLE, this bit will be set to 0b. 0b: The ADC is IDLE. There is no pending trigger to service. 1b: The ADC is processing a conversion.	0x0

Field	Access	Description	Reset Value
8 LSI	RO	Indicate that loop mode1 state machine returns to idle.	0x1
9 TC_ERR	R/W1C	Trigger conflict error flag. Write 1 to this bit to clear TC_ERR.	0x0
31:10 RSVD_31_10	RO	Reserved	0x0

34.3.5 ADC Interrupt Enable Register (ADC_IE)

34.3.5.1 Offset

Register	Offset
ADC_IE	0x010

34.3.5.2 Function

This register is used to store interrupt enable bits of ADC.

34.3.5.3 Fields

Field	Access	Description	Reset Value
0 FWM_IE	RW	FIFO Watermark Interrupt Enable Configures ADC to generate watermark interrupt request when RDY flag is asserted. 0b: FIFO watermark interrupt is not enabled. 1b: FIFO watermark interrupt is enabled.	0x0
1 FOF_IE	RW	Result FIFO Overflow Interrupt Enable Configures ADC to generate overflow interrupt requests when FOF flag is asserted. 0b: FIFO overflow interrupt is not enabled. 1b: FIFO overflow interrupt is enabled.	0x0
2 TCOM_IE	RW	Trigger Completion Interrupt Enable 0b: Trigger completion interrupt is not enabled. 1b: Trigger completion interrupt is enabled.	0x0
3 CMP_IE	RW	Compare Interrupt Enable 0b: Compare interrupt is not enabled. 1b: Compare completion interrupt is enabled.	0x0
4 FWMDE	RW	FIFO Watermark DMA Enable Configures ADC to generate DMA request when RDY flag is asserted. 0b: DMA request disabled. 1b: DMA request enabled.	0x0
5 ERR_INT_IE	RW	Trigger Conflict Error Interrupt Enable 0b: Trigger conflict error interrupt is not enabled. 1b: Trigger conflict error interrupt is enabled.	0x0
31:6 RSVD_31_6	RO	Reserved	0x0

34.3.6 ADC Configure Information (ADC_CFG)

34.3.6.1 Offset

Register	Offset
ADC_CFG	0x014

34.3.6.2 Function

This register is used to store ADC configure information.

34.3.6.3 Fields

Field	Access	Description	Reset Value
0 DIFF_MODE	RW	Differential Mode Select 0b: Single-Ended Mode selected. 1b: Differential Mode selected.	0x0
5:1 CHSELP	RW	Input Channel-P Select 00000b: Channel-P[0] selected. 00001b: Channel-P[1] selected. 00010b: Channel-P[2] selected. ... 10011b: Channel-P[19] selected. 10100b: bandgap selected, if bandgap was monitored under ADC, PMU_CSR[REF_BUF_1V_EN] must be set to buffer bandgap output. 11001b: DAC selected, if DAC was monitored under ADC, CMP_DCR[DAC_BUFF_EN] must be set to buffer DAC output. 11100b: LDO_CORE selected, only ADC0 supports this channel. 11101b: LDO_FLASH selected, only ADC0 supports this channel. 11110b: LDO_CLK selected, only ADC0 supports this channel. Others: Reserved. Note: The input pin selection of some special channels (ADC0_CH8, ADC0_CH9, ADC1_CH14, ADC1_CH15) is configured with SCM_MISCCTL2[24:27].	0x0
7:6 CHSELN	RW	Input Channel-N Select 00b: Channel-N[0] selected. 01b: Channel-N[1] selected. 10b: Channel-N[2] selected. 11b: Channel-N[3] selected. Note: The input pin selection of some special channels (ADC0_CH8, ADC0_CH9, ADC1_CH14, ADC1_CH15) is configured with SCM_MISCCTL2[24:27].	0x0
10:8 RSVD_10_8	RO	Reserved	0x0
11 CTYPE	RW	Conversion Type 0b: Single conversion. 1b: Continuous conversions.	0x0
14:12 CMPS	RW	Compare Select Select the compare method. 000b: Compare disabled. 001b: Less-than CVL. 010b: Greater-than CVL. 011b: Equal-to CVL.	0x0

Field	Access	Description	Reset Value
		100b: Within-range [CVL,CVH]. 101b: Out-of-range [0,CVL) ∪ (CVH,FULL]. Others: Compare disabled.	
15 HWT_EN	RW	Hardware Trigger Enable 0b: Hardware trigger disable. 1b: Hardware trigger enable.	0x0
16 REF_SEL	RW	Reference Select 0b: Select external reference VREFH and VREFL. 1b: Tied VREFH to VDDA; Tied VREFL to VSSA. Note: This field is recommended to be set to 0.	0x0
18:17 RES	RW	Resolution Select Select the resolution for conversion. 00b: 12-bit. 01b: 10-bit. 10b: 8-bit. 11b: 12-bit.	0x0
21:19 AVGS	RW	Hardware Average Select Select how many ADC conversions are averaged to create the ADC result. 000b: Single conversion. 001b: 4 conversions averaged. 010b: 8 conversions averaged. 011b: 16 conversions averaged. 100b: 32 conversions averaged. Others: Single conversion.	0x0
31:22 STS	RW	Sample Time Select Select the time interval between two samples when average is enabled or in continuous mode. NOTE STS*FUNCTION_CLK>500ns. For better accuracy, it is recommended to set STS*FUNCTION_CLK>=1000ns to get the sufficient sampling time.	0x0

34.3.7 ADC Compare Value Register (ADC_CV)

34.3.7.1 Offset

Register	Offset
ADC_CV	0x018

34.3.7.2 Function

This register is used to store the ADC compare threshold.

34.3.7.3 Fields

Field	Access	Description	Reset Value
11:0 CVL	RW	Compare Value Low	0x0
15:12 RSVD_15_12	RO	Reserved	0x0

Field	Access	Description	Reset Value
27:16 CVH	RW	Compare Value High CVH is only used in Within-Range/Out-of-Range mode.	0x0
31:28 RSVD_31_28	RO	Reserved	0x0

34.3.8 ADC Data Register (ADC_DATA_RD)

34.3.8.1 Offset

Register	Offset
ADC_DATA_RD	0x020

34.3.8.2 Function

This register is used to be as an entry to ADC FIFO.

34.3.8.3 Fields

Field	Access	Description	Reset Value
16:0 ADC_DATA_RD	RO	16:12 channel number 11:0 conversion result	0x0
31:17 RSVD_31_17	RO	Reserved	0x0

34.3.9 ADC FIFO Configuration Register(ADC_FCTRL)

34.3.9.1 Offset

Register	Offset
ADC_FCTRL	0x024

34.3.9.2 Function

This bit is used to configure the ADC FIFO.

34.3.9.3 Fields

Field	Access	Description	Reset Value
4:0 FCOUNT	RO	FIFO Counter This read-only field indicates the number of data words that are stored in the FIFO.	0x0
15:5 RSVD_15_5	reserve	Reserved	0x0
19:16 FWMARK	RW	FIFO Watermark FWMARK is a programmable threshold setting. When the number of data words stored in the FIFO is greater than the value in this field, the RDY flag is asserted to indicate stored data has reached the programmable threshold. When FWM_IE is set, an interrupt request is generated. When FWMDE is set, a DMA request is generated.	0x0
24:20	RW	FIFO Depth Redefine	0x10

Field	Access	Description	Reset Value
FDRD		Redefine the depth of FIFO. At least 1, FDRD >16 or FDRD = 0 are reserved. 00001b: FIFO depth is 1 00010b: FIFO depth is 2 ... 01111b: FIFO depth is 15 10000b: FIFO depth is 16 10001b: Reserved ... 11111b: Reserved	
31:25 RSVD_31_25	RO	Reserved	0x0

34.3.10 ADC Software Trigger Register (ADC_SWTRIG)

34.3.10.1 Offset

Register	Offset
ADC_SWTRIG	0x028

34.3.10.2 Function

This register is used to trigger ADC conversion by software.

34.3.10.3 Fields

Field	Access	Description	Reset Value
0 ADC_SWTRIG	WO	ADC Software Trigger Write 1 to this bit to trigger ADC conversion.	0x0
31:1 RSVD_31_1	RO	Reserved	0x0

34.3.11 ADC Command Buff Register0 (ADC_CMD_BUFF0)

34.3.11.1 Offset

Register	Offset
ADC_CMD_BUFF0	0x030

34.3.11.2 Function

This register is the ADC command buffer register.

34.3.11.3 Fields

Field	Access	Description	Reset Value
4:0 CMD0	RW	cmd_buff0, the user could fill the field with the p channel number, trigger mode does not support different mode	0x0
5 CMD0_INT_EN	RW	cmd0 interrupt enable	0x0

Field	Access	Description	Reset Value
7:6 RSVD_7_6	RO	Reserved	0x0
12:8 CMD1	RW	cmd_buff1, the user could fill the field with the p channel number, trigger mode does not support different mode	0x0
13 CMD1_INT_EN	RW	cmd1 interrupt enable	0x0
15:14 RSVD_15_14	RO	Reserved	0x0
20:16 CMD2	RW	cmd_buff2, the user could fill the field with the p channel number, trigger mode does not support different mode	0x0
21 CMD2_INT_EN	RW	cmd2 interrupt enable	0x0
23:22 RSVD_23_22	RO	Reserved	0x0
28:24 CMD3	RW	cmd_buff3, the user could fill the field with the p channel number, trigger mode does not support different mode	0x0
29 CMD3_INT_EN	RW	cmd3 interrupt enable	0x0
31:30 RSVD_31_30	RO	Reserved	0x0

34.3.12 ADC Command Buff Register1 (ADC_CMD_BUFF1)

34.3.12.1 Offset

Register	Offset
ADC_CMD_BUFF1	0x034

34.3.12.2 Function

This register is the ADC command buffer register.

34.3.12.3 Fields

Field	Access	Description	Reset Value
4:0 CMD4	RW	cmd_buff4, the user could fill the field with the p channel number, trigger mode does not support different mode	0x0
5 CMD4_INT_EN	RW	cmd4 interrupt enable	0x0
7:6 RSVD_7_6	RO	Reserved	0x0
12:8 CMD5	RW	cmd_buff5, the user could fill the field with the p channel number, trigger mode does not support different mode	0x0
13 CMD5_INT_EN	RW	cmd5 interrupt enable	0x0
31:14 RSVD_31_14	RO	Reserved	0x0

34.4 Function description

34.4.1 ADC voltage reference select

There are two types of voltage reference for ADC: voltage reference high (VREFH) and voltage reference low (VREFL). VREFH and VREFL can be selected from the external pin or tied to a fixed voltage by controlling ADC_CFG[REF_SEL], when this bit is set to 1, VREFH is tied to VDDA, and VREFL is set to VSSA; when this bit is set to 0, VREFH and VREFL are supplied from the external pin.

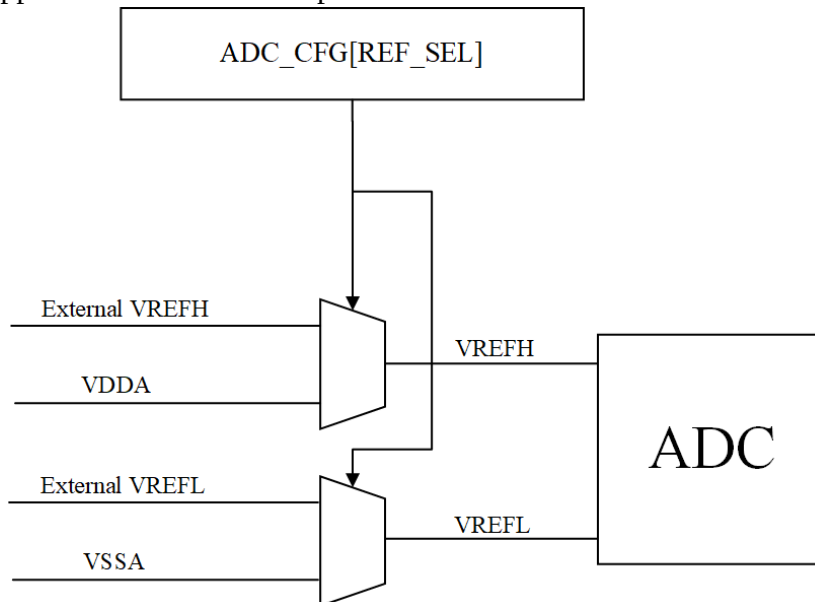


Figure 34-2 ADC VREF selection diagram

34.4.2 Channel and mode select

ADC can support Single-Ended mode and Differential mode with ADC_CFG[DIFF_MODE] to select the operation mode. Single-Ended mode only needs one input channel, and the user can use ADC_CFG[CHSELP] to select one of the 20 P channels as the operation channel. N channel is useless in Single-Ended mode. If trig mode is enabled, the trigger function state machine will output signal to select the operation channel.

The ADC supports up to three differential channel inputs in Differential mode. Differential mode needs two channels for input, P channel and N channel. The user needs to use ADC_CFG[CHSELP] to select the P channel and use ADC_CFG[CHSELN] to select the N channel. Please note that, when ADC_CFG[CHSELN] was set to 3, the N channel was connected to the VSS.

For conversion times, the user could select single mode and continue mode by setting ADC_CFG[CTYPE]. In single mode, the conversion will be only executed once and the conversion result will be put into ADC_DATA. In continue mode, the conversion result will be put into the FIFO. After the FIFO reaches the watermark, the ADC will generate an IRQ or DMA request to inform CPU.

34.4.3 Start trigger select

After enabling the ADC with ADC_CTRL[ADC_EN] set, the ADC still needs a trigger to start the conversion. There are two trigger sources in ADC: software trigger and hardware trigger. For software trigger, the user can trigger the conversion after writing 1 to ADC_SWTRIG[ADC_SWTRIG]. If the user wants to trigger another conversion, the user should write the ADC_SWTRIG bit again. For hardware trigger, it is an input signal for ADC. This input can come from the external pin or from other source on the chip through the TMU

module. The ADC will start one conversion at the posedge of the hardware trigger.

34.4.4 Loop trigger mode and map trigger mode

There are 6 input triggers from TDG. The ADC module provides two trigger modes for the 6 triggers: loop trigger mode and map trigger mode. If the user wants to use the two modes, the user should configure `ADC_CTRL[TRIG_MODE_ENABLE]` first, and use `ADC_CTRL[TRIG_MODE]` to select enable loop trigger mode or map trigger mode.

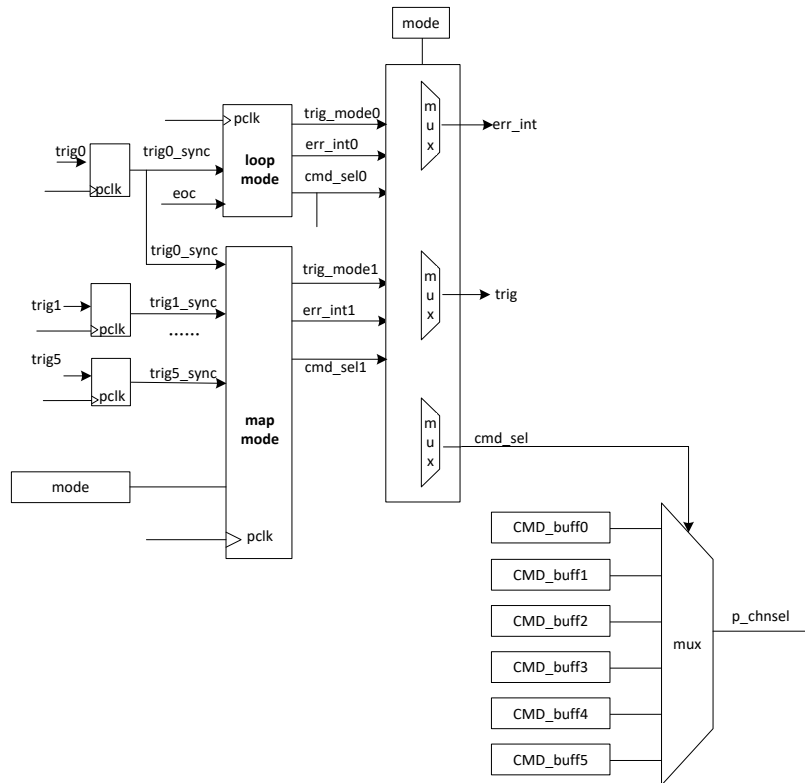


Figure 34-3 ADC Loop Trigger Mode and Mapping Trigger Mode Diagram

In loop trigger mode, only trigger0 is valid. When trigger0 changes from 0 to 1, ADC will be triggered to loop from `CMD_BUFF0` to `CMD_BUFF5` in which the conversion channel and `TCOMP` interrupt enable bit for this channel are stored. The user can use `ADC_CTRL[LMD]` to control the loop depth, for example, when the `ADC_CTRL[LMD]` was configured to 3, the loop will execute `CMD_BUFF0`, `CMD_BUFF1`, `CMD_BUFF2` and then the loop will end. When the loop is not ended but if there is another trigger0 posedge coming, the loop will go on and will set `TRIG_CONF_ERR` flag in `ADC_STAT`. In loop trigger mode, trigger1 to trigger5 are invalid, which will be ignored if asserted.

In map trigger mode, trigger0 to trigger5 are all valid, one trigger maps one `CMD_BUFF`, trigger0 maps `CMD_BUFF0`, trigger1 maps `CMD_BUFF1` and so on. If another trigger comes when last conversion is not finished, `TRIG_CONF_ERR` flag in `ADC_STAT` will be set. The conversion result and the conversion channel number will be pushed into the FIFO after conversion is finished.

34.4.5 Compare function

After the conversion is done and the result is sampled, it may trigger a compare function decided by the configuration of `ADC_CFG[CMPS]`, and the ADC output will be compared with the value in the register `CVL` and `CVH`. If the ADC output meets the condition, hardware can generate an IRQ or DMA request. The detail condition of compare function is listed as below:

Table 34-3 The configuration of compare function

ADC_CFG[CMPS]	Compare Condition
000b	Compare disable
001b	Less than CVL
010b	Greater than CVL
011b	Equal-to CVL
100b	Within-range [CVL,CVH]
101b	Out-of-range [0,CVL) ∪ (CVH,FULL]
others	Compare disable

34.4.6 Conversion function

For conversion function, the user could select the operation mode: Differential mode or Single-Ended mode, select the input P channel and N channel, select to trigger IRQ request or DMA request after conversion is done. The detail configuration is listed as below:

1. Set ADC_CFG[DIFF_MODE] to select the operation mode.
2. Set ADC_CTRL[STABLE_T] to select the stable time, the unit is the function clock period.
3. Set ADC_FCTRL[FWMARK] to setup the threshold to trigger DMA or IRQ request.
4. Set ADC_IE to enable IRQ or DMA request.
5. Set ADC_CFG[AVGS] to select the average times.
6. Set ADC_CFG[STS] to set the sample time.
7. Set ADC_CFG[RES] to select the conversion resolution.
8. Set ADC_CTRL[IREF_SEL] to internal Iref.
9. Set ADC_CTRL[ADC_EN] to enable ADC.
10. Set ADC_SWTRIG[ADC_SWTRIG] to trigger the conversion or set ADC_CFG[HWT_EN] to enable hardware trigger.

34.4.7 Conversion time

The calculation method for the ADC sampling conversion time is as follows:

1. For a single sampling, assuming the average number of conversions is M, the time taken from triggering the ADC sampling to the completion of the conversion (denoted as Tsingle) is given by:

$$T_{\text{single}} = \text{Stable time} + \text{Conversion time} + \text{Sample time} \times (M-1)$$

- The conversion time ranges from 300ns to 700ns.
2. For loop mode, assuming there are N channels to be converted, the total time taken from triggering the ADC sampling to the completion of all conversions is:

$$T_{\text{total}} = T_{\text{single}} \times N$$

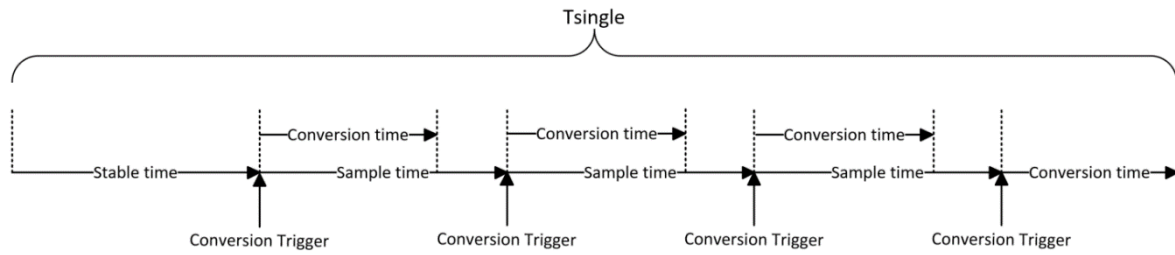


Figure 34-4 The time of the ADC conversion process

Chapter 35

Comparator (CMP)

35.1 Introduction

The comparator is a module that compares two analog input voltages and outputs the compare result.

The compare result can be selected as sampled, window sampled, or filtered.

The analog comparator can work in two speeds: high speed comparator (HSCMP) and low-speed comparator (LSCMP).

There are two analog muxes in the analog part, each of them has 13 external analog inputs and an 8-bit resolution internal DAC input.

The DAC output can be an input channel of the analog comparator. The module supports programmable hysteresis control for both HSCMP and LSCMP. Users can choose to sample/window/filter the analog comparator output.

35.2 Features

The CMP supports the following key features:

- The CMP can be operated over the entire supply range
- The hysteresis is programmable for both HSCMP and LSCMP
- The IRQ/DMA request can be generated on rising-edge, falling-edge, or both rising or falling edges of the comparator output
- The analog output of comparator can be:
 - Sampled
 - Filtered
 - Inverted
- The 8 bits resolution DAC reference voltage can be VDDA or VREFH, and the DAC output can be routed to the comparator as an input.

35.2.1 Block diagram

The following figure is a block diagram of the comparator (CMP) module.

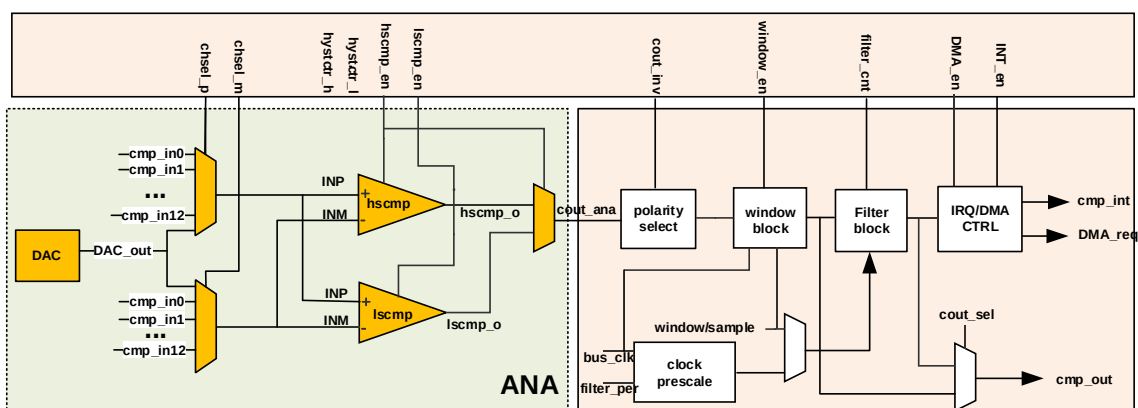


Figure 35-1 CMP block diagram

35.2.2 Instantiation and channel information

The following table lists the CMP instantiation and channel information for each chip in the product series.

Configuration	K144MC	K146MC	K148MC
Instance(s)	CMP	CMP	CMP
Channel(s)	13	13	13

35.2.3 Signal description

Table 35-1 CMP signal descriptions

Signal	Description	I/O
CMP_IN0 - CMP_IN12	Analog voltage inputs	I
CMP_OUT	Output of comparator	O

35.3 Memory map and register definition

This section provides the memory map and register definitions of the CMP module.

35.3.1 Memory map

CMP base address: 0x40073000

Table 35-2 CMP memory map

Offset	Register	Width (In bits)	Reset Value
0x4	CMP Parameter Register (CMP_PARAM)	32	0x00000002
0x8	Comparator Control Register 0 (CMP_CCR0)	32	0x00000001
0xC	Comparator Control Register 1 (CMP_CCR1)	32	0x00000000
0x10	Comparator Control Register 2 (CMP_CCR2)	32	0x00000000
0x14	CMP DAC Control Register (CMP_DCR)	32	0x00000000
0x18	CMP Interrupt Enable Register (CMP_IER)	32	0x00000000
0x1C	CMP Comparator Status Register (CMP_CSR)	32	0x00000000

35.3.2 CMP Parameter Register (CMP_PARAM)

35.3.2.1 Offset

Register	Offset
CMP_PARAM	0x4

35.3.2.2 Function

This is the parameter register.

35.3.2.3 Fields

Field	Access	Description	Reset Value
3:0 DAC_RES	RO	DAC resolution 0000b: 4-bit DAC 0001b: 6-bit DAC 0010b: 8-bit DAC 0011b: 10-bit DAC 0100b: 12-bit DAC 0101b: 14-bit DAC 0110b: 16-bit DAC	0x2
31:4 RSVD_31_4	RO	Reserved	0x0

35.3.3 Comparator Control Register 0 (CMP_CCR0)

35.3.3.1 Offset

Register	Offset
CMP_CCR0	0x8

35.3.3.2 Function

This is the comparator control register 0.

35.3.3.3 Fields

Field	Access	Description	Reset Value
0 CMP_SEL	RW	Low speed/high speed comparator select 0b: Low speed comparator 1b: High speed comparator	0x1
1 CMP_EN	RW	Comparator enable 0b: Disable 1b: Enable	0x0
2 SOC_TRG_EN	RW	SOC trigger enable 0b: Disable 1b: Enable CMP can be enabled by signal from SoC trigger when the trigger signal is 1.	0x0
3 STOP_EN	RW	Configure whether CMP can generate interrupt or DMA in STOP mode 0b: Disable 1b: Enable	0x0
31:4 RSVD_31_4	RO	Reserved	0x0

35.3.4 Comparator Control Register 1 (CMP_CCR1)

35.3.4.1 Offset

Register	Offset
CMP_CCR1	0xC

35.3.4.2 Function

This is the comparator control register1.

35.3.4.3 Fields

Field	Access	Description	Reset Value
0 WINDOW_EN	RW	Window mode enable When enabled, the input sample/window signal is used as a CMP_OUT window. The analog CMP will be output to SoC or pad only when the window signal is asserted. 0b: Window mode disable. 1b: Window mode enable.	0x0
1 SAMPLE_EN	RW	Sample mode enable. When enabled, the input sample/window signal is used as a sample signal to sample the comparator output. Users can only choose either sample enable or window enable. 0b: Sample mode disable. 1b: Sample mode enable.	0x0
2 DMA_EN	RW	DMA enable. 0b: Generate interrupt if CSR[CFR]&IER[CFR_IE] or CSR[CFF]&IER[CFF_IE] is set. 1b: Generate DMA request if CSR[CFR]&IER[CFR_IE] or CSR[CFF]&IER[CFF_IE] is set.	0x0
3 COUT_INV	RW	Comparator invert Invert the comparator output 0b: Do not invert the comparator output. 1b: Invert the comparator output.	0x0
4 COUT_SEL	RW	CMP_OUT select 0b: Filter output. 1b: Window output.	0x0
5 COUT_PEN	RW	CMP output to pad 0b: Do not send cout to PAD. 1b: Send the cout to a PAD.	0x0
6 COUT_OWE	RW	Users can define the COUTA level when the window is closed. 0b: When window is closed, COUTA holds the value of the last sample. 1b: When window is closed, COUTA comes from the definition of COUTA_OW.	0x0
7 COUT_OW	RW	Users can define the COUTA level when window is closed. 0b: When window is closed, COUTA is 0. 1b: When window is closed, COUTA is 1.	0x0
15:8 RSVD_15_8	RO	Reserved	0x0
18:16 FILT_CNT	RW	Filter sample count 000b: Filter is bypassed. 001b: 1 consecutive sample must agree. 010b: 3 consecutive sample must agree. 011b: 4 consecutive sample must agree. 100b: 5 consecutive sample must agree. 101b: 6 consecutive sample must agree. 110b: 7 consecutive sample must agree. 111b: 8 consecutive sample must agree.	0x0

Field	Access	Description	Reset Value
		Note: Filter block samples the comparator output only on the rising edge of sample signal.	
23:19 RSVD_23_19	RO	Reserved	0x0
31:24 FILT_PER	RW	Filter sample period When filter mode is enabled, this field defines the sample period of the filter. It is calculated: $P_{\text{filter}} = \text{FILT\_PER} * P_{\text{bus}}$ P_{filter} : the sample period of filter block P_{bus} : the period of bus clock NOTE SAMPLE_EN=0, FILT_PER=0 means bypassing the filter. SAMPLE_EN=1, FILT_PER has no effect.	0x0

35.3.5 Comparator Control Register 2 (CMP_CCR2)

35.3.5.1 Offset

Register	Offset
CMP_CCR2	0x10

35.3.5.2 Fields

Field	Access	Description	Reset Value
1:0 HS_HSTCTR	RW	High-speed comparator hysteresis selection 00b: No hysteresis 01b: 21mV hysteresis 10b: 34mV hysteresis 11b: 45mV hysteresis	0x0
3:2 RSVD_3_2	RO	Reserved	0x0
5:4 LS_HSTCTR	RW	Low-speed comparator hysteresis selection 00b: No hysteresis 01b: 16mV hysteresis 10b: 25mV hysteresis 11b: 32mV hysteresis	0x0
15:6 RSVD_15_6	RO	Reserved	0x0
19:16 HS_PSEL	RW	High speed comparator PMUX select: 0000b: CMP_IN[0] 0001b: CMP_IN[1] 1100b: CMP_IN[12] 1101b: Reserved 1110b: Reserved 1111b: DAC output	0x0
23:20 HS_MSEL	RW	High speed comparator NMUX select: 0000b: CMP_IN[0] 0001b: CMP_IN[1] 1100b: CMP_IN[12]	0x0

Field	Access	Description	Reset Value
		1101b: Reserved 1110b: Reserved 1111b: DAC output	
27:24 LS_PSEL	RW	Low speed comparator PMUX select: 0000b: CMP_IN[0] 0001b: CMP_IN[1] 1100b: CMP_IN[12] 1101b: Reserved 1110b: Reserved 1111b: DAC output	0x0
31:28 LS_MSEL	RW	Low speed comparator NMUX select: 0000b: CMP_IN[0] 0001b: CMP_IN[1] 1100b: CMP_IN[12] 1101b: Reserved 1110b: Reserved 1111b: DAC output	0x0

35.3.6 CMP DAC Control Register (CMP_DCR)

35.3.6.1 Offset

Register	Offset
CMP_DCR	0x14

35.3.6.2 Fields

Field	Access	Description	Reset Value
0 DAC_EN	RW	DAC enable 0b: Disable 1b: Enable	0x0
1 DAC_BUFF_EN	RW	The enable of DAC output to ADC 0b: Disable 1b: Enable	0x0
7:2 RSVD_7_2	RO	Reserved	0x0
8 VREF_SEL	RW	DAC reference select 0b: VDDA 1b: VREFH	0x0
15:9 RSVD_15_9	RO	Reserved	0x0
23:16 DAC_SEL	RW	DAC output voltage select The DAC output voltage is calculated as: $DAC\ output = (Vref/256) * (DAC\_SEL + 1)$	0x0
31:24 RSVD_31_24	RW	Reserved	0x0

35.3.7 CMP Interrupt Enable Register (CMP_IER)

35.3.7.1 Offset

Register	Offset
CMP_IER	0x18

35.3.7.2 Fields

Field	Access	Description	Reset Value
0 CFR_IE	RW	Comparator rising edge interrupt enable 1b: Enable rising edge interrupt/DMA request. If CSR[CFR]=1, an interrupt/DMA request is generated. 0b: Disable rising edge interrupt/DMA request.	0x0
1 CFF_IE	RW	Comparator falling edge interrupt enable 1b: Enable falling edge interrupt/DMA request. If CSR[CFF]=1, an interrupt/DMA request is generated. 0b: Disable falling edge interrupt/DMA request.	0x0
31:2 RSVD_31_2	RW	Reserved	0x0

35.3.8 Comparator Status Register (CMP_CSR)

35.3.8.1 Offset

Register	Offset
CMP_CSR	0x1C

35.3.8.2 Fields

Field	Access	Description	Reset Value
0 CFR	R/W1C	Comparator output rising edge detect flag. Write 1 to clear.	0x0
1 CFF	R/W1C	Comparator output falling edge detect flag. Write 1 to clear.	0x0
7:2 RSVD_7_2	RO	Reserved	0x0
8 COUT	RO	Comparator output. 0b: The comparator outputs low level. 1b: The comparator outputs high level.	0x0
31:9 RSVD_31_9	RO	Reserved	0x0

35.4 Function description

This section provides the functional information of the CMP module.

35.4.1 Polarity selection

CMP_CCR1[COU_INV] is for inverting the comparator output. When setting this bit, the polarity of the comparator will be inverted. This bit will also drive CMP output when CMP is disabled.

35.4.2 Basic mode

In this mode, the window block and filter block are bypassed. The signal after polarity selection is output directly. In basic mode, CMP_CCR1[WINDOW_EN], CMP_CCR1[SAMPLE_EN] and CMP_CCR1[FILT_PER] are set to 0.

35.4.3 Window control

After the polarity selection module, there are window control blocks. Users could enable the window function by setting CMP_CCR1[WINDOW_EN]. When enabled, the input sample/window signal is used as an output window. Only when window signal is asserted could the polarity module output be output to next block or output to CMP_OUT; when the window signal is deasserted, this block will output a fixed value decided by CMP_CCR1[COUTA_OW].

When CMP_CCR1[WINDOW_EN] is set to 0, the window block is bypassed, the output signal of polarity block will pass through the window block without any change.

NOTE

The sample mode and the window mode cannot be enabled at the same time. If window mode is enabled, the sample mode must be disabled.

35.4.4 Filter mode

The window block output will go to the filter block. This block could be used as a filter or as a sample module. When used as a filter, CMP_CCR1[SAMPLE_EN] is set to 0, and CMP_CCR1[FILT_PER] is not set to 0. When they are both set to 0, this block will be bypassed. The filter operates on the unfiltered and unsynchronized and optionally inverted comparator output COUT_ANA and generates the filtered and synchronized output CMP_OUT.

CMP_CCR1[FILT_PER] is used to specify the sampling period in bus clock of the output filter. CMP_CCR1[FILT_CNT] defines the consecutive sample count number. The output value can change only when the sample count number reaches the number. Setting CMP_CCR1[FILT_CNT] to 0 means bypassing the filter.

35.4.5 Sample mode

The filter block can also be used only to sample the window block output signal when CMP_CCR1[SAMPLE_EN] is set. When only used as a sample block, the window block output will be sampled whenever a rising edge is detected on the filter block clock input.

When the filter block is used in filter mode, the filter block input clock is derived from the clock prescaler, but when used in sample mode, the input clock comes from the input WINDOW/SAMPLE signal.

NOTE

The user cannot enable the sample mode and the window mode at the same time. If the sample mode is enabled, the window mode must be disabled.

35.4.6 DAC operation

There is an 8-bit resolution DAC in the analog part. The user could set CMP_DCR[DAC_EN] to enable DAC, and use CMP_DCR[VREF_SEL] to select the DAC reference voltage. The 8-bit field CMP_DCR[DAC_SEL] is used to select the DAC output voltage. The DAC output voltage should be:

$$DACO = (VREF/256) * (DAC\_SEL + 1)$$

35.4.7 IRQ/DMA request

Two flags can generate the IRQ requests or DMA request:

- Rising edge detected flag.
- Falling edge detected flag.

The user can set `CMP_IER[CFF_IE/CFR_IE]` to enable one or both of the flags.

- When `CMP_CCR1[DMA_EN]` is set to 0, an IRQ request will raise if the enabled flag is asserted.
- When `CMP_CCR1[DMA_EN]` is set to 1, a DMA request will raise if the enabled flag is asserted.

Chapter 36

Universal Asynchronous Receiver/Transmitter (UART)

36.1 Introduction

This chapter defines a Universal Asynchronous Receiver/Transmitter (UART) and Local Interconnected Network (LIN). ZX LIN controller implements some process to the LIN frame by hardware based on the Universal Asynchronous Receiver/Transmitter (UART) PHY layer. The purpose of UART is to reduce the loading of software. The controller can support LIN protocol 1.3, 2.0, 2.1, 2.2. This module contains an UART PHY layer with a full UART function and a LIN controller with checksum, baud rate auto sync, PID filter and other LIN functions. The user can use this module as a standard UART or use it as a LIN master/slave.

36.1.1 Features

The UART module contains the following features:

- Support APB3 interface for register access
- Programmable baud rates
- Transmit and receive FIFO depths of 16
- Programmable character bit length, stop bits length and optional parity bits
- Independent TX and RX DMA request supported
- Asynchronous IRQ supported with multi-interrupt sources
- Line break generation and detection
- Support 9-bit data transfer

Features of the LIN include:

- Baud rate up to 20Kbits/s
- Master/slave mode
- Header and response auto send/receive
- Classic and enhanced checksum calculation and check
- PID parity calculation and check
- Up to 16 PID filter
- Receiver timeout detection
- Baud rate auto sync

36.1.2 Block diagram

The following figure is a block diagram of the Universal Asynchronous Receiver/Transmitter (UART) module.

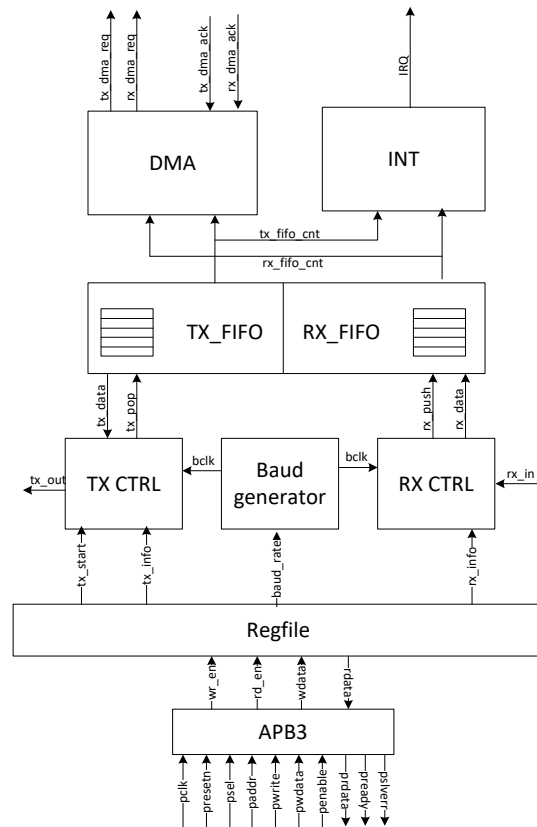


Figure 36-1 UART block diagram

The following figure is a block diagram of the Local Interconnected Network (LIN) module.

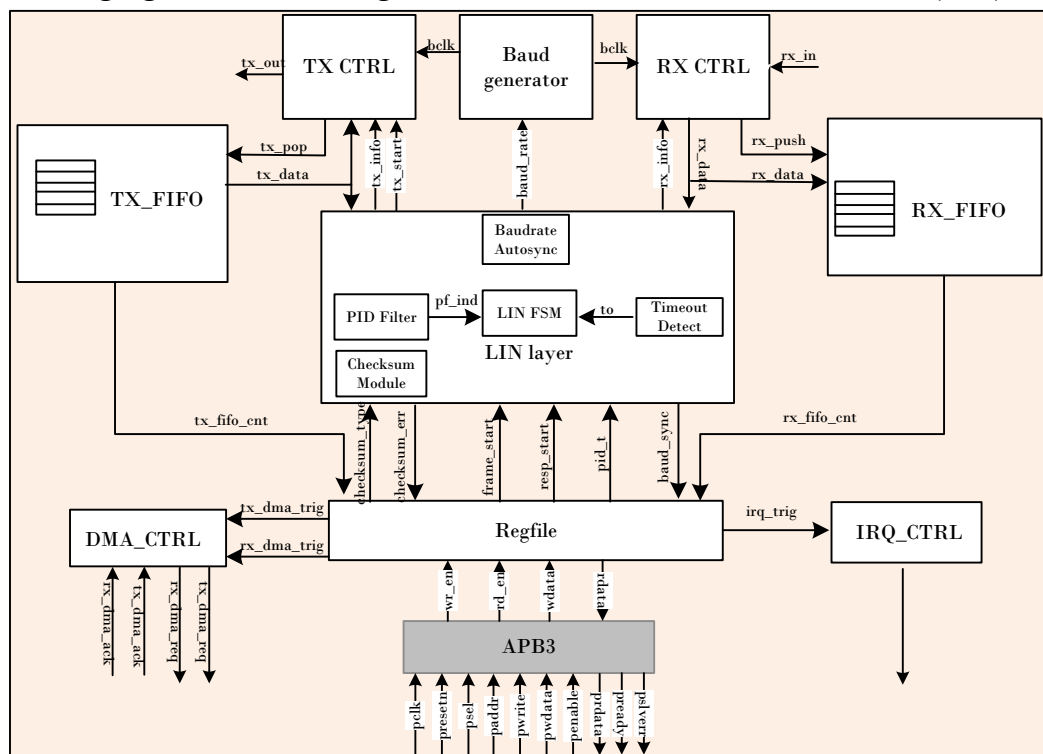


Figure 36-2 LIN block diagram

36.1.3 Instantiation information

The following table lists the UART instantiation information for each chip in the product series.

Configuration	K144MC	K146MC	K148MC
Instance(s)	UART0 (LIN0)	UART0 (LIN0)	UART0 (LIN0)
	UART1 (LIN1)	UART1 (LIN1)	UART1 (LIN1)
	UART2 (LIN2)	UART2 (LIN2)	UART2 (LIN2)
	UART3 (LIN3)	UART3 (LIN3)	UART3 (LIN3)
	UART4 (LIN4)	UART4 (LIN4)	UART4 (LIN4)
	UART5 (LIN5)	UART5 (LIN5)	UART5 (LIN5)

36.2 Signal description

Table 36-1 UART signal descriptions

Signal	Description	I/O
Tx	Transmit data.	Output
Rx	Receive data.	Input
RTS	Request to send.	Output
CTS	Clear to send.	Input

36.3 Memory map and register definition

This section provides the memory map and register definitions of the UART/LIN module.

36.3.1 Memory map

UART0 (LIN0) base address: 0x4006A000

UART1 (LIN1) base address: 0x4006B000

UART2 (LIN2) base address: 0x4006C000

UART3 (LIN3) base address: 0x40070000

UART4 (LIN4) base address: 0x40071000

UART5 (LIN5) base address: 0x40072000

Table 36-2 UART/LIN memory map

Offset	Register	Width (In bits)	Reset value
0x0	Receive Buffer Register (UART_RBR)	32	0x00000000
0x0	Transmit Holding Register (UART_THR)	32	0x00000000
0x0	Divisor Latch Low (UART_DLL)	32	0x00000000
0x4	Divisor Latch High (UART_DLH)	32	0x00000000
0x4	Interrupt Enable Register (UART_IER)	32	0x00000000
0x8	Interrupt Identity Register (UART_IIR)	32	0x00000001
0x8	FIFO Control Register	32	0x00000000

Offset	Register	Width (In bits)	Reset value
	(UART_FCR)		
0xC	Line Control Register (UART_LCR)	32	0x00000000
0x10	Autoflow Control Register (UART_AFCR)	32	0x00000000
0x14	Line Status Register (UART_LSR)	32	0x00000060
0x18	Modem Status Register (UART_MSR)	32	0x00000000
0x7C	UART Status Register (UART_USR)	32	0x00000000
0xC0	Fraction Divisor Register (UART_FD)	32	0x00000000
0xC4	Receive Address Register (UART_RAR)	32	0x00000000
0xC8	Transmit Address Register (UART_TAR)	32	0x00000000
0xCC	Line Extended Control Register (UART_LCR_EXT)	32	0x00000000
0x100	LIN Mode Control Register (UART_LIN_CTL)	32	0x00000000
0x104	LIN Response Length Register (UART_LIN_RSP_LENGTH)	32	0x00000000
0x108	LIN ID Register (UART_LIN_ID_VALUE)	32	0x00000000
0x10C	LIN Checksum Register (UART_LIN_CHECKSUM)	32	0x00000000
0x110	LIN Delimiter Field Length (UART_LIN_DEL_LENGTH)	32	0x00000000
0x114	LIN PID Filter 0 (UART_LIN_PID_FILTER_0)	32	0x00000000
0x118	LIN PID Filter 1 (UART_LIN_PID_FILTER_1)	32	0x00000000
0x11C	LIN PID Filter 2 (UART_LIN_PID_FILTER_2)	32	0x00000000
0x120	LIN PID Filter 3 (UART_LIN_PID_FILTER_3)	32	0x00000000
0x124	LIN PID Filter Vague (UART_LIN_PID_FILTER_VAGUE)	32	0x00000000
0x128	LIN PID Filter Control Register (UART_LIN_PID_FILTER_CTRL)	32	0x00000000

36.3.2 Receive Buffer Register (UART_RBR)

36.3.2.1 Offset

Register	Offset
UART_RBR	0x0

36.3.2.2 Function

This register is used for reading the received data.

36.3.2.3 Fields

Field	Access	Description	Reset value
7:0 RBR_LSB	RO	Data byte received on the serial input port in UART mode. The data in this register is valid only if the Data Ready (DR) bit in the Line Status Register (UART_LSR) is set.	0x0
8 RBR_MSB	RO	Data byte received on the serial input port in UART mode for the MSB 9th bit.	0x0
31:9 RSVD_31_9	RO	Reserved	0x0

36.3.3 Transmit Holding Register (UART_THR)

36.3.3.1 Offset

Register	Offset
UART_THR	0x0

36.3.3.2 Function

This register is used for writing the transmit data.

36.3.3.3 Fields

Field	Access	Description	Reset value
7:0 THR_LSB	WO	Data to be transmitted on the serial output port in UART mode. Data should only be written to the UART_THR when the UART_THR Empty (THRE) bit (UART_LSR[5]) is set.	0x0
8 THR_MSB	WO	Data to be transmitted on the serial output port in UART mode for the MSB 9th bit.	0x0
31:9 RSVD_31_9	RO	Reserved	0x0

36.3.4 Divisor Latch Low (UART_DLL)

36.3.4.1 Offset

Register	Offset
UART_DLL	0x0

36.3.4.2 Function

This register is used for lower 8 bits of divisor register.

36.3.4.3 Fields

Field	Access	Description	Reset value
7:0 DL_L	RW	Lower 8 bits of a 16-bit, divisor latch register that contains the baud rate divisor for the UART. This register can be accessed only when the DLAB bit	0x0

Field	Access	Description	Reset value
		(UART_LCR[7]) is set and the UART is not busy.	
31:8 RSVD_31_8	RO	Reserved	0x0

36.3.5 Divisor Latch High (UART_DLH)

36.3.5.1 Offset

Register	Offset
UART_DLH	0x4

36.3.5.2 Function

This register is used for upper 8 bits of divisor register.

36.3.5.3 Fields

Field	Access	Description	Reset value
7:0 DL_H	RW	Upper 8 bits of a 16-bit, divisor latch register that contains the baud rate divisor for the UART. The output baud rate is equal to the UART function clock frequency divided by sixteen times the value of the baud rate divisor, as follows: baud rate = (function clock frequency) / (16 * divisor). This register can be accessed only when the DLAB bit (UART_LCR[7]) is set and the UART is not busy.	0x0
31:8 RSVD_31_8	RO	Reserved.	0x0

36.3.6 Interrupt Enable Register (UART_IER)

36.3.6.1 Offset

Register	Offset
UART_IER	0x4

36.3.6.2 Function

This register is used for enabling or masking interrupt.

36.3.6.3 Fields

Field	Access	Description	Reset value
0 ERBFI	RW	Enable received data available interrupt. This is used to enable/disable the generation of Received Data Available Interrupt. 0b: Disabled. 1b: Enabled.	0x0
1 ETBEI	RW	Enable transmit holding register empty interrupt. This is used to enable/disable the generation of Transmitter Holding Register Empty Interrupt.	0x0

Field	Access	Description	Reset value
		0b: Disabled. 1b: Enabled.	
2 ELSI	RW	Enable receiver line status interrupt. This is used to enable/disable the generation of Receiver Line Status Interrupt. 0b: Disabled. 1b: Enabled.	0x0
3 EDSSI	RW	This is used to enable/disable the generation of modem status interrupt. This is the fourth highest priority interrupt. 0b: Disabled. 1b: Enabled.	0x0
6:4 RSVD_6_4	RO	Reserved	0x0
7 PTIME	RW	TX FIFO under watermark interrupt enable This bit is only useful when FIFO is enabled and UART_IER[ETBEI] is set. 0b: When FIFO is enabled and UART_IER[ETBEI] is set, THRE interrupt will set when TX FIFO is empty. 1b: When FIFO is enabled and UART_IER[ETBEI] is set, THRE interrupt will set when TX FIFO under watermark.	0x0
8 HEADER_DONE_INT_EN	RW	Header operation done interrupt enable. 0b: Disabled. 1b: Enabled.	0x0
9 RSP_DONE_INT_EN	RW	Response operation done interrupt enable. 0b: Disabled. 1b: Enabled.	0x0
10 ASYNC_INT_EN	RW	Asynchronous interrupt enable. 0b: Disabled. 1b: Enabled.	0x0
31:11 RSVD_31_11	RO	Reserved	0x0

36.3.7 Interrupt Identity Register (UART_IIR)

36.3.7.1 Offset

Register	Offset
UART_IIR	0x8

36.3.7.2 Function

This register is used for identifying the interrupt type.

36.3.7.3 Fields

Field	Access	Description	Reset value
3:0 IID	RO	Interrupt ID. This indicates the highest priority pending interrupt which can be one of the following types: 0000b: CTS flag assert 0001b: No interrupt pending	0x1

Field	Access	Description	Reset value
		0010b: THR empty 0100b: Received data available 0110b: Receiver line status 0111b: Busy detect 1100b: Character time out 1101b: LIN: header operation done 1110b: LIN: response operation done	
5:4 RSVD_5_4	RO	Reserved	0x0
7:6 FIFOSE	RO	FIFOs enabled. This is used to indicate whether the FIFOs are enabled or disabled. 00b: Disabled. 11b: Enabled. 01b,10b: Reserved	0x0
31:8 RSVD_31_8	RO	Reserved	0x0

36.3.8 FIFO Control Register (UART_FCR)

36.3.8.1 Offset

Register	Offset
UART_FCR	0x8

36.3.8.2 Function

This register is used to control the FIFO configuration.

36.3.8.3 Fields

Field	Access	Description	Reset value
0 FIFOE	WO	FIFO enable Write 1 to enable FIFO.	0x0
1 RFIFOR	WO	RCVR FIFO reset This resets the control portion of the receive FIFO and treats the FIFO as empty.	0x0
2 XFIFOR	WO	XMIT FIFO reset This resets the control portion of the transmit FIFO and treats the FIFO as empty.	0x0
3 RSVD_3	WO	Reserved	0x0
5:4 TET	WO	This is used to select the empty threshold level at which the THRE Interrupts are generated when the mode is active. 00b: FIFO empty 01b: 2 characters in the FIFO 10b: FIFO $\frac{1}{4}$ full 11b: FIFO $\frac{1}{2}$ full Note: Programmable THRE interrupt mode can be configured by PTIME field.	0x0
7:6 RCVR	WO	This is used to select the trigger level in the receiver FIFO at which the Received Data Available Interrupt is generated. 00b: 1 character in the FIFO	0x0

Field	Access	Description	Reset value
		01b: FIFO ¼ full 10b: FIFO ½ full 11b: FIFO 2 less than full	
31:8 RSVD_31_8	RO	Reserved	0x0

36.3.9 Line Control Register (UART_LCR)

36.3.9.1 Offset

Register	Offset
UART_LCR	0xC

36.3.9.2 Function

This register is the transmit and receive control register.

36.3.9.3 Fields

Field	Access	Description	Reset value
1:0 DLS	RW	Data length select. 00b: 5 bits 01b: 6 bits 10b: 7 bits 11b: 8 bits	0x0
2 STOP	RW	Number of stop bits. 0b: 1 stop bit 1b: 1.5 stop bits when DLS (UART_LCR[1:0]) is 0, or else 2 stop bits	0x0
3 PEN	RW	Parity enable. 0b: Parity disable 1b: Parity enable	0x0
4 EPS	RW	Even parity select. 0b: Odd parity 1b: Even parity	0x0
5 RSVD_5	RW	Reserved.	0x0
6 BC	RW	Break control bit. This is used to cause a break condition to be transmitted to the receiving device. When UART_LCR[LBKEN] is set to 1: If set BC bit to 1, the serial output is forced to output logic 0, the length of which is defined by UART_LCR[LBKM]. After the sent break reaches the set length, BC will be cleared automatically. When UART_LCR[LBKEN] is set to 0: If set BC bit to 1, the serial output is forced to output logic 0 until the BC bit is cleared.	0x0
7 DLAB	RW	Divisor latch access bit. This bit is used to enable reading and writing of the Divisor Latch register (UART_DLL and UART_DLH) to set the baud rate of the UART. Note: UART_RBR, UART_THR, UART_IER registers cannot be accessed when this bit is	0x0

Field	Access	Description	Reset value
		asserted: UART_DLR register cannot be accessed when this bit is deasserted. 0h: Divisor unlatch. 1h: Divisor latch.	
8 LBKEN	RW	Used for UART send break to clear break send automatically. 0b: Clear break send automatically function disable. 1b: Clear break send automatically function enable.	0x0
13:9 LBKM	RW	The length of break in LIN mode, the length=LBKM+4'd11 Note: Due to internal clock synchronization, the actual output break length will be shorter than the set length, with a maximum difference of no more than 1 Tbit.	0x0
16:14 IDLE_DET_LENGTH	RW	Idle detection length. The unit of idle detection length is 1-bit character. 0h: 1-bit character 1h: 2-bit character 2h: 4-bit character 3h: 8-bit character 4h: 16-bit character 5h: 32-bit character 6h: 64-bit character 7h: 128-bit character Note: Idle detection starts from the beginning of stop bits field. Therefore, it is recommended to set idle detection length longer than stop bits.	0x0
17 IDLE_DET_EN	RW	Idle detect enable. UART supports idle line detect function, the detection length is defined by UART_LCR [IDLE_DET_LENGTH]. 0b: Idle detection is disabled 1b: Idle detection is enabled	0x0
18 AUTO_SYNC_EN	RW	LIN baud rate auto sync enable bit. 0b: Disabled 1b: Enabled	0x0
19 DEBUG_EN	RW	Debug enable. When enabled, hardware will not clean the receiver FIFO automatically.	0x0
31:20 RSVD_31_20	RO	Reserved	0x0

36.3.10 Autoflow Control Register (UART_AFCR)

36.3.10.1 Offset

Register	Offset
UART_AFCR	0x10

36.3.10.2 Function

This register is used to control the autoflow function.

36.3.10.3 Fields

Field	Access	Description	Reset value
0 RSVD_0	RO	Reserved	0x0
1 RTS	RW	Request to send. This is used to directly control the Request to Send output. The Request to Send output is used to inform the modem or data set that the UART is ready to exchange data. When UART_AFCR[AFCE] is set to 0, the RTS signal is set low by programming UART_AFCR[RTS] to high. When auto flow control is enabled and FIFO is enabled, the RTS output is controlled in the same way, but is also gated with the receiver FIFO threshold trigger. RTS is inactive high when above the threshold. 0b: It represents RTS is deasserted (logic 1). 1b: It represents RTS is asserted (logic 0).	0x0
3:2 RSVD_3_2	RO	Reserved	0x0
4 LB	RW	Loop back bit. This is used to put the UART into a diagnostic mode for test purposes.	0x0
5 AFCE	RW	Auto flow control enable bit 0b: Auto Flow Control Mode disabled. 1b: Auto Flow Control Mode enabled.	0x0
31:6 RSVD_31_6	RO	Reserved	0x0

36.3.11 Line Status Register (UART_LSR)

36.3.11.1 Offset

Register	Offset
UART_LSR	0x14

36.3.11.2 Function

This register is used to indicate line status, reading this register will clear the status.

36.3.11.3 Fields

Field	Access	Description	Reset value
0 DR	RO	Data ready bit. This is used to indicate that the receiver contains at least one character in the RBR or the receiver FIFO. 0b: No data ready 1b: Data ready	0x0
1 OE	RO	Overrun error bit. This is used to indicate the occurrence of an overrun error. This occurs if a new data character was received before the previous data was read. 0b: No overrun error 1b: Overrun error	0x0
2 PE	RO	Parity error bit. This is used to indicate the occurrence of a parity	0x0

Field	Access	Description	Reset value
		error in the receiver if the Parity Enable (PEN) bit (UART_LCR[3]) is set. 0b: No parity error 1b: Parity error	
3 FE	RO	Framing error bit. This is used to indicate the occurrence of a framing error in the receiver. A framing error occurs when the receiver does not detect a valid STOP bit in the received data. 0b: No frame error 1b: Frame error	0x0
4 BI	RO	Break interrupt bit. This is used to indicate the detection of a break sequence on the serial input data. 0b: No break sequence is detected. 1b: Break sequence is detected.	0x0
5 THRE	RO	Transmit holding register empty bit. This bit is set whenever data is sent and no new data has been written to the THR or TX FIFO. This also causes a THRE Interrupt to occur, if the THRE Interrupt is enabled. 0b: THR has data 1b: THR is empty The reset value is 1.	0x1
6 TEMT	RO	Transmitter empty bit. 0b: Transmitter has data. 1b: Transmitter is empty. The reset value is 1.	0x1
7 RFE	RO	Receiver FIFO error bit. This is used to indicate if there is at least one parity error, framing error, or break indication in the FIFO. 0b: No error in RX FIFO. 1b: Error in RX FIFO.	0x0
8 ADDR_RCVD	RO	Address received bit. 0b: Indicates that the character is data. 1b: Indicates that the character is an address.	0x0
9 ASYNC_INT	RO	Asynchronous interrupt flag. 0b: No Asynchronous interrupt. 1b: Asynchronous interrupt happens.	0x0
10 SYNC_FIELD_ERROR	RO	Sync field error in LIN mode. 0b: No sync field error. 1b: Sync field error happens.	0x0
11 PID_ERR	RO	PID error in LIN mode. Read clear 0b: No PID error. 1b: PID error happens.	0x0
12 CHECKSUM_ERROR	RO	Checksum error flag in LIN mode. Read clear 0b: No checksum error. 1b: Checksum error happens.	0x0
13 TO_ERROR	RO	Timeout error in LIN mode. Read clear 0b: No timeout error. 1b: Timeout error happens.	0x0
14 HEADER_OP	RO	Header operation done flag in LIN mode. 0b: Header operation hasn't done.	0x0

Field	Access	Description	Reset value
_DONE		1b: Header operation has done. NOTE: When HEADER_OP_DONE flag was set, SYNC_FIELD_ERROR, PID_ERR and TO_ERROR flags may also be set.	
15 RSP_OP_DONE	RO	Response operation done flag in LIN mode. 0b: Response operation hasn't done. 1b: Response operation has done. NOTE: When RSP_OP_DONE flag was set, CHECKSUM_ERROR and TO_ERROR flags may also be set.	0x0
31:16 RSVD_31_16	RO	Reserved	0x0

36.3.12 Modem Status Register (UART_MSR)

36.3.12.1 Offset

Register	Offset
UART_MSR	0x18

36.3.12.2 Function

This register is used to indicate the modem status.

36.3.12.3 Fields

Field	Access	Description	Reset value
3:0 RSVD_3_0	RO	Reserved	0x0
4 CTS	RO	Clear to Send. This is used to indicate the current state of the modem control line CTS_n. This bit is the complement of CTS_n. 0b: CTS_n input is deasserted (logic 1) 1b: CTS_n input is asserted (logic 0)	0x0
31:5 RSVD_31_5	RO	Reserved	0x0

36.3.13 UART Status Register (UART_USR)

36.3.13.1 Offset

Register	Offset
UART_USR	0x7C

36.3.13.2 Function

The register is used for indicating the UART transfer status.

36.3.13.3 Fields

Field	Access	Description	Reset value

Field	Access	Description	Reset value
0 BUSY	RO	UART busy. 0b: UART is idle or inactive. 1b: UART is busy (actively transferring data).	0x0
31:1 RSVD_31_1	RO	Reserved	0x0

36.3.14 Fraction Divisor Register (UART_FD)

36.3.14.1 Offset

Register	Offset
UART_FD	0xC0

36.3.14.2 Function

This is the fraction part of divisor register.

36.3.14.3 Fields

Field	Access	Description	Reset value
3:0 FD	RW	Fractional divisor. Fractional value is determined by (Fraction Divisor)/16.	0x0
31:4 RSVD_31_4	RO	Reserved	0x0

36.3.15 Receive Address Register (UART_RAR)

36.3.15.1 Offset

Register	Offset
UART_RAR	0xC4

36.3.15.2 Function

This is the receiver address register.

36.3.15.3 Fields

Field	Access	Description	Reset value
7:0 RAR	RW	This is an address matching register during receive mode. If the 9th bit is set in the incoming character then the remaining 8-bits will be checked against this register value. If the match happens then subsequent characters with 9th bit set to 0 will be treated as data byte until the next address byte is received. This register is applicable only in 9 bits mode and UART_LCR_EXT[ADDR_MATCH] is set to 1.	0x0
31:8 RSVD_31_8	RO	Reserved	0x0

36.3.16 Transmit Address Register (UART_TAR)

36.3.16.1 Offset

Register	Offset
UART_TAR	0xC8

36.3.16.2 Function

This is the transmit address register.

36.3.16.3 Fields

Field	Access	Description	Reset value
7:0 TAR	RW	This is an address matching register during transmit mode. UART sends 9 bits character with 9th bit set to 1 and remaining 8 bits address will be sent from this register provided UART_LCR_EXT[SEND_ADDR] bit is set to 1. This register is used only to send the address. The normal data should be sent by programming THR register.	0x0
31:8 RSVD_31_8	RO	Reserved	0x0

36.3.17 Line Extended Control Register (UART_LCR_EXT)

36.3.17.1 Offset

Register	Offset
UART_LCR_EXT	0xCC

36.3.17.2 Function

This is the line extended control register.

36.3.17.3 Fields

Field	Access	Description	Reset value
0 DLS_E	RW	Extension for DLS. This bit is used to enable 9-bit data for transmit and receive transfers. 0b: Number of data bits selected by DLS. 1b: 9 bits per character.	0x0
1 ADDR_MATCH	RW	Address Match Mode. This bit is used to enable the address match feature during reception. 0b: Normal mode. UART will start to receive the data and 9-bit character will be formed and written into the receive Rx FIFO. The user is responsible to read the data and differentiate address and data. 1b: Address match mode. UART will wait until the	0x0

Field	Access	Description	Reset value
		incoming character with 9th bit is set to 1. Further, check is made to see if the address matches with what is programmed in "Receive Address Register". If match is found, then subsequent characters will be treated as valid receiving data.	
2 SEND_ADDR	RW	Send address control bit. This bit is used as a control knob for the user to determine when to send the address during transmit mode. 0b: 9-bit character will be transmitted with 9th bit set to 0 and the remaining 8 bits will be taken from the TxFIFO which is programmed through 8-bit wide THR register. 1b: 9-bit character will be transmitted with 9th bit set to 1 and the remaining 8 bits will match to what is being programmed in "Transmit Address Register". UART clears the SEND_ADDR bit after the address character starts transmitting on the UART line.	0x0
3 TRANSMIST_MODE	RW	Transmit mode control bit. This bit is used to control the type of transmit mode during 9-bit data transfers. 0b: The user needs to program the address into UART_TAR and data into the UART_THR. 1b: In this mode of operation, UART_THR is 9-bit wide. Address: 9th bit is set to 1, Data: 9th bit is set to 0. NOTE UART_TAR is not applicable in this mode of operation.	0x0
31:4 RSVD_31_4	RO	Reserved	0x0

36.3.18 LIN Mode Control Register (UART_LIN_CTL)

36.3.18.1 Offset

Register	Offset
UART_LIN_CTL	0x100

36.3.18.2 Function

This is the control register.

36.3.18.3 Fields

Field	Access	Description	Reset value
0 LIN_MODE	RW	LIN mode enable 0b: The device acts as a UART. 1b: The device acts as a LIN.	0x0
1 MASTER_MODE	RW	Master mode. 0b: The device is a slave.	0x0

Field	Access	Description	Reset value
		1b: The device is a master.	
2 HEADER_OP_START	RW	Header operation start trigger This bit will be cleared when header operation is done. When master mode bit is set, the device is set as a master, then it will send a header when this bit is set. When master mode bit is cleared, it will start to receive a header when this bit is set.	0x0
3 RSP_OP_START	RW	Response operation start trigger This bit will be cleared when response operation is done.	0x0
4 RSP_DIR	RW	Response operation direction in LIN mode 0b: Receive response. 1b: Send response.	0x0
31:5 RSVD_31_5	RO	Reserved	0x0

36.3.19 LIN Response Length Register (UART_LIN_RSP_LENGTH)

36.3.19.1 Offset

Register	Offset
UART_LIN_RSP_LENGTH	0x104

36.3.19.2 Function

This is the LIN response length register.

36.3.19.3 Fields

Field	Access	Description	Reset value
3:0 RSP_LENGTH	RW	LIN response length register, this length does not include checksum field.	0x0
31:4 RSVD_31_4	RO	Reserved	0x0

36.3.20 LIN ID Register (UART_LIN_ID_VALUE)

36.3.20.1 Offset

Register	Offset
UART_LIN_ID_VALUE	0x108

36.3.20.2 Function

This is the LIN ID register.

36.3.20.3 Fields

Field	Access	Description	Reset value

Field	Access	Description	Reset value
5:0 ID	RW	LIN ID register When the header is received as a slave, the received ID is stored in this register. When the device is set as a master, the user could write only 6 bits ID in this register, hardware will calculate the parity. PID is composed of a 6-bit ID and a 2-bit parity.	0x0
31:6 RSVD_31_6	RO	Reserved	0x0

36.3.21 LIN Checksum Register (UART_LIN_CHECKSUM)

36.3.21.1 Offset

Register	Offset
UART_LIN_CHECKSUM	0x10C

36.3.21.2 Function

This is the LIN checksum register.

36.3.21.3 Fields

Field	Access	Description	Reset value
0 CHECKSUM_TYPE	RW	LIN checksum type 0b: Classic checksum 1b: Enhanced checksum	0x0
7:1 RSVD_7_1	RO	Reserved	0x0
31:8 RSVD_31_8	RO	Reserved	0x0

36.3.22 LIN Delimiter Field Length (UART_LIN_DEL_LENGTH)

36.3.22.1 Offset

Register	Offset
UART_LIN_DEL_LENGTH	0x110

36.3.22.2 Function

This is the LIN delimiter field length.

36.3.22.3 Fields

Field	Access	Description	Reset value
2:0 LIN_DEL_LENGTH	RW	LIN delimiter field length	0x0
31:3 RSVD_31_3	RO	Reserved	0x0

36.3.23 LIN PID Filter 0 (UART_LIN_PID_FILTER_0)

36.3.23.1 Offset

Register	Offset
UART_LIN_PID_FILTER_0	0x114

36.3.23.2 Function

This is the LIN PID filter 0.

36.3.23.3 Fields

Field	Access	Description	Reset value
5:0 FID0	RW	Filter PID0	0x0
7:6 RSVD_7_6	RO	Reserved	0x0
13:8 FID1	RW	Filter PID1	0x0
15:14 RSVD_15_14	RO	Reserved	0x0
21:16 FID2	RW	Filter PID2	0x0
23:22 RSVD_23_22	RO	Reserved	0x0
29:24 FID3	RW	Filter PID3	0x0
31:30 RSVD_31_30	RO	Reserved	0x0

36.3.24 LIN PID Filter 1 (UART_LIN_PID_FILTER_1)

36.3.24.1 Offset

Register	Offset
UART_LIN_PID_FILTER_1	0x118

36.3.24.2 Function

This is the LIN PID filter 1.

36.3.24.3 Fields

Field	Access	Description	Reset value
5:0 FID4	RW	Filter PID4	0x0
7:6 RSVD_7_6	RO	Reserved	0x0
13:8 FID5	RW	Filter PID5	0x0
15:14 RSVD_15_14	RO	Reserved	0x0

Field	Access	Description	Reset value
21:16 FID6	RW	Filter PID6	0x0
23:22 RSVD_23_22	RO	Reserved	0x0
29:24 FID7	RW	Filter PID7	0x0
31:30 RSVD_31_30	RO	Reserved	0x0

36.3.25 LIN PID Filter 2 (UART_LIN_PID_FILTER_2)

36.3.25.1 Offset

Register	Offset
UART_LIN_PID_FILTER_2	0x11C

36.3.25.2 Function

This is the LIN PID filter 2.

36.3.25.3 Fields

Field	Access	Description	Reset value
5:0 FID8	RW	Filter PID8	0x0
7:6 RSVD_7_6	RO	Reserved	0x0
13:8 FID9	RW	Filter PID9	0x0
15:14 RSVD_15_14	RO	Reserved	0x0
21:16 FID10	RW	Filter PID10	0x0
23:22 RSVD_23_22	RO	Reserved	0x0
29:24 FID11	RW	Filter PID11	0x0
31:30 RSVD_31_30	RO	Reserved	0x0

36.3.26 LIN PID Filter 3 (UART_LIN_PID_FILTER_3)

36.3.26.1 Offset

Register	Offset
UART_LIN_PID_FILTER_3	0x120

36.3.26.2 Function

This is the LIN PID filter 3.

36.3.26.3 Fields

Field	Access	Description	Reset value
5:0 FID12	RW	Filter PID12	0x0
7:6 RSVD_7_6	RO	Reserved	0x0
13:8 FID13	RW	Filter PID13	0x0
15:14 RSVD_15_14	RO	Reserved	0x0
21:16 FID14	RW	Filter PID14	0x0
23:22 RSVD_23_22	RO	Reserved	0x0
29:24 FID15	RW	Filter PID15	0x0
31:30 RSVD_31_30	RO	Reserved	0x0

36.3.27 LIN PID Filter Vague (UART_LIN_PID_FILTER_VAGUE)

36.3.27.1 Offset

Register	Offset
UART_LIN_PID_FILTER_VAGUE	0x124

36.3.27.2 Function

This is the LIN PID filter vague.

36.3.27.3 Fields

Field	Access	Description	Reset value
5:0 FID_VAGUE	RW	Vague filter PID	0x0
31:6 RSVD_31_6	RO	Reserved	0x0

36.3.28 LIN PID Filter Control Register (UART_LIN_PID_FILTER_CTRL)

36.3.28.1 Offset

Register	Offset
UART_LIN_PID_FILTER_CTRL	0x128

36.3.28.2 Function

This is the LIN PID filter control register.

36.3.28.3 Fields

Field	Access	Description	Reset value
16:0 FID_MASK	RW	The enable bit of the filter ID 0-15 and the vague filter. Bit 0-15 controls filter 0-15. Bit 16 controls the vague filter. The meaning of each bit value is: 0b: This filter is disabled. 1b: This filter is enabled.	0x0
22:17 FID_VAGUE_MASK	RW	The enable bit of the filter vague ID If the vague filter is enabled, this field can control which bits in the vague filter are valid. 0b: The corresponding bit in the vague filter does not take effect. 1b: The corresponding bit in the vague filter is valid.	0x0
23 LIN_FILTER_EN	RW	The enable bit of the filter	0x0
31:24 RSVD_31_24	RO	Reserved	0x0

36.4 UART function description

This section provides the functional information of the UART module.

36.4.1 Transmit programming

The flow diagram in the following figure shows the programming sequence for setting up transmission.

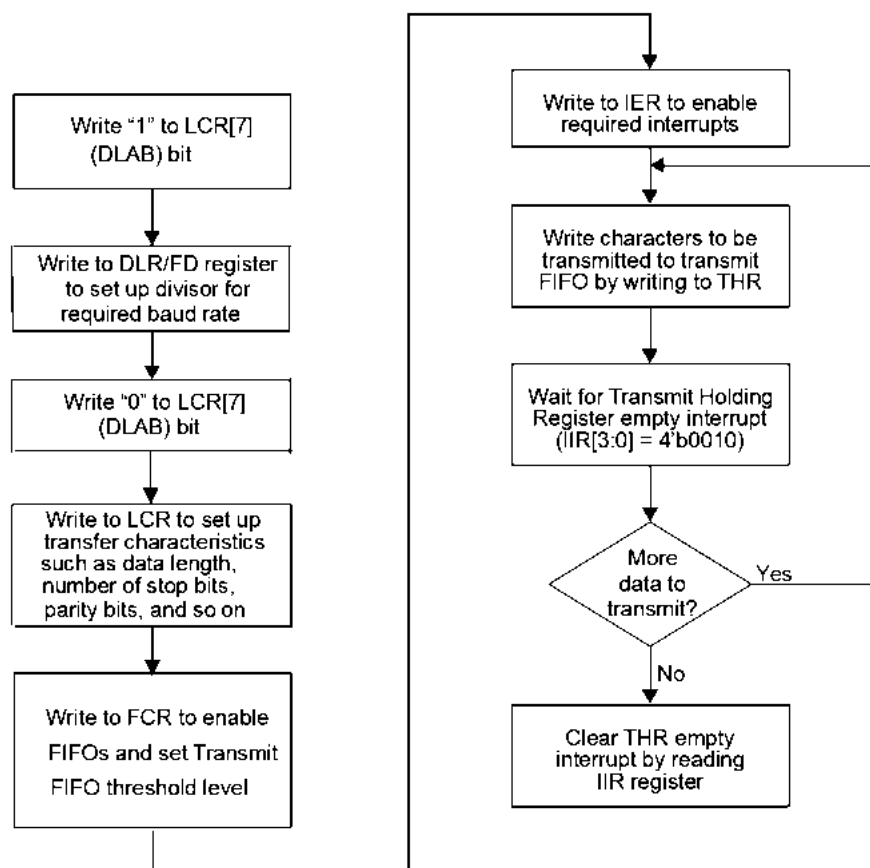


Figure 36-3 UART Transmit Programming Example

36.4.2 Receive programming

The flow diagram in the following Figure shows the programming sequence for setting up reception.

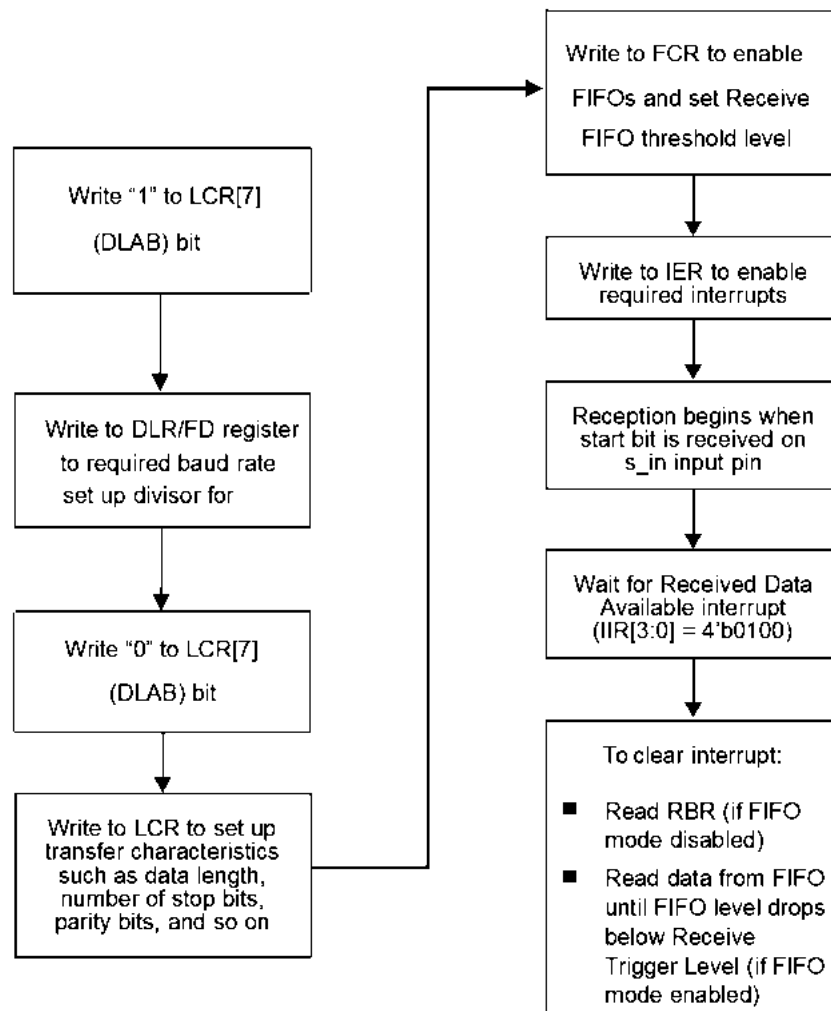


Figure 36-4 UART Receive Programming Example

36.4.3 Tx/Rx data format

The module provides several register bits to control the transmit or receive character format as below:

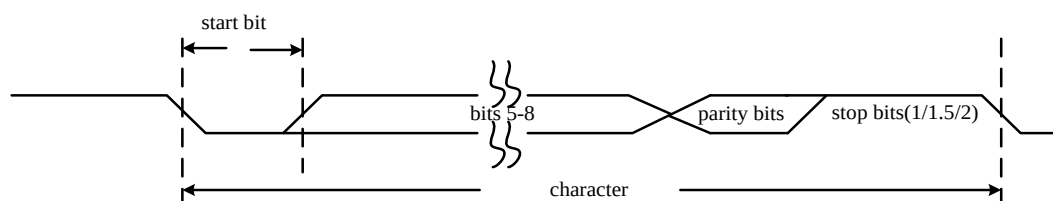


Figure 36-5 8 bits UART Character format

A UART character includes four-bit fields:

- 1 start bit (0)

- 5-8 bits data
- 1 parity bits
- 1/1.5/2 stop bits.

The user can use the register UART_LCR to control the character format. UART_LCR[DLS] controls the data length from 5 to 8 bits for the user to transmit or receive. UART_LCR[PEN] enables and disables parity generation and detection in transmitted and received serial characters respectively. UART_LCR[EPS] selects between even and odd parity when the parity is enabled (PEN set to 1).

- If set to 1, an even number of logic 1 is transmitted or checked.
- If set to 0, an odd number of logic 1 is transmitted or checked.

UART_LCR[STOP] is used to select the number of stop bits per character that the peripheral transmits and receives.

- If set to 0, one stop bit is transmitted in the serial data.
- If set to 1 and data length is set to 5 bits, then the stop bits are set to 1.5 bits. Otherwise, the stop bits are set to 2 bits.

36.4.4 Baud rate clock generation

The output baud rate clock is determined by the module function clock and the divisor. The output baud rate clock is equal to the function clock frequency divided by sixteen times the value of baud rate divisor. The formula is as below:

$$\text{baud rate clock} = (\text{function clock freq}) / (16 * \text{divisor})$$

The function clock comes from SoC. The frequency and clock source can be configured in the PARCC module. For more details of the configuration, please refer to PARCC.

The module supports the integer divisor and the fraction divisor.

If the required baud rate is very slow, and the generated baud rate error is acceptable with only an integer divisor, the user can only use the integer divisor. Otherwise, the user needs to configure the fraction divisor.

The module provides registers UART_DLH and UART_DLL to configure the integer divisor for the baud rate, and the divisor is equal to {UART_DLH, UART_DLL}.

If UART_DLH and UART_DLL are all set to 0, it means UART is disabled.

The register UART_FD[3:0] is used to configure the fraction divisor for the baud rate. The fraction divisor can be calculated as below:

$$f\_div = \text{UART\_FD}[3:0] / 16$$

so the calculated divisor = {UART_DLH, UART_DLL} + f_div. The user can use the formula below to calculate how to configure the integer part and the fraction part.

$$\text{divisor} = (\text{function clock freq}) / (16 * \text{desired baud rate})$$

36.4.5 9-Bit data transfer

The module can also be configured to have 9-bit character, the 9-bit character format is shown as below, and the 9th bit is between 8th bit and parity bits:

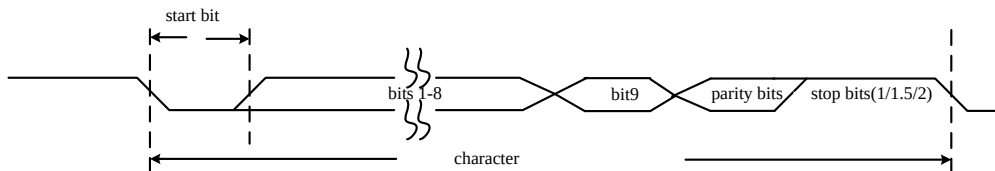


Figure 36-6 9 bits UART Character Format

By enabling the 9-bit data transfer mode, UART can be used in multi-drop systems where one master is connected to multiple slaves in a system. The master communicates with one of the slaves. When the master wants to transfer a block of data to a slave, it first sends an address byte to identify the target slave. The address byte is differentiated from the data byte by setting the 9th bit of the data byte to 1. If the 9th bit is set to 1, then the character

represents an address byte. All the slave systems compare the address byte with their own address and only the target slave (in which the address has matched) is enabled to receive data from the master. The master then starts transmitting data bytes to the target slave. The non-addressed slave systems ignore the incoming data until a new address byte is received. The user can configure UART_LCR_EXT[DLS_E] to enable the 9th bit data for transmitting and receiving transfers. For more detailed configuration, see the related register description.

36.4.6 Tx&Rx FIFO operation

The module includes two separate FIFO for RX and TX, and the FIFO depth is 16. Users can configure UART_FCR[FIFOE] to enable the TX FIFO or RX FIFO. If not enabled, UART can only store one character in TXR or RXR. When enabled, FIFO will be enabled, users can fill the TX FIFO by writing TXR and read the RX FIFO by reading RXR.

UART_FCR[TET] and UART_FCR[RCVR] are used to set the threshold of TX FIFO and RX FIFO, at which the interrupt or DMA request is generated. The FIFO will be reset whenever UART_FCR[FIFOE] is changed. Users can also reset the FIFO by setting UART_FCR[RFIFOR] and UART_FCR[XFIFOR].

36.4.7 DMA and interrupt

There are several interrupt sources in the module. Users can configure the Interrupt Enable Register to enable the interrupt. They can also enable Received Data Available, transmitter data empty, receiver status interrupt based on their need.

- Receive Data Available
 - In FIFO mode, the interrupt will be triggered when the RX FIFO reaches the threshold.
 - In none-FIFO mode, the interrupt will be triggered when there is data in UART_RBR.
- Transmit Data Empty
 - In FIFO mode, the interrupt will be triggered when the transmit FIFO is below the threshold.
 - In a none-FIFO mode, the interrupt will be triggered when the THR data is sent out.
- Receiver Status Error
 - Overrun error. It is for indicating the occurrence of an overrun error. The overrun error occurs if a new data character was received before the previous data was read.
 - Parity error. It is for indicating the occurrence of a parity error in the receiver if the Parity Enable (PEN) bit (UART_LCR[3]) is set.
 - Frame error. This is for indicating the occurrence of a framing error in the receiver. A framing error occurs when the receiver does not detect a valid STOP bit in the received data.
 - Break interrupt. This is for indicating the detection of a break sequence on the serial input data.
- Character Timeout:
 - This interrupt will be asserted when the received FIFO is not empty, and the line is idle for some time. The idle character length is configurable.
- LIN header operation done
 - The header operation done interrupt will be asserted when a header is received or transmitted, or the header operation is timed out.
- LIN response operation done
 - The response operation done interrupt will be asserted when a response is received or transmitted, or the response operation is timed out.
- Busy detect interrupt

- Busy detect interrupt is enabled by default and cannot be disabled. Write access to UART_LCR register is not allowed when UART in TX or RX process, which means there is data sending out or data receiving or when the RX FIFO is not empty, if it is written, an interrupt will be asserted and the IID will be set to 4'b0111; if users want to change the LCR register, it should guarantee that the RX FIFO is empty and there is no data transmitting or receiving.

The module also supports two DMA signals dma_req and dma_ack for TX and RX individually. The DMA request will be asserted when there is data to be read or this is no other data to transmit.

36.5 LIN function description

36.5.1 Frame auto send and receive

The LIN frame contains two parts: header and response. The header consists of a 13-bit break field, an 8-bit sync field, and an 8-bit PID field. The header is always sent by the master. The response consists of a data field and checksum field, and the master or the slave can send the response.

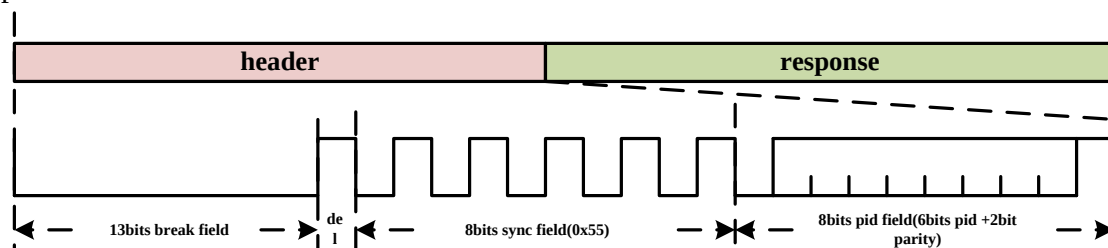


Figure 36-7 LIN Format

36.5.1.1 Header send

The header can only be sent by a master. In the controller, the user only needs to configure 6-bit ID and the break field and trigger the header operation, then hardware will respond to send out the full header including calculation of the PID parity.

Table 36-3 Header send process

Step	Software	Hardware
1	—Set the baud rate —Enable interrupt —Set the ID(6 bits) —Set the break field length, set the delimiter length —Set the module to be LIN & master mode	—Wait for the header operation start bits to be set
2	—Set header operation start bit and wait for the interrupt	—Send break —Send break delimiter —Send sync field —Calculate the PID parity and send the 8 bits PID —Set header operation done —Assert the interrupt
3	—Handle the interrupt, execute the interrupt function	

36.5.1.2 Header receive

The header will be received by a slave after the master sends it out. The slave will first detect the break field and adjust the baud rate after the sync field is received, then the slave will

receive the PID field and check the parity.

Table 36-4 Header receive process

Step	Software	Hardware
1	—Set the baud rate —Set the interrupt —Set the module to be LIN & slave mode	—Wait for the header operation start bits to be set
2	—Set header operation start bit and wait for interrupt	—Receive break field —Receive sync field, assert timeout if the sync field was not received at the given time —Calculate the baud rate and update —Receive PID field —Calculate PID parity, set the PID error bit if check fails —Assert interrupt after header operation is done if filter passes
3	—Handle the interrupt and execute the interrupt function	

36.5.1.3 Response send

Master or slave can send responses. Hardware will be responsible for sending the data fields by byte and calculating the checksum field according to the checksum type and the PID value that the software sets.

Table 36-5 Response send process

Step	Software	Hardware
1	—Set the baud rate —Set the response length —Set the interrupt —Set the checksum type —Set PID value or use the PID of header —Fill the transmit FIFO with the response data —Set the module to be LIN —Set response direction to send —Set response operation start	—Wait for response operation start trigger
2	—Wait for interrupt	—Send the data in TX FIFO one by one according to the response length —Get the PID from the register —Calculate the checksum according to the checksum type —Send the checksum —Set the response operation done flag and assert the interrupt
3	—Handle the interrupt and execute the interrupt function	

36.5.1.4 Response receive

Master or slave can receive responses. Hardware will be responsible for receiving the data and storing the received data into FIFO. When the received data reaches response length -1, the hardware will calculate the checksum according to the checksum type. If the received

checksum matches the checksum calculated, the response done operation flag will be set and interrupted will be asserted.

Table 36-6 Response receive process

Step	Software	Hardware
1	—Set the baud rate —Set the response length —Set the interrupt —Set checksum type —Set PID value or use the PID of header —Set the module to be LIN —Set response direction to receive —Set response operation start	—Wait for response operation start trigger
2	—Wait for interrupt	—Wait for the start bit —Receive the first byte —Receive the other bytes until the received FIFO reaches the configured response length —Calculate the checksum —Receive the checksum byte, compare the checksum byte with the calculated result —Assert an interrupt
3	—Handle the interrupt and execute the interrupt function	

36.5.2 Checksum calculation and check

LIN in this module supports Classic Checksum and Enhanced Checksum calculation and checks both of them for response transmission and reception. Users can use `UART_LIN_CHECKSUM[CHECKSUM_TYPE]` to select which type of checksum to use.

- For transmitting the response
 - Users should first configure the LIN response length that does not include checksum with the register `UART_LIN_RSP_LENGTH`. If using the Enhanced Checksum, users should also configure the PID before starting the transmission. Hardware will respond to calculate the checksum after sending out the last response byte and sending out the checksum after the response.
- For receiving the response
 - Users should also guarantee that PID in `UART_LIN_ID_VALUE` is the right one, after configuring the response type register to select the checksum type. Users can use the saved one when receiving or transmitting the header, or use a new PID.

36.5.3 PID parity calculation and check

The module supports LIN PID parity calculation and the check function. When sending the header, users only need to fill the `UART_LIN_ID_VALUE[5:0]`. Then hardware will calculate the two parity bits and fill into the header.

When receiving the header as a slave, hardware will calculate the parity bits and compare the calculated result with the parity bits that are received. No matter the parity bits are right or not, all the received 8 bits of PID will be stored in the `UART_LIN_ID_VALUE` register. Software can decide how to use the received data.

36.5.4 PID filter

When used as a slave, the module supports a PID filter list of 16 and a vague filter PID. These 17 filter PID can be enabled and disabled by setting LIN_PID_FILTER_CTRL[FID_MASK]. When enabled, hardware will compare the received PID with the PID in filter list. If the received PID is in the PID filter list, hardware will assert an interrupt to wakeup core. If the PID is not in the filter list, hardware will go on to receive the next header until the header PID is in the filter list.

When enabling the vague PID function, users also need to configure the LIN_PID_FILTER_CTRL[FID_VAGUE_MASK]. When comparing the vague PID, hardware will only compare the bits that are enabled in LIN_PID_FILTER_CTRL[FID_VAGUE_MASK].

36.5.5 Baud rate auto sync

In LIN protocol, the sync field is used to correct the baud rate error between master and slave. The module provides a function to sync the baud rate by hardware by setting UART_LCR[AUTO_SYNC_EN]. Suppose the auto sync function is used, the user should first set the DLH and DLL registers to the desired baud rate. Hardware will calculate the real baud rate based on the received sync field, then write it back to DLH and DLL registers. The master and the slave are synced when the PID field is received after sync.

Chapter 37

Serial Peripheral Interface (SPI)

37.1 Introduction

This chapter defines the Serial Peripheral Interface (SPI).

37.1.1 Features

The SPI supports the following key features:

- Configurable master mode and slave mode
- Full duplex synchronous transfers
- The depth of 4 entries that separates RX FIFO and TX FIFO
- Up to 2 slave peripheral select signals (SS) supported when in the master mode
- DMA request support for RX FIFO and TX FIFO separately
- Multiple interrupt conditions:
 - TX FIFO empty (txe)
 - TX FIFO overflow (txo)
 - RX FIFO full (rxf)
 - RX FIFO overflow (rxo)
 - RX FIFO underflow (rxu)
- Serial clock (SCK) with programmable polarity and phase
- Programmable data frame size from 4 to 32

37.1.2 Block diagram

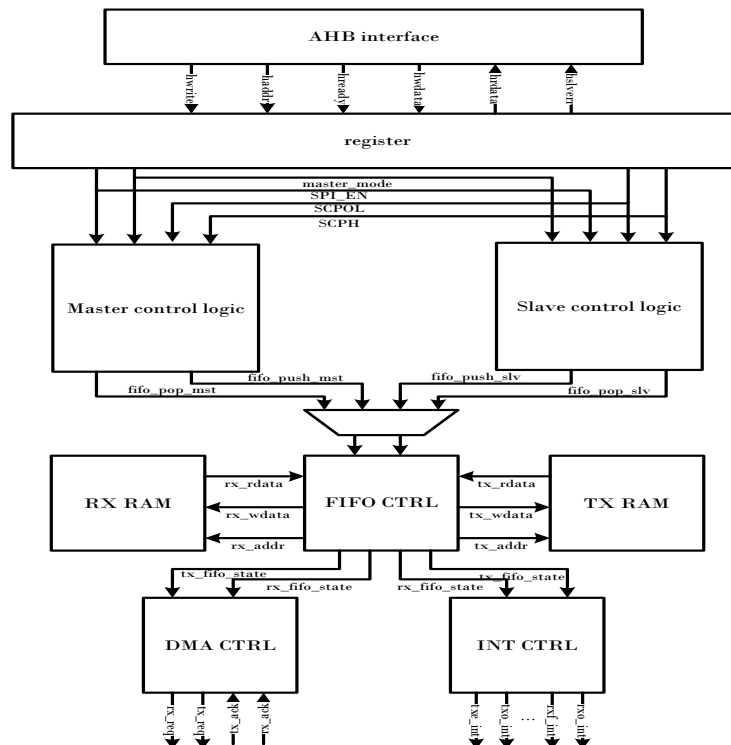


Figure 37-1 SPI block diagram

The above figure is a block diagram of the Serial Peripheral Interface (SPI) module.

37.1.3 Instantiation Information

The following table lists the SPI instantiation information for each chip in the product series.

Configuration	K144MC	K146MC	K148MC
Instance(s)	SPI0	SPI0	SPI0
	SPI1	SPI1	SPI1
	SPI2	SPI2	SPI2
	SPI3	SPI3	SPI3

37.2 Signal description

The SPI works in a master-slave mode, and the SPI device consists of at least four wires as below.

Table 37-1 Signal descriptions

Wires	Descriptions
SPI_SOUT	Output
SPI_SIN	Input
SPI_SCK	Clock signal, generated by the master device
SPI_PCS0 SPI_PCS1	Slave select signals controlled by the master device.

Note: PCS0 and PCS1 lines should be pulled up in case of configuring slave device without master device in SPI bus.

37.3 Memory map and register definition

This section provides the memory map and register definitions of the SPI module.

37.3.1 Memory map

SPI0 base address: 0x4002C000

SPI1 base address: 0x4002D000

SPI2 base address: 0x40030000

SPI3 base address: 0x40031000

Table 37-2 SPI memory map

Offset	Register	Width (In bits)	Reset Value
0x0	Control Register 0 (SPI_CTRLR0)	32	0x00270000
0x4	Control Register 1 (SPI_CTRLR1)	32	0x00000000
0x8	SPI Enable Register (SPI_SSENr)	32	0x00000000
0xC	Baud Rate Select Register (SPI_BAUDR)	32	0x00000000
0x10	FIFO Threshold Level Register (SPI_FTLR)	32	0x00000000
0x14	FIFO Level Register (SPI_FLR)	32	0x00000000
0x18	Status Register	32	0x00000006

Offset	Register	Width (In bits)	Reset Value
	(SPI_SR)		
0x1C	Interrupt Enable Register (SPI_IER)	32	0x0000003F
0x20	Interrupt Status Register (SPI_ISR)	32	0x00000000
0x24	Transmit FIFO Overflow Interrupt Clear Register (SPI_TXOICR)	32	0x00000000
0x28	Receive FIFO Overflow Interrupt Clear Register (SPI_RXOICR)	32	0x00000000
0x2c	Receive FIFO Underflow Interrupt Clear Register (SPI_RXUICR)	32	0x00000000
0x34	Interrupt Clear Register (SPI_ICR)	32	0x00000000
0x38	DMA Control Register (SPI_DMACR)	32	0x00000000
0x3C	DMA Transmit Data Level (SPI_DMATDLR)	32	0x00000000
0x40	DMA Receive Data Level (SPI_DMARDLR)	32	0x00000000
0x60	Data Register (SPI_DR_LOW)	32	0x00000000
0xEC	Data Register (SPI_DR_HIGH)	32	0x00000000

37.3.2 Control Register 0 (SPI_CTRLR0)

37.3.2.1 Offset

Register	Offset
SPI_CTRLR0	0x0

37.3.2.2 Function

This register controls the serial data transfer. The user cannot write to this register when the SPI is enabled.

37.3.2.3 Fields

Field	Access	Description	Reset Value
5:0 RSVD_5_0	RO	Reserved	0x0
6 SCPH	RW	0b: Serial clock toggles in the middle of first data bit 1b: Serial clock toggles at the start of first data bit	0x0
7 SCPOL	RW	0b: Inactive state of serial clock is low 1b: Inactive state of serial clock is high	0x0
9:8 TMOD	RW	Select the mode of the transfer for serial communication. 00b: Transmit & Receive 01b: Transmit Only 10b: Receive Only	0x0

Field	Access	Description	Reset Value
		11b: Reserved	
15:10 RSVD_15_10	RO	Reserved	0x0
20:16 DFS_32	RW	They are used to select the data frame size in 32-bit mode. When the data frame size is programmed to be less than 32 bits, the received data is automatically right-justified by the receive logic, with the upper bits of the receive FIFO zero-padded. 00000b: Reserved 00001b: Reserved 00010b: Reserved 00011b: 4-bit serial data transfer 00100b: 5-bit serial data transfer 00101b: 6-bit serial data transfer ... 11110b: 31-bit serial data transfer 11111b: 32-bit serial data transfer	0x7
21 MST_MODE	RW	0b: Slave mode enable 1b: Master mode enable	0x1
31:22 RSVD_31_22	RO	Reserved	0x0

37.3.3 Control Register 1 (SPI_CTRLR1)

37.3.3.1 Offset

Register	Offset
SPI_CTRLR1	0x4

37.3.3.2 Function

This register exists only when the SPI is configured as a master device. Control register 1 controls the end of serial transfers when in the receive-only mode.

37.3.3.3 Fields

Field	Access	Description	Reset Value
15:0 NDF	RW	When TMOD = 10b, this register field sets the number of data frames to be continuously received by the SPI. The actual number of frames that has been transferred is NDF + 1. Note: An arbitrary data needs to be written to the TX FIFO to start receiving.	0x0
31:16 RSVD_31_16	RO	Reserved	0x0

37.3.4 SPI Enable Register (SPI_SSENr)

37.3.4.1 Offset

Register	Offset
SPI_SSENr	0x8

37.3.4.2 Function

This register enables and disables the SPI, and selects SPI slave.

37.3.4.3 Fields

Field	Access	Description	Reset Value
0 SPI_EN	RW	Enable and disable all SPI operations. 0b: Disable 1b: Enable	0x0
15:1 RSVD_15_1	RO	Reserved	0x0
17:16 SER	RW	Each bit in this register corresponds to a slave select line of the master SPI device. 00b: Neither of the PCS0 and PCS1 is selected. 01b: Only PCS0 is selected. 10b: Only PCS1 is selected. 11b: Both PCS0 and PCS1 are selected. NOTE - When SPI is a slave, these two bits are reserved, and only PCS0 is valid. - When SPI is a master, this field must be configured with at least one PCS; otherwise, SPI will not start data transfer.	0x0
31:18 RSVD_31_18	RO	Reserved	0x0

37.3.5 Baud Rate Select Register (SPI_BAUDR)

37.3.5.1 Offset

Register	Offset
SPI_BAUDR	0xC

37.3.5.2 Function

The register derives the frequency of the serial clock that regulates the data transfer.

37.3.5.3 Fields

Field	Access	Description	Reset Value
15:0 SCKDV	RW	The frequency of the SPI_SCK is derived from the following equation: $F_{spi\_sck} = F_{spi\_clk} / SCKDV$, where SCKDV is any even value between 2 and 65534 and the spi_clk performs as functional clock of the SPI device.	0x0
31:16	RO	Reserved	0x0

Field	Access	Description	Reset Value
RSVD_31_16			

37.3.6 FIFO Threshold Level Register (SPI_FTLR)

37.3.6.1 Offset

Register	Offset
SPI_FTLR	0x10

37.3.6.2 Function

This register controls the threshold value for the transmit FIFO memory and receive FIFO memory.

37.3.6.3 Fields

Field	Access	Description	Reset Value
1:0 TFT	RW	Transmit FIFO Threshold. TXE is set whenever the number of words in the transmit FIFO is equal to or less than TFT. Writing a value equal or greater than the FIFO size will be truncated, that is to say, the field can only be written when wdata [7 : 0] is not larger than FIFO depth 3. 00b: Transmit FIFO empty interrupt is asserted when 0 data entries are present in transmit FIFO 01b: Transmit FIFO empty interrupt is asserted when 1 or less data entry is present in transmit FIFO 10b: Transmit FIFO empty interrupt is asserted when 2 or less data entries are present in transmit FIFO 11b: Transmit FIFO empty interrupt is asserted when 3 or less data entries are present in transmit FIFO	0x0
15:2 RSVD_15_2	RO	Reserved	0x0
17:16 RFT	RW	Receive FIFO Threshold. RXF is set whenever the number of words in the receive FIFO is greater than RFT. Writing a value equal or greater than the FIFO size will be truncated, that is to say, the field can only be written when wdata [23 : 16] is not larger than FIFO depth 3. 00b: Receive FIFO full interrupt is asserted when 1 or more data entries are present in receive FIFO 01b: Receive FIFO full interrupt is asserted when 2 or more data entries are present in receive FIFO 10b: Receive FIFO full interrupt is asserted when 3 or more data entries are present in receive FIFO 11b: Receive FIFO full interrupt is asserted when 4 or more data entries are present in receive FIFO	0x0
31:18 RSVD_31_18	RO	Reserved	0x0

37.3.7 FIFO Level Register (SPI_FLR)

37.3.7.1 Offset

Register	Offset
SPI_FLR	0x14

37.3.7.2 Function

This register contains the number of valid data entries in the transmit FIFO memory and receive FIFO memory.

37.3.7.3 Fields

Field	Access	Description	Reset Value
2:0 TXTFL	RO	Contains the number of valid data entries in the transmit FIFO.	0x0
15:3 RSVD_15_3	RO	Reserved	0x0
18:16 RXTFL	RO	Contains the number of valid data entries in the receive FIFO.	0x0
31:19 RSVD_31_19	RO	Reserved	0x0

37.3.8 Status Register (SPI_SR)

37.3.8.1 Offset

Register	Offset
SPI_SR	0x18

37.3.8.2 Function

This is a read-only register used to indicate the current transfer status, FIFO status, and any transmission/reception errors that may have occurred.

37.3.8.3 Fields

Field	Access	Description	Reset Value
0 BUSY	RO	When set, indicates that a serial transfer is in progress 0b: SPI is idle or disabled 1b: SPI is actively transferring data	0x0
1 TFNF	RO	Transmit FIFO Not Full. Set when the transmit FIFO contains one or more empty locations, and cleared when the FIFO is full. 0b: Transmit FIFO is full 1b: Transmit FIFO is not full	0x1
2 TFE	RO	0b: Transmit FIFO is not empty 1b: Transmit FIFO is empty	0x1
3 RFNE	RO	0b: Receive FIFO is empty 1b: Receive FIFO is not empty	0x0
4 RFF	RO	When the receive FIFO is completely full, this bit is set. When the receive FIFO contains one or more empty locations, this bit is cleared.	0x0

Field	Access	Description	Reset Value
		0b: Receive FIFO is not full 1b: Receive FIFO is full	
5 TXE	RO	Set if the transmit FIFO is empty when a transfer is started 0b: No error 1b: Transmission error	0x0
31:6 RSVD_31_6	RO	Reserved	0x0

37.3.9 Interrupt Enable Register (SPI_IER)

37.3.9.1 Offset

Register	Offset
SPI_IER	0x1C

37.3.9.2 Function

This read/write register disables or enables all interrupts generated by the SPI.

37.3.9.3 Fields

Field	Access	Description	Reset Value
0 TXEIE	RW	Transmit FIFO empty interrupt enable. 0b: txe interrupt is disabled 1b: txe interrupt is enabled	0x1
1 TXOIE	RW	Transmit FIFO overflow interrupt enable. 0b: txo interrupt is disabled 1b: txo interrupt is enabled	0x1
2 RXUIE	RW	Receive FIFO underflow interrupt enable. 0b: rxu interrupt is disabled 1b: rxu interrupt is enabled	0x1
3 RXOIE	RW	Receive FIFO overflow interrupt enable. 0b: rxo interrupt is disabled 1b: rxo interrupt is enabled	0x1
4 RXFIE	RW	Receive FIFO full interrupt enable. 0b: rxf interrupt is disabled 1b: rxf interrupt is enabled	0x1
31:5 RSVD_31_5	RO	Reserved	0x1

37.3.10 Interrupt Status Register (SPI_ISR)

37.3.10.1 Offset

Register	Offset
SPI_ISR	0x20

37.3.10.2 Function

This register reports the status of the SPI interrupts after they have been masked and the status of the SPI interrupts prior to masking.

37.3.10.3 Fields

Field	Access	Description	Reset Value
0 TXEIS	RO	The bit is under the control of SPI_IER[TXEIE] and it will be set when TX FIFO is empty and SPI_IER[TXEIE] is enabled. 0b: TX FIFO empty interrupt is not asserted. 1b: TX FIFO empty interrupt is asserted.	0x0
1 TXOIS	RO	The bit is under the control of SPI_IER[TXOIE] and it will be set when TX FIFO is full and another write request is asserted and SPI_IER[TXOIE] is enabled. 0b: TX FIFO overflow interrupt is not asserted. 1b: TX FIFO overflow interrupt is asserted.	0x0
2 RXUIS	RO	The bit is under the control of SPI_IER[RXUIE] and it will be set when RX FIFO is empty and another read request is asserted and SPI_IER[RXUIE] is enabled. 0b: RX FIFO underflow interrupt is not asserted. 1b: RX FIFO underflow interrupt is asserted.	0x0
3 RXOIS	RO	The bit is under the control of SPI_IER[RXOIE] and it will be set only when RX FIFO is full and another write request is asserted and SPI_IER[RXOIE] is enabled. 0b: RX FIFO overflow interrupt is not asserted. 1b: RX FIFO overflow interrupt is asserted.	0x0
4 RXFIS	RO	The bit is under the control of SPI_IER[RXFIE] and it will be set only when RX FIFO is full and SPI_IER[RXFIE] is enabled. 0b: RX FIFO full interrupt is not asserted. 1b: RX FIFO full interrupt is asserted.	0x0
15:5 RSVD_15_5	RO	Reserved	0x0
16 TXEIR	RO	The bit is not under the control of SPI_IER[TXEIE] and it will be set when TX FIFO is empty regardless of whether SPI_IER[TXEIE] is enabled or not. 0b: TX FIFO is not empty. 1b: TX FIFO is empty.	0x0
17 TXOIR	RO	The bit is not under the control of SPI_IER[TXOIE] and it will be set when TX FIFO is full and another write request is asserted regardless of whether SPI_IER[TXOIE] is enabled or not. 0b: TX FIFO overflow event has not occurred. 1b: TX FIFO overflow event has occurred.	0x0
18 RXUIR	RO	The bit is not under the control of SPI_IER[RXUIE] and it will be set when RX FIFO is empty and another read request is asserted regardless of whether SPI_IER[RXUIE] is enabled or not. 0b: RX FIFO underflow event has not occurred. 1b: RX FIFO underflow event has occurred.	0x0
19 RXOIR	RO	The bit is not under the control of SPI_IER[RXOIE] and it will be set when RX FIFO is full and another write request is asserted regardless of whether SPI_IER[RXOIE] is enabled or not..	0x0

Field	Access	Description	Reset Value
		0b: RX FIFO overflow event has not occurred. 1b: RX FIFO overflow event has occurred.	
20 RXFIR	RO	The bit is not under the control of SPI_IER[RXFIE] and it will be set when RX FIFO is full regardless of whether SPI_IER[RXFIE] is enabled or not. 0b: RX FIFO is not full. 1b: RX FIFO is full.	0x0
31:21 RSVD_31_21	RO	Reserved	0x0

37.3.11 Transmit FIFO Overflow Interrupt Clear Register (SPI_TXOICR)

37.3.11.1 Offset

Register	Offset
SPI_TXOICR	0x24

37.3.11.2 Function

This register reflects the status of the interrupt.

37.3.11.3 Fields

Field	Access	Description	Reset Value
0 TXOICR	RO	Read this bit to clear Transmit FIFO Overflow interrupt flag.	0x0
31:1 RSVD_31_1	RO	Reserved	0x0

37.3.12 Receive FIFO Overflow Interrupt Clear Register (SPI_RXOICR)

37.3.12.1 Offset

Register	Offset
SPI_RXOICR	0x28

37.3.12.2 Function

This register reflects the status of the interrupt.

37.3.12.3 Fields

Field	Access	Description	Reset Value
0 RXOICR	RO	Read this bit to clear Receive FIFO Overflow interrupt flag.	0x0

Field	Access	Description	Reset Value
31:1 RSVD_31_1	RO	Reserved	0x0

37.3.13 Receive FIFO Underflow Interrupt Clear Register (SPI_RXUICR)

37.3.13.1 Offset

Register	Offset
SPI_RXUICR	0x2c

37.3.13.2 Function

This register reflects the status of the interrupt.

37.3.13.3 Fields

Field	Access	Description	Reset Value
0 RXUICR	RO	Read this bit to clear Receive FIFO Underflow Interrupt flag.	0x0
31:1 RSVD_31_1	RO	Reserved	0x0

37.3.14 Interrupt Clear Register (SPI_ICR)

37.3.14.1 Offset

Register	Offset
SPI_ICR	0x34

37.3.14.2 Function

This register reflects the status of the interrupt. A read from this register clears the txo, rxu, rxo interrupt.

37.3.14.3 Fields

Field	Access	Description	Reset Value
0 ICR	RO	Read this bit to clear all interrupt flags.	0x0
31:1 RSVD_31_1	RO	Reserved	0x0

37.3.15 DMA Control Register (SPI_DMACR)

37.3.15.1 Offset

Register	Offset

Register	Offset
SPI_DMACR	0x38

37.3.15.2 Function

The register is used to enable the DMA controller interface operation.

37.3.15.3 Fields

Field	Access	Description	Reset Value
0 RDMAE	RW	This bit enables/disables the receive FIFO DMA channel. 0b: Receive DMA is disabled 1b: Receive DMA is enabled	0x0
1 TDMAE	RW	This bit enables/disables the transmit FIFO DMA channel. 0b: Transmit DMA is disabled 1b: Transmit DMA is enabled	0x0
31:2 RSVD_31_2	RO	Reserved	0x0

37.3.16 DMA Transmit Data Level (SPI_DMATDLR)

37.3.16.1 Offset

Register	Offset
SPI_DMATDLR	0x3C

37.3.16.2 Function

This register controls the level at which a DMA request is made by the transmit logic.

37.3.16.3 Fields

Field	Access	Description	Reset Value
1:0 DMATDL	RW	Transmit Data Level. It is equal to the watermark level; That is the TX FIFO DMA request is generated when the number of valid data entries in the transmit FIFO is equal to or below this field value, and TDMAE = 1. The fields can only be written when the data written to [7 : 0] is not larger than FIFO depth 3. 00b: dma_tx_req is asserted when 0 data entries are present in the transmit FIFO. 01b: dma_tx_req is asserted when 1 or less data entry is present in the transmit FIFO. 10b: dma_tx_req is asserted when 2 or less data entries are present in the transmit FIFO. 11b: dma_tx_req is asserted when 3 or less data entries are present in the transmit FIFO.	0x0
31:2 RSVD_31_2	RO	Reserved	0x0

37.3.17 DMA Receive Data Level (SPI_DMARDLR)

37.3.17.1 Offset

Register	Offset
SPI_DMARDLR	0x40

37.3.17.2 Function

This register controls the level at which a DMA request is made by the receive logic.

37.3.17.3 Fields

Field	Access	Description	Reset Value
1:0 DMARDL	RW	Receive Data Level. The watermark level equals DMARDL + 1; That is the RX FIFO DMA request is generated when the number of valid data entries in the receive FIFO is equal to or above this field value + 1, and RDMAE = 1. The fields can only be written when the data written to [7 : 0] is not larger than FIFO depth 3. 00b: dma_rx_req is asserted when 1 or more data entries are present in the receive FIFO. 01b: dma_rx_req is asserted when 2 or more data entries are present in the receive FIFO. 10b: dma_rx_req is asserted when 3 or more data entries are present in the receive FIFO. 11b: dma_rx_req is asserted when 4 or more valid data entries are present in the receive FIFO.	0x0
31:2 RSVD_31_2	RO	Reserved	0x0

37.3.18 Data Register (SPI_DR_LOW)

37.3.18.1 Offset

Register	Offset
SPI_DR_LOW	0x60

37.3.18.2 Function

The data register is a 16/32-bit read/write buffer for the transmit/receive FIFOs.

37.3.18.3 Fields

Field	Access	Description	Reset Value
31:0 DR	RW	When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data is moved into the transmit FIFO buffer. Read = Receive FIFO buffer Write = Transmit FIFO buffer	0x0

37.3.19 Data Register (SPI_DR_HIGH)

37.3.19.1 Offset

Register	Offset
SPI_DR_HIGH	0xEC

37.3.19.2 Function

The data register is a 16/32-bit read/write buffer for the transmit/receive FIFOs.

37.3.19.3 Fields

Field	Access	Function	Reset Value
31:0 DR	RW	From 0x60 to 0xEC, one of the registers is read, data in the receive FIFO buffer is accessed. When it is written to, data is moved into the transmit FIFO buffer. Read = Receive FIFO buffer Write = Transmit FIFO buffer	0x0

37.4 Function description

This section provides the functional information of the SPI module.

37.4.1 Operation mode

The SPI in the chip can be configured as a master or a slave by setting SPI_CTRLR0[MST_MODE]. When this bit is set, that means SPI is configured as a master, master control logic is activated, and slave control logic is disabled.

37.4.1.1 Master

In SPI master mode, the module initiates the serial transfers by controlling the SCK and the PCS signals. The register SPI_BAUDR determines the baudrate of SCK, and the register SER determines which slave is selected. When a bit in SER is set, the corresponding slave select line from the master is activated when a serial transfer begins. However, setting or clearing bits in SER has no effect until a transfer has started. Thus users should enable the corresponding bits before a transfer begins.

37.4.1.2 Slave

In SPI slave mode, the module responds to transfer initiated by an SPI master. As the slave does not initiate the transfer, the clock polarity, clock phase, and frame size must be set correctly for successful communication with the master.

NOTE

- If the PCS is inactive, the slave SOUT pin will send data when there is a clock.
- If pin is not configured as the slave PCS by pinmux, the slave PCS is always active and data transfer begins when there is a clock.

37.4.2 Transfer mode

The module supports three transfer modes when transferring data by setting

SPI_CTRLR0[TMOD]: transmit and receive, transmit only, and receive only.

37.4.2.1 Transmit and receive

When SPI_CTRLR0[TMOD] = 2'b00, both transmit and receive logic in the control logic are valid. The data transfer occurs as expected according to the selected frame format. Transmit data is from the transmit FIFO and is pushed into the shift register. It is finally sent through the transmit data signal (txd) line to the target device. The received data comes in from the receive data signal (rxd) line, shifts to the receiver register, and finally is pushed into receive FIFO at the end of the data frame.

37.4.2.2 Transmit only

When SPI_CTRLR0[TMOD] = 2'b01, the device is configured to transmit only. The input from rxd line will be ignored and should not be pushed into the receive FIFO. Transmit data is popped from the transmit FIFO and sent through the txd line to the target device as normal. As the received data is invalid, users should mask interrupts from the receive logic when in transmit mode only.

37.4.2.3 Receive only

When SPI_CTRLR0[TMOD] = 2'b10, the device is configured to receive only, transmit data is invalid. The output signal txd line will remain at a constant logic level during the transmission. The received data from the target device is moved from the receive shift register into the receive FIFO at the end of each data frame. Users should mask interrupts originating from the transmit logic when this mode is entered.

37.4.3 Clock configuration

The limitation between the SPI bit-rate clock and the SPI functional clock:

- As a master, the minimum frequency of SPI function clock is 2 times the maximum expected frequency of the bit-rate clock.
- As a slave, the minimum frequency of SPI function clock is 6 times the maximum expected frequency of the bit-rate clock.

37.4.4 Transfer formats

The module provides two bits: SPI_CTRLR0[SCPOL] and SPI_CTRLR0[SCPH], for the user to control the serial clock polarity and phase.

37.4.4.1 SCPH=0

SPI_CTRLR0[SCPH] is used to control the relationship between the serial clock and the slave select signal. When SPI_CTRLR0[SCPH]=0, data transmission begins on the falling edge of the slave select signal. The first data bit is captured by the master and slave peripherals on the first edge of the serial clock. Therefore, valid data must be present on the txd and rxd lines prior to the first serial clock edge. When SPI_CTRLR0[SCPH] = 0, data will be sampled on the odd edge of SCK and will change on the even edge of SCK. The inactive state of SCK is controlled by SPI_CTRLR0[SCPOL]. When SPI_CTRLR0[SCPOL] = 1, the inactive state of SCK is high. When SPI_CTRLR0[SCPOL] = 0, the inactive state of SCK is low.

For the SPI master, PCS becomes high level between frames and the serial clock is held to its default value.

For the SPI slave, master shall set PCS to high level between frames. Otherwise, the transmission will fail.

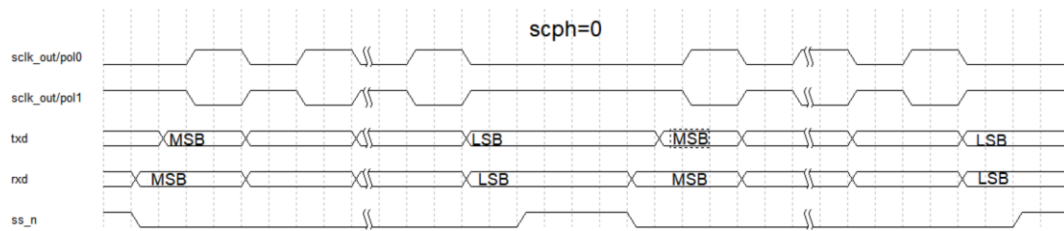


Figure 37-2 SPI transfer waveform (SCPH=0)

37.4.4.2 SCPH=1

When `SPI_CTRLR0[SCPH] = 1`, the device requires the first SCK edge before the first data bit becomes available on the slave SOUT pin. In this format, the master and slave devices change the data on their txd lines on the odd SCK edges and sample the data on their rxn pins on the even-numbered SCK edges. Finally, the master initiates the transfer by asserting the PCS signal to the slave.

The diagram below shows the SPI continue transmit mode. The PCS is held active for the duration of the transfer.

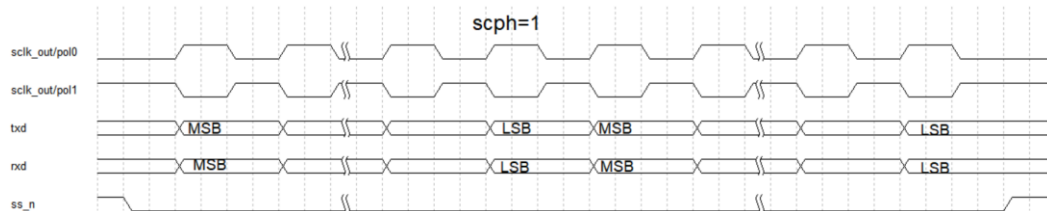


Figure 37-3 SPI transfer waveform (SCPH=1)

37.4.5 Data frame size and FIFO

The module supports 4-32 bits data frame, users can configure the data frame size by setting `SPI_CTRLR0[DFS_32]`, the calculation of data frame size, and `SPI_CTRLR0[DFS_32]` as below:

$$\text{Data frame size} = \text{SPI\_CTRLR0[DFS\_32]} + 1$$

As the FIFO width is 32 bits, when the data frame size is programmed to be less than 32 bits, the received data is automatically right-justified by the receive logic with the upper bits of the receive FIFO zero-padded. Therefore, users should make sure the transmit data is right-justified before writing into the transmit FIFO. The transmit logic ignores the unused upper bits when transmitting the data.

37.4.6 DMA and interrupt

The module has two separate DMA requests for receive FIFO and transmit FIFO. Users can configure the DMA request trigger condition by setting `SPI_DMATDLR` and `SPI_DMARDLR` registers. After the DMA transfer is done, the module will receive a DMA acknowledge signal.

The module has several conditions to generate interrupt requests, with each request having a flag bit in the register `SPI_ISR`. Users can enable/disable these interrupts by configuring the register `SPI_IER`.

37.5 Initialization application information

All devices in the SPI bus are idle before they are used.

The following steps initialize the master device for the serial transfer:

1. Write `SPI_CTRLR0` register bit [21] to logic '1' to be master. Then, write other

SPI_CTRLR0 fields to match the required transfer.

2. If a transfer is receiving only, write number of frames into SPI_CTRLR1.
3. Write SPI_BAUDR to set the transfer baud rate.
4. Write SPI_FTLR to set FIFO threshold levels.
5. Write the SPI_IER register to set interrupt masks.
6. Write the SPI_SSENr register bit [17:16] to choose PCS0/PCS1 as chip select signal.
7. Write the SPI_SSENr register bit [0] to logic '1' to enable the master.
8. Write the transmitting data into DATA register to start the data transfer.

The following steps initialize the target slave device for the serial transfer:

1. Write SPI_CTRLR0 register bit [21] to logic '0' to be slave. Write SPI_CTRLR0 to match the required transfer.
2. Write SPI_FTLR to set FIFO threshold levels.
3. Write the SPI_IER register to set interrupt masks.
4. Write the SPI_SSENr register bit [0] to logic '1' to enable the slave. If the slave is to transmit data, write data into transmit FIFO.
5. Start serial transmission to write the transmitted data to the transmit FIFO of the master.
6. Users can poll the busy state or use SPI_ISR to determine when the serial transfer is complete.

Chapter 38

Inter-Integrated Circuit (I2C)

38.1 Introduction

The purpose of this chapter is to define an Inter-Integrated Circuit (I2C).

38.1.1 I2C terminology

- SDA: serial data line.
- SCL: serial clock line.
- Receiver: I2C master can work as a master receiver when it reads data from the I2C bus. I2C slave can also work as a slave receiver when it receives data from the I2C bus.
- Transmitter: I2C master can work as a master transmitter via transmitting data to the I2C bus. I2C slave can also work as a slave transmitter via responding to master's read request interrupt and sending data to the I2C bus.
- Master: I2C acts as a master that can launch a transfer, generates its SCL, and terminates its transmission via stop command. The transmitter or receiver can work as an I2C master.
- Arbitrate: When there are two or more I2C masters connected to the I2C bus, I2C only allows one master one time to occupy the bus.
- Synchronization: When there are two or more I2C masters attempting to occupy the I2C bus, it defines the procedure to synchronize SCL clock.
- Start (re-start): I2C will launch a transfer via start or re-start condition. Once the SDA line changes from 1 to 0 when SCL keeps 1, I2C will start a transfer, and bus becomes busy.
- Stop: I2C will terminate current transfer via stop condition. Once the SDA line changes from 0 to 1 when SCL keeps 1, the I2C transfer will be terminated.

38.1.2 Features

I2C supports the following key features:

- I2C includes serial data line SDA and serial clock line SCL.
- Support the following speeds:
 - high-speed mode (≤ 2.6 Mb/s)
 - fast mode (≤ 400 Kb/s) or fast mode plus (≤ 1000 Kb/s)
 - standard mode (0 to 100Kb/s)
- Support 8/16/32 bits access type with APB interface.
- Support both I2C master and slave mode.
- Support 7-bit and 10-bit address match.
- Support read and write mixed transfer on 7-bit or 10-bit mode.
- Support SCL synchronization when multi-master arbitrates their SCL.
- Support separate TXFIFO and RXFIFO.
- Support I2C interrupt or polling mode.
- Support DMA request for RXFIFO and TXFIFO separately.
- SDA configurable hold time.
- Support recovering I2C bus when SDA/SCL is stuck at low.

38.1.3 Block diagram

The following figure is a block diagram of the Inter-Integrated Circuit (I2C) module.

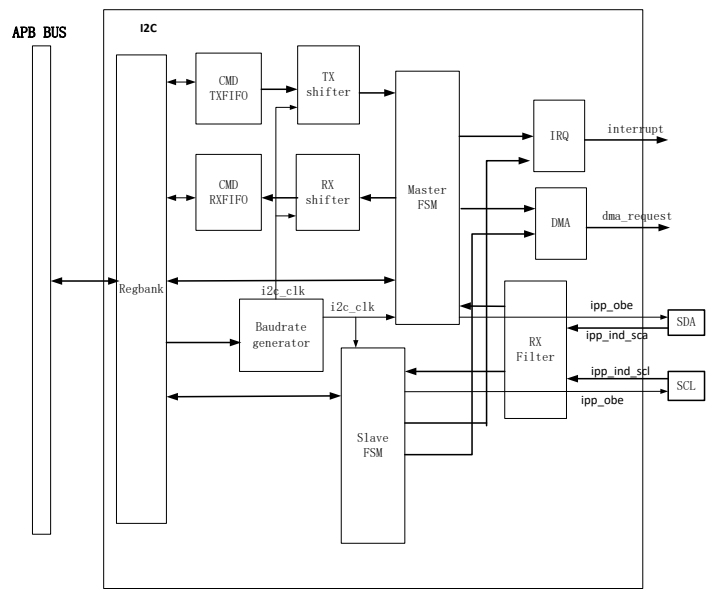


Figure 38-1 I2C block diagram

38.1.4 Instantiation information

The following table lists the I2C instantiation information for each chip in the product series.

Configuration	K144MC	K146MC	K148MC
Instance(s)	I2C0	I2C0	I2C0
	I2C1	I2C1	I2C1

38.2 Signal description

The I2C bus is a two-wire serial interface consisting of a serial data line (SDA) and a serial clock (SCL). These wires carry information between the devices connected to the bus.

Table 38-1 I2C signal descriptions

Signal	Name	2-Wire Scheme	I/O
SCL	I2C clock line	SCL	I/O
SDA	I2C data line	SDA	I/O

38.3 Memory map and register definition

This section provides the memory map and register definitions of the I2C module.

38.3.1 Memory map

I2C0 base address: 0x40066000
I2C1 base address: 0x40058000

Table 38-2 I2C Memory map

Offset	Register	Width (In bits)	Reset Value
0x04	I2C Parameter Register (I2C_PARAMETER)	32	0x00000227

Offset	Register	Width (In bits)	Reset Value
0x08	I2C Configure0 Register (I2C_CONFIG0)	32	0x00000082
0x0C	I2C Configure1 Register (I2C_CONFIG1)	32	0x000006C7
0x10	I2C SDA Setup Timing Register (I2C_SDA_SETUP_TIMING)	32	0x00000064
0x14	I2C SDA Hold Timing Register (I2C_SDA_HOLD_TIMING)	32	0x00000001
0x18	I2C DMA Control Register (I2C_DMA_CTRL)	32	0x00000000
0x1C	I2C Interrupt Enable Register (I2C_INT_ENABLE)	32	0xCBC00007
0x20	I2C Rx/Tx Data Buffer and Command Register (I2C_COMMAND_DATA)	32	0x00000000
0x24	I2C Destination Address Register (I2C_DEST_ADDR)	32	0x00000055
0x28	I2C Slave Address Register (I2C_SLAVE_ADDR)	32	0x00000055
0x2C	Standard Speed I2C SCL High Count Register (I2C_STD_SCL_HCNT)	32	0x00000190
0x30	Standard Speed I2C SCL Low Count Register (I2C_STD_SCL_LCNT)	32	0x000001D6
0x34	Fast Mode I2C SCL High Count Register (I2C_FST_SCL_HCNT)	32	0x0000003C
0x38	Fast Mode I2C SCL Low Count Register (I2C_FST_SCL_LCNT)	32	0x00000082
0x3C	High Speed I2C SCL High Count Register (I2C_HS_SCL_HCNT)	32	0x00000006
0x40	High Speed I2C SCL Low Count Register (I2C_HS_SCL_LCNT)	32	0x00000010
0x44	I2C Receive FIFO Water Mark Register (I2C_RXFIFO_WATER_MARK)	32	0x00000000
0x48	I2C Transmit FIFO Water Mark Register (I2C_TXFIFO_WATER_MARK)	32	0x00000000
0x4C	I2C Transmit FIFO Valid Entry Counter Register (I2C_TX_FIFO_CNT)	32	0x00000000
0x50	I2C Receive FIFO Valid Entry Counter Register (I2C_RX_FIFO_CNT)	32	0x00000000
0x54	I2C Fast and Standard Spike Suppression Counter Register (I2C_FSTD_SPKCNT)	32	0x00000005
0x58	I2C High Speed Spike Suppression Counter Register (I2C_HS_SPKCNT)	32	0x00000001
0x5C	I2C Interrupt Status Register (I2C_STATUS0)	32	0x00000000
0x60	I2C Status Register (I2C_STATUS1)	32	0x000000C0
0x64	I2C Error Status Register (I2C_ERROR_STATUS)	32	0x00000000
0x68	I2C SCL Stuck at Low Timeout Register (I2C_SCL_LOW_TIMEOUT)	32	0xFFFFFFFF
0x6c	I2C SDA Stuck at Low Timeout Register (I2C_SDA_LOW_TIMEOUT)	32	0xFFFFFFFF

Offset	Register	Width (In bits)	Reset Value
0x70	I2C Read Clear Error Status Register (I2C_RD_CLR_ERR_STATUS)	32	0x00000000

38.3.2 I2C Parameter Register (I2C_PARAMETER)

38.3.2.1 Offset

Register	Offset
I2C_PARAMETER	0x04

38.3.2.2 Function

This register is for configuration parameters of the module.

38.3.2.3 Fields

Field	Access	Description	Reset Value
1:0 I2C_SUPPORT_SPEED	RO	0x0: Reserved 0x1: Support standard speed 0x2: Support fast speed or fast speed plus 0x3: Support high speed	0x3
2 DMA_SUPPORT	RO	DMA feature supported 0x0: Do not support DMA 0x1: Support DMA feature	0x1
6:3 TXFIFO_DEPTH	RO	I2C TXFIFO depth is 4 bytes	0x4
10:7 RXFIFO_DEPTH	RO	I2C RXFIFO depth is 4 bytes	0x4
31:11 RSVD_31_11	RO	Reserved	0x0

38.3.3 I2C Configure0 Register (I2C_CONFIG0)

38.3.3.1 Offset

Register	Offset
I2C_CONFIG0	0x08

38.3.3.2 Fields

Field	Access	Description	Reset Value
0 MODULE_EN	RW	I2C module enable bit 0b: I2C module is disabled. 1b: I2C module is enabled.	0x0
1 SLAVE_ACK_G ENCALL	RW	Acknowledge of general call. 0b: Do not give ack when receiving general call and could not allow generating general call interrupt. 1b: Give ack when receiving general call and allow generating general call interrupt.	0x1

Field	Access	Description	Reset Value
		Note: This bit can directly configured when the slave bus is idle. If there is transfer, I2C module should be disabled to re-configure this bit.	
2 SLAVE_NACK	RW	Generate NACK only for I2C slave mode. 0b: Generates ACK/NACK normally. 1b: A slave will generate NACK after data byte is received. This data will be aborted, not pushed into RX FIFO. Note: This bit can be directly configured when the slave bus is idle. If there is transfer, I2C module should be disabled to re-configure this bit.	0x0
3 SDA_RECOVER_EN	RW	NOTE This bit is only valid when MASTER_BUS_RECOVER_EN = 1. If SDA is stuck at low phase when Tx abort case occurs, this bit is used to send at most 9 SCL clks and stop to release the SDA bus line. This bit will get auto cleared.	0x0
4 MASTER_ABORT	RW	This bit can be set to 1 to abort I2C transfer when I2C is enabled as master mode only. I2C will issue a stop signal, and flush TX FIFO after completing the interrupt. It will be automatically cleared after the abort is done. 0b: Abort is done or has no abort event. 1b: I2C will abort after the current transfer when I2C is enabled	0x0
5 MASTER_BUS_RECOVER_EN	RW	In I2C master mode, it will be used to be combined with SDA_RECOVER_EN and SDA SCL timeout counters. 0b: Bus recover feature is disabled. 1b: Bus recover feature is enabled.	0x0
6 MASTER_BLK_TXFIFO	RW	Master block TX bit, when I2C is in master mode. NOTE There is a need to priorly set this bit when I2C is idle, and TX FIFO is empty. It will block TX FIFO transfer when any data is put into TX FIFO until this bit is cleared to 0. 0b: I2C will start transmission automatically once the first data is available in the TX FIFO. 1b: It will block transmission of I2C data even when TX FIFO has data to transmit.	0x0
9:7 H_MCODE	RW	I2C supports up to 8 high-speed masters on the I2C bus. Each master has one unique master ID, such as 0, 1, 2, ..., 7. These bits are valid only for the master mode. Only when the I2C module is disabled, the writing of these bits is valid.	0x1
31:10 RSVD_31_10	RO	Reserved	0x0

38.3.4 I2C Configure1 Register (I2C_CONFIG1)

38.3.4.1 Offset

Register	Offset
I2C_CONFIG1	0x0C

38.3.4.2 Function

The register writing is valid only when the I2C module is disabled.

38.3.4.3 Fields

Field	Access	Description	Reset Value
0 SLAVE_MODE_DIS	RW	0b: Slave is enabled 1b: Slave is disabled By default, I2C slave is disabled.	0x1
1 MASTER_MODE_EN	RW	0b: Master is disabled 1b: Master is enabled By default, I2C master is enabled.	0x1
2 MASTER_RESTART_EN	RW	I2C master re-start enable bit 0b: Disabled 1b: Enabled I2C master could not do the following items if MASTER_RESTART_EN= 0: • Send a start byte. • Perform any high-speed mode operation. • Perform direction changes in combined format mode. • Perform a read operation with a 10-bit address.	0x1
3 SLAVE_STOP_DET_EN	RW	For I2C slave mode, 0b: It generates the STOP_DETECT interrupt flag no matter the slave address matches or not. 1b: When set to 1, it allows generating STOP_DETECT interrupt flag only when the slave address matches.	0x0
4 MASTER_STOP_DET_EN	RW	For I2C master mode, 0b: It generates the STOP_DETECT interrupt flag no matter what the master's state is. 1b: When set to 1, it allows generating a STOP_DETECT interrupt flag only when the I2C master is active.	0x0
5 TXFIFO_EMPTY_EN	RW	This bit controls the generation of the TXFIFO_EMPTY interrupt. 0b: I2C_STATUS0[TXFIFO_EMPTY] is set to 1 when the transmit buffer is at or below the threshold value set in the I2C_TXFIFO_WATER_MARK register. 1b: I2C_STATUS0[TXFIFO_EMPTY] is set to 1 when the transmit buffer is at or below the threshold value set in the I2C_TXFIFO_WATER_MARK register and the transmission of the address/data from	0x0

Field	Access	Description	Reset Value
		the internal shift register for the most recently popped command is completed.	
7:6 SPEED_SEL	RW	I2C speed select bits It depends on which speed type the I2C operates: 0x00: Reserved 0x01: I2C standard mode (up to 100 Kb/s) 0x02: Fast mode (up to 400 Kb/s) or fast mode plus (up to 1 Mb/s) 0x03: High speed mode (up to 2.6 Mb/s)	0x3
8 HOLD_EN_RXFIFO_FULL	RW	This bit is used to determine whether to hold the I2C bus after RX FIFO is full. 0b: I2C bus will not be held after RX FIFO is full. 1b: Allow holding the I2C bus after RX FIFO is full.	0x0
9 SLAVE_10BIT_ADDR_SEL	RW	This bit is for I2C slave mode only. It controls whether the I2C module chooses 7-bit or 10-bit addressing. 0b: 7-bit addressing. 1b: 10-bit addressing.	0x1
10 MASTER_10BIT_ADDR_SEL	RW	This bit is for master mode only. It controls whether the I2C module chooses 7-bit or 10-bit addressing. 0b: 7-bit addressing. 1b: 10-bit addressing.	0x1
31:11 RSVD_31_11	RO	Reserved	0x0

38.3.5 I2C SDA Setup Timing Register (I2C_SDA_SETUP_TIMING)

38.3.5.1 Offset

Register	Offset
I2C_SDA_SETUP_TIMING	0x10

38.3.5.2 Function

It controls I2C SDA setup timing requirement when I2C acts as slave transmitter.

38.3.5.3 Fields

Field	Access	Description	Reset Value
7:0 SDA_SETUP_TIMING	RW	The I2C SDA setup timing is calculated via $(\text{SDA\_SETUP\_TIMING}-1) * (\text{I2C function clock period})$. For example, if I2C requires SDA setup time is 2000ns, I2C function clock is 10M, then SDA_SETUP_TIMING should be configured as 21. NOTE This register can be written only when the I2C	0x64

Field	Access	Description	Reset Value
		module is disabled. SDA_SETUP_TIMING value at least equals 2.	
31:8 RSVD_31_8	RO	Reserved	0x0

38.3.6 I2C SDA Hold Timing Register (I2C_SDA_HOLD_TIMING)

38.3.6.1 Offset

Register	Offset
I2C_SDA_HOLD_TIMING	0x14

38.3.6.2 Function

The register is used to configure SDA hold timing counter when SCL is in low phase. Write access is allowed to the whole register only when I2C is disabled. This register timing values are in unit of I2C function clock period.

38.3.6.3 Fields

Field	Access	Description	Reset Value
15:0 SDA_TX_HOLD_TIMING	RW	Sets the required SDA hold time in units of I2C function clock period, when I2C module acts as a transmitter. For master mode, drive on SDA takes IC_SDA_TX_HOLD cycles of I2C function clock after driving SCL to logic 0. When IC_SDA_TX_HOLD is set 0, it takes 1 cycle of I2C function clock to drive on SDA after driving SCL to logic 0. For slave mode, (SPKLEN is FSTD_SPKLEN for non-high-speed mode, and HS_SPKLEN under high speed mode) when IC_SDA_TX_HOLD is set lesser than SPKLEN + 7, drive on SDA takes SPKLEN + 7 cycles of I2C function clock after driving SCL to logic 0. otherwise, drive on SDA takes IC_SDA_TX_HOLD cycles of I2C function clock after driving SCL to logic 0.	0x1
23:16 SDA_RX_HOLD_TIMING	RW	Sets the required SDA hold time in units of I2C function clock period, when I2C module acts as a receiver. The maximum values of SDA_RX_HOLD_TIMING that can be programmed in the register for the respective speed modes as following: For master mode, Standard Mode: STD_SCL_HCNT – FSTD_SPKLEN – 3 Fast Mode or Fast Mode Plus: FST_SCL_HCNT – FSTD_SPKLEN – 3 High Speed (IC_CAP_LOADING =100):Min {FST_SCL_HCNT – FSTD_SPKLEN – 3,	0x0

Field	Access	Description	Reset Value
		HS_SCL_LCNT – HS_SPKLEN – 3} High Speed(IC_CAP_LOADING =400): Min {FST_SCL_HCNT – FSTD_SPKLEN – 3, (HS_SCL_LCNT/2) – HS_SPKLEN – 3} In Slave mode, make sure the SDA_RX_HOLD_TIMING does not exceed the maximum SCL fall time.	
31:24 RSVD_31_24	RO	Reserved	0x0

38.3.7 I2C DMA Control Register (I2C_DMA_CTRL)

38.3.7.1 Offset

Register	Offset
I2C_DMA_CTRL	0x18

38.3.7.2 Function

This register can be programmed regardless of whether I2C module is enabled or not.

38.3.7.3 Fields

Field	Access	Description	Reset Value
0 RXFIFO_DMA_EN	RW	0b: Disable I2C DMA Rx feature. 1b: Enable I2C DMA Rx feature.	0x0
1 TXFIFO_DMA_EN	RW	0b: Disable I2C DMA Tx feature. 1b: Enable I2C DMA Tx feature.	0x0
3:2 DMA_TXFIFO_WATERMARK	RW	I2C DMA TXFIFO watermark. This bit field controls at which level an I2C DMA Tx request is made by the transmit logic. The watermark level =DMA_TXFIFO_WATERMARK; I2C DMA Tx request signal is generated when the number of valid data entries in TXFIFO is equal to or below this field value, and TXFIFO_DMA_EN=1.	0x0
5:4 DMA_RXFIFO_WATERMARK	RW	I2C DMA RX FIFO watermark. This bit field controls at which level the receive logic makes an I2C DMA Rx request. The watermark level = DMA_RXFIFO_WATERMARK+1; I2C DMA Rx request is generated when the number of valid data entries in the RX FIFO is equal to or more than this field value + 1, and RXFIFO_DMA_EN = 1. For instance, when DMA_RXFIFO_WATERMARK is 0, the DMA request is asserted when 1 or more data entries are present in the RX FIFO.	0x0
31:6 RSVD_31_6	RO	Reserved	0x0

38.3.8 I2C Interrupt Enable Register (I2C_INT_ENABLE)

38.3.8.1 Offset

Register	Offset
I2C_INT_ENABLE	0x1C

38.3.8.2 Function

This register is I2C interrupt enable control register.

38.3.8.3 Fields

Field	Access	Description	Reset Value
0 I2C_CALL_IE	RW	I2C general call interrupt enable 0b: Disable 1b: Enable	0x1
1 I2C_TX_OVF_IE	RW	I2C Tx overflow interrupt enable 0b: Disable 1b: Enable	0x1
2 I2C_ERROR_ABORT_IE	RW	I2C error abort interrupt enable 0b: Disable 1b: Enable	0x1
3 I2C_ACTIVITY_DET_IE	RW	I2C activity interrupt enable 0b: Disable 1b: Enable	0x0
4 I2C_STOP_DET_IE	RW	I2C stop detect interrupt enable 0b: Disable 1b: Enable	0x0
5 I2C_START_DET_IE	RW	I2C start detect interrupt enable 0b: Disable 1b: Enable	0x0
21:6 RSVD_21_6	RO	Reserved	0x0
22 SLAVE_RD_REQ_IE	RW	Enable for slave read request interrupt. 0b: Disable 1b: Enable	0x1
23 RX_DONE_IE	RW	If I2C is a slave-transmitter, this bit is set to 1 if the master does not acknowledge a transmitted byte. It indicates that transmission is done on the last byte of the transmission. 0b: Disable 1b: Enable	0x1
24 RX_UNDER_IE	RW	Read RX FIFO When it is empty, it will cause interrupt setting. 0b: Disable 1b: Enable	0x1
25 RX_OVF_IE	RW	RX FIFO overflow interrupt enable 0b: Disable 1b: Enable	0x1
26 RESTART_DET_IE	RW	Restart detect interrupt enable 0b: Disable 1b: Enable	0x0
27	RW	SCL stuck at low interrupt enable	0x1

Field	Access	Description	Reset Value
SCL_STUCK_AT_LOW_IE		0b: Disable 1b: Enable	
29:28 RSVD_29_28	RO	Reserved	0x0
30 RX_FULL_IE	RW	RX FIFO full interrupt enable 0b: Disable 1b: Enable	0x1
31 TX_EMPTY_IE	RW	TX FIFO empty interrupt enable 0b: Disable 1b: Enable	0x1

38.3.9 I2C Rx/Tx Data Buffer and Command Register (I2C_COMMAND_DATA)

38.3.9.1 Offset

Register	Offset
I2C_COMMAND_DATA	0x20

38.3.9.2 Function

It only supports the 32-bit write option.

38.3.9.3 Fields

Field	Access	Description	Reset Value
7:0 DATA	RW	Transfer data of FIFO.	0x0
8 CMD_MASTER_DIRECTION	WO	This bit controls I2C transfer direction. 0b: TX 1b: RX	0x0
9 CMD_STOP	WO	This CMD bit controls whether to issue a STOP signal after the byte is sent or received, considering TX FIFO will hold bus after empty. 0b: Master will not issue a STOP signal after the byte is sent or received, and bus will be held. 1b: Master will issue a STOP signal after the byte is sent or received, and bus will be released.	0x0
10 CMD_RESTART	WO	This CMD bit controls whether to launch a re_start option before transferring data byte. 0b: If MASTER_RESTART_EN=1, the master will resend data only when direction changes. Otherwise, it has no effect. If MASTER_RESTART_EN=0, it has no effect. The slave address only matches one time. 1b: If MASTER_RESTART_EN=1, the master will resend the address after the	0x0

Field	Access	Description	Reset Value
		data byte. If MASTER_RESTART_EN=0, the master will issue a pair of stop and start after the data byte.	
31:11 RSVD_31_11	RO	Reserved	0x0

38.3.10 I2C Destination Address Register (I2C_DEST_ADDR)

38.3.10.1 Offset

Register	Offset
I2C_DEST_ADDR	0x24

38.3.10.2 Function

The register writing is valid only when the I2C module is disabled.

38.3.10.3 Fields

Field	Access	Description	Reset Value
9:0 DEST_ADDR	RW	This is the destination address for any master transaction. NOTE These bits are ignored for the I2C general call.	0x55
11:10 MASTER_CMD_SEL	RW	Master command select bits will choose different transfer addresses. 00b: Transfer DEST_ADDR [9:0] normally. 01b: Transfer DEST_ADDR [9:0] normally. 10b: General Call Address - I2C will launch a General Call (transfer data is 0x00). Only writes may be performed. Any read option will result in asserting ERROR_ABORT flag. I2C will exit general call mode via clearing I2C_DEST_ADDR [11] to 0. 11b: I2C transfer start byte (transfer data is 0x01), should no slave ack.	0x0
31:12 RSVD_31_12	RO	Reserved	0x0

38.3.11 I2C Slave Address Register (I2C_SLAVE_ADDR)

38.3.11.1 Offset

Register	Offset
I2C_SLAVE_ADDR	0x28

38.3.11.2 Function

The register writing is valid only when the I2C module is disabled.

38.3.11.3 Fields

Field	Access	Description	Reset Value
9:0 SLV_ADDR	RW	For 7-bit I2C address mode, only SLV_ADDR [6:0] is used. NOTE The default values cannot be any of the reserved address locations: 0x00 to 0x07 and 0x78 to 0x7f. Therefore, the device's correct operation is not guaranteed if you program the I2C_SLAVE_ADDR or I2C_DEST_ADDR to a reserved value. This register can be written only when the I2C interface is disabled, corresponding to I2C_CONFIG0[0] being set to 0. Writes at other times have no effect.	0x55
31:10 RSVD_31_10	RO	Reserved	0x0

38.3.12 Standard Speed I2C SCL High Count Register (I2C_STD_SCL_HCNT)

38.3.12.1 Offset

Register	Offset
I2C_STD_SCL_HCNT	0x2C

38.3.12.2 Function

The register writing is valid only when the I2C module is disabled.

38.3.12.3 Fields

Field	Access	Description	Reset Value
15:0 STD_SCL_HCNT	RW	NOTE The minimum configured value is 6. Therefore, any written values that are less than 6 will be blocked. If using 8-bit write access, then you must configure STD_SCL_HCNT[7:0] first, then STD_SCL_HCNT [15:8]. When configuring this register, users must make sure: $\text{STD\_SCL\_HCNT} + \text{I2C\_FSTD\_SPKCNT} + 0x7 \leq 0xffff$. This value should not be larger than 65525 because I2C uses STD_SCL_HCNT+10 to generate idle conditions.	0x190
31:16 RSVD_31_16	RO	Reserved	0x0

38.3.13 Standard Speed I2C SCL Low Count Register (I2C_STD_SCL_LCNT)

38.3.13.1 Offset

Register	Offset
I2C_STD_SCL_LCNT	0x30

38.3.13.2 Function

The register writing is valid only when the I2C module is disabled.

38.3.13.3 Fields

Field	Access	Description	Reset Value
15:0 STD_SCL_LCNT	RW	NOTE The minimum configured value is 8. Any written values that are smaller than 8 will be blocked. If using 8-bit write access, you must configure STD_SCL_LCNT[7:0] first, then STD_SCL_LCNT[15:8]. When configuring this register, users must make sure: STD_SCL_LCNT<=0xfffe.	0x1D6
31:16 RSVD_31_16	RO	Reserved	0x0

38.3.14 Fast Mode I2C SCL High Count register (I2C_FST_SCL_HCNT)

38.3.14.1 Offset

Register	Offset
I2C_FST_SCL_HCNT	0x34

38.3.14.2 Function

The register writing is valid only when the I2C module is disabled.

38.3.14.3 Fields

Field	Access	Description	Reset Value
15:0 FST_SCL_HCNT	RW	Note: the minimum configure value is 6, any written values less than 6 will be blocked. If using 8 bits write access, then users must configure FST_SCL_HCNT[7:0] first, then FST_SCL_HCNT[15:8]. When configuring this register, users must make sure: FST_SCL_HCNT+I2C_FSTD_SPKCNT+0x7<=0xffff.	0x3C
31:16 RSVD_31_16	RO	Reserved	0x0

38.3.15 Fast Mode I2C SCL Low Count Register (I2C_FST_SCL_LCNT)

38.3.15.1 Offset

Register	Offset
I2C_FST_SCL_LCNT	0x38

38.3.15.2 Function

The register writing is valid only when the I2C module is disabled.

38.3.15.3 Fields

Field	Access	Description	Reset Value
15:0 FST_SCL_LCNT	RW	Note: the minimum configure value is 8, any written values less than 8 will be blocked. If using 8 bits write access, then users must configure FST_SCL_LCNT[7:0] first, then FST_SCL_LCNT[15:8]. When configuring this register, users must make sure: FST_SCL_LCNT ≤ 0xfffe.	0x82
31:16 RSVD_31_16	RO	Reserved	0x0

38.3.16 High Speed I2C SCL High Count Register (I2C_HS_SCL_HCNT)

38.3.16.1 Offset

Register	Offset
I2C_HS_SCL_HCNT	0x3C

38.3.16.2 Function

The register writing is valid only when the I2C module is disabled.

38.3.16.3 Fields

Field	Access	Description	Reset Value
15:0 HS_SCL_HCNT	RW	NOTE The minimum configured value is 6, any written values less than 6 will be blocked. If using 8 bits write access, then users must configure HS_SCL_HCNT[7:0] first, then HS_SCL_HCNT[15:8]. When configuring this register, users must make sure: HS_SCL_HCNT + I2C_HS_SPKCNT + 0x7 ≤ 0xffff.	0x6

Field	Access	Description	Reset Value
31:16 RSVD_31_16	RO	Reserved	0x0

38.3.17 High Speed I2C SCL Low Count Register (I2C_HS_SCL_LCNT)

38.3.17.1 Offset

Register	Offset
I2C_HS_SCL_LCNT	0x40

38.3.17.2 Function

The register writing is valid only when the I2C module is disabled.

38.3.17.3 Fields

Field	Access	Description	Reset Value
15:0 HS_SCL_LCNT	RW	NOTE The minimum configured value is 8, any written values less than 8 will be blocked. If using 8 bits write access, then users must configure HS_SCL_LCNT[7:0] first, then HS_SCL_LCNT[15:8]. When configuring this register, users must make sure: HS_SCL_LCNT<=0xfffe.	0x10
31:16 RSVD_31_16	RO	Reserved	0x0

38.3.18 I2C Receive FIFO Water Mark Register (I2C_RXFIFO_WATER_MARK)

38.3.18.1 Offset

Register	Offset
I2C_RXFIFO_WATER_MARK	0x44

38.3.18.2 Fields

Field	Access	Description	Reset Value
1:0 RXFIFO_WATER_MARK	RW	The valid range is 0-3. If RXFIFO_WATER_MARK=3, that means the RX FIFO valid entry water mark level is 4. Once RX FIFO entries reach or are above (RXFIFO_WATER_MARK+1), I2C will generate RX FIFO full flag. Do not set this register to a value that is larger than the RX FIFO maximum depth =3.	0x0
31:2 RSVD_31_2	RO	Reserved	0x0

38.3.19 I2C Transmit FIFO Water Mark Register (I2C_TXFIFO_WATER_MARK)

38.3.19.1 Offset

Register	Offset
I2C_TXFIFO_WATER_MARK	0x48

38.3.19.2 Fields

Field	Access	Description	Reset Value
1:0 TXFIFO_WATER_MARK	RW	The valid range is 0-3. If TXFIFO_WATER_MARK=3, that means the TXFIFO valid entry water mark level is 3. Once TXFIFO entries reach or are below (TXFIFO_WATER_MARK), I2C will generate TXFIFO empty flag. Do not set this register to a value that is larger than TXFIFO maximum depth=3.	0x0
31:2 RSVD_31_2	RO	Reserved	0x0

38.3.20 I2C Transmit FIFO Valid Entry Counter Register (I2C_TX_FIFO_CNT)

38.3.20.1 Offset

Register	Offset
I2C_TX_FIFO_CNT	0x4C

38.3.20.2 Fields

Field	Access	Description	Reset Value
2:0 TX_FIFO_CNT	RO	The valid data entries number in TX FIFO. It will be cleared via disabling the I2C module (MODULE_EN to 0) or when an ERROR_ABORT event occurs.	0x0
31:3 RSVD_31_3	RO	Reserved	0x0

38.3.21 I2C Receive FIFO Valid Entry Counter Register (I2C_RX_FIFO_CNT)

38.3.21.1 Offset

Register	Offset
I2C_RX_FIFO_CNT	0x50

38.3.21.2 Fields

Field	Access	Description	Reset Value
2:0 RX_FIFO_CNT	RO	The valid data entries number in RX FIFO. It will be cleared via disabling the I2C module (MODULE_EN to 0) or an ERROR_ABORT event occurs.	0x0
31:3 RSVD_31_3	RO	Reserved	0x0

38.3.22 I2C Fast and Standard Spike Suppression Counter Register (I2C_FSTD_SPKCNT)

38.3.22.1 Offset

Register	Offset
I2C_FSTD_SPKCNT	0x54

38.3.22.2 Function

The register writing is valid only when the I2C module is disabled. This register must be programmed with a minimum value of 1. When speed mode is fast mode plus, this register must be programmed with a minimum value of 2.

38.3.22.3 Fields

Field	Access	Description	Reset Value
7:0 FSTD_SPKLEN	RW	Spike length counter for fast and standard mode. SDA or SCL input keeps stable until the spike counter reaches the configured values. Then, the internal synchronous SDA and SCL are updated with the input value, and the counter is reset and stopped. SDA or SCL changes again before the spike counter reaches its count limit value. When this happens, the spike counter is reset and stopped, the internal synchronous SDA and SCL are not updated with the input value.	0x5
31:8 RSVD_31_8	RO	Reserved	0x0

38.3.23 I2C High Speed Spike Suppression Counter Register (I2C_HS_SPKCNT)

38.3.23.1 Offset

Register	Offset
I2C_HS_SPKCNT	0x58

38.3.23.2 Function

The register writing is valid only when the I2C module is disabled. This register must be programmed with a minimum value of 1.

38.3.23.3 Fields

Field	Access	Description	Reset Value
7:0 HS_SPKLEN	RW	Spike length counter for high speed mode. SDA or SCL input keeps stable until spike counter reaches this configure value. The internal synchronous SDA or SCL is updated with the input value, and the counter is reset and stopped. SDA or SCL changes again before spike counter reaches its count limit value. When this happens, spike counter is reset and stopped, the internal synchronous SDA or SCL is not updated with the input value.	0x1
31:8 RSVD_31_8	RO	Reserved	0x0

38.3.24 I2C Interrupt Status Register (I2C_STATUS0)

38.3.24.1 Offset

Register	Offset
I2C_STATUS0	0x5C

38.3.24.2 Fields

Field	Access	Description	Reset Value
0 GENERAL_CALL	W1C	This bit will be set only when an I2C general call address is received, and it is acknowledged. This bit is set to 0 just after writing 1 to the I2C_STATUS0[GENERAL_CALL].	0x0
1 TXFIFO_O VERUN	W1C	This bit will be set when TXFIFO goes above TX FIFO depth and the CPU attempts to issue another I2C command by writing to the I2C_COMMAND_DATA register. When I2C is disabled, this bit keeps its level until the I2C master or slave state machines go into idle.	0x0
2 ERROR_A BORT	W1C	When this bit is set to 1, the I2C_ERROR_STATUS register indicates why the transmit abort takes place.	0x0
3 ACTIVITY	W1C	This bit captures the I2C module activity and stays set until it is cleared. There are four ways to clear it: • Disable the I2C module • Write 1 to the I2C_STATUS0[ACTIVITY] • Write 0 to the I2C_INT_ENABLE[I2C_ACTIVITY_DET_IE] • System reset	0x0
4 STOP_DET ECT	W1C	Indicates whether there is a stop event occurring on the I2C bus.	0x0

Field	Access	Description	Reset Value
5 START_DETECT	W1C	Indicates whether there is a start or restart event occurring on the I2C bus.	0x0
21:6 RSVD_21_6	RO	Reserved	0x0
22 SLAVE_READ_REQ	W1C	This bit will be set to 1 when I2C acts as a slave transmitter. I2C holds the I2C bus in a wait state (SCL=0) until the I2C slave read request interrupt is serviced. CPU must respond to this interrupt and then write the requested data to the I2C_COMMAND_DATA register. Therefore, this bit is set to 0 just after writing 1 to the I2C_STATUS0[SLAVE_READ_REQ].	0x0
23 RXFIFO_DONE	W1C	Only valid for I2C as a slave-transmitter. This bit is set to 1 if the master does not acknowledge a transmitted byte. This occurs on the last byte of the transmission, indicating that the transmission is done.	0x0
24 RXFIFO_UNDERUN	W1C	Set if any read-option to RX FIFO when it is empty from the I2C_COMMAND_DATA register. If the module is disabled (I2C_CONFIG0[0]=0), this bit keeps its level until the master or slave state machines go into idle. When I2C enable goes to 0, this interrupt is cleared.	0x0
25 RXFIFO_OVERUN	W1C	Set if RX FIFO is completely filled above I2C Rx buffer depth=4. The I2C module acknowledges this, but any data bytes received after the FIFO is full are lost. If the module is disabled (I2C_CONFIG0[0]=0), this bit keeps its level until the master or slave state machines go into idle, and when I2C enable goes to 0, this interrupt is cleared.	0x0
26 RESTART_DET	W1C	Indicates whether a re-start condition has occurred on the I2C interface when I2C module is operating in slave mode and the slave is the addressed slave.	0x0
27 SCL_STUCK_AT_LOW	W1C	Indicates SCL line is stuck at low.	0x0
29:28 RSVD_29_28	RO	Reserved	0x0
30 RXFIFO_FULL	RO	Set when RX FIFO reaches or goes above (RXFIFO_WATER_MARK +1). It is automatically cleared by hardware when the buffer level goes below the threshold. This bit is cleared once I2C_CONFIG0[0] is set to 0, regardless of any activity that continues.	0x0
31 TXFIFO_EMPTY	RO	When TXFIFO_EMPTY_EN = 0: This bit is set to 1 when the transmit buffer is at or below the value set in the I2C_TXFIFO_WATER_MARK register. When TXFIFO_EMPTY_EN = 1: This bit is set to 1 when the transmit buffer is at or below the value set in the I2C_TXFIFO_WATER_MARK register and the transmission of the address or data for the most recently popped command is completed.	0x0

38.3.25 I2C Status Register (I2C_STATUS1)

38.3.25.1 Offset

Register	Offset
I2C_STATUS1	0x60

38.3.25.2 Fields

Field	Access	Description	Reset Value
0 MASTER_IS_ACTIVE	RO	When master is not under the idle bit, this bit will be set to 1. 0b: MASTER FSM is idle 1b: MASTER FSM is not idle	0x0
1 SLAVE_IS_ACTIVE	RO	When slave is not under idle bit, this bit will be set to 1. 0b: SLAVE FSM is idle 1b: SLAVE FSM is not idle	0x0
2 TXFIFO_EMPTY_MASTER_HOLD	RO	This bit indicates that TX FIFO is empty, and the previously transferred command does not have a stop bit, so the master holds the bus.	0x0
3 TXFIFO_EMPTY_SLAVE_HOLD	RO	This bit indicates when the TX FIFO is empty, the slave will hold the bus until the slave TX FIFO has data to transmit for the read request.	0x0
4 RXFIFO_FULL_MASTER_HOLD	RO	This bit indicates the RX FIFO is full and receives new data, so the master will hold the bus.	0x0
5 RXFIFO_FULL_SLAVE_HOLD	RO	This bit indicates the RX FIFO is full and receives new data, so the slave will hold the bus.	0x0
6 TXFIFO_IS_NOT_FULL	RO	TX FIFO is not full flag 1 means the TX FIFO still has one or more empty locations.	0x1
7 TXFIFO_IS_EMPTY	RO	TXFIFO is empty flag 1 means the TX FIFO is completely empty.	0x1
8 RXFIFO_IS_NOT_EMPTY	RO	RX FIFO is not empty When RX FIFO already has one or more entries, this bit will be set to 1.	0x0
9 RXFIFO_IS_FULL	RO	RX FIFO is full flag When the RX FIFO is completely full, this bit will be set to 1.	0x0
10 SDA_ERROR_RECOVER_STUCK_LOW	RO	SDA error recovers from being stuck at low phase This bit is set when SDA failed to recover, and is still stuck at low phase.	0x0
11 SLAVE_IS_DISABLED_UNDER_ACTIVE	RO	Slave receiver is disabled under active state. When read as 1, it means slave is potentially disabled (I2C enable changes from 1 to 0) during slave transfer phase.	0x0
12	RO	Slave receives data discard.	0x0

Field	Access	Description	Reset Value
SLAVE_RX_DATA_DISCARD		When read as 1, it means I2C slave is aborted with at least one data byte from I2C transfer due to I2C_CONFIG0[0] from 1 to 0.	
13 I2C_IS_ENABLED	RO	I2C enable flag. 0b: I2C is disabled. 1b: I2C is enabled.	0x0
31:14 RSVD_31_14	RO	Reserved	0x0

38.3.26 I2C Error Status Register (I2C_ERROR_STATUS)

38.3.26.1 Offset

Register	Offset
I2C_ERROR_STATUS	0x64

38.3.26.2 Fields

Field	Access	Description	Reset Value
0 ERR_GEN_CALL_NO_ACK	RO	1b: I2C module in master mode sent a General Call, and no slave on the bus acknowledged the General Call.	0x0
1 ERR_GEN_CALL_READ	RO	1b: I2C module in master mode sent a General Call, but the user programmed the byte following the General Call to be a read from the bus (I2C_COMMAND_DATA[9] is set to 1).	0x0
2 ERR_START_BYTE_ACK_DET	RO	1b: Master has sent a START Byte, and the START Byte was acknowledged (wrong behavior).	0x0
3 ERR_SBYTE_NORSTRT	RO	To clear Bit 3, the source of the ERR_SBYTE_NORSTRT must be fixed first. Restart must be enabled (I2C_CONFIG1[2]=1), the MASTER_CMD_SEL bit must be 0 or 1 (I2C_DEST_ADDR[11:10]=00 or I2C_DEST_ADDR[11:10]=11). Once the source of the ERR_SBYTE_NORSTRT is fixed, this bit can be cleared in the same manner as other bits in this register. However, if the source of the ERR_SBYTE_NORSTRT is not fixed before attempting to clear this bit, bit 3 clears for one cycle and then gets re-asserted. 1b: The restart is disabled (MASTER_RESTART_EN bit (I2C_CONFIG1[2]) = 0), and the user is trying to send a START Byte.	0x0
4 ERR_H_NO_RSTRT	RO	1b: The restart is disabled (MASTER_RESTART_EN bit	0x0

Field	Access	Description	Reset Value
		(I2C_CONFIG1[2]) = 0), and the user is trying to use the master to transfer data in high-speed mode.	
5 ERR_H_MCODE_ACK_DET	RO	1b: Master is in high-speed mode, and the high-speed master code was acknowledged (wrong behavior).	0x0
6 ERR_7BIT_ADDR_NO_ACK	RO	1b: Master is in 7-bit addressing mode and the address sent was not acknowledged by any slave.	0x0
7 ERR_10BIT_ADDR1_NO_ACK	RO	1b: Master is in 10-bit address mode, and the first 10-bit address byte was not acknowledged by any slave.	0x0
8 ERR_10BIT_ADDR2_NO_ACK	RO	1b: Master is in 10-bit address mode, and the second address byte of the 10-bit address was not acknowledged by any slave.	0x0
9 ERR_10BIT_READ_NO_RSTRT	RO	1b: Will assert 1 when read 10-bit under MASTER_RESTART_EN=0.	0x0
10 ERR_DATA_NO_ACK	RO	1b: This is a master-mode only bit. Master has received an acknowledgment for the address. However, when it sent data byte(s) following the address, it did not receive an acknowledgement from the remote slave(s).	0x0
11 ERR_MASTER_LOST	RO	1b: Master has lost arbitration, or if I2C_ERROR_STATUS[13] is also set, then the slave transmitter has lost arbitration.	0x0
12 ERR_MASTER_DIS	RO	1b: Users try to initiate a master operation with the master mode disabled.	0x0
13 ERR_SLAVE_ARBLOST	RO	1b: Slave lost the bus while transmitting data to a remote master. I2C_ERROR_STATUS[11] is set at the same time.	0x0
14 ERR_MASTER_ABRT	RO	This is a master-mode-only bit. Master has detected the transfer abort (I2C_CONFIG0[4]).	0x0
15 ERR_SLAVE_READ_REQ	RO	1b: When the processor side responds to a slave mode request for data to be transmitted to a remote master and the user writes a 1 in CMD_MASTER_DIRECTION (bit 8) of I2C_COMMAND_DATA register, this will be set.	0x0
16 ERR_SLAVE_FLUSH_TXFIFO	RO	1b: Slave has received a read command, and some data exists in the TX FIFO so the slave issues an ERROR_ABORT interrupt to flush old data in TX FIFO.	0x0
17 ERR_SDA_LOW_TIMEOUT	RO	1b: SDA is stuck at low and timeout error.	0x0
20:18 ERR_TXFIFO_FLUSH_CNT	RO	This field indicates the number of TX FIFO data commands that are flushed due to the ERROR_ABORT interrupt. It	0x0

Field	Access	Description	Reset Value
		is cleared whenever I2C is disabled.	
31:21 RSVD_31_21	RO	Reserved	0x0

38.3.27 I2C SCL Stuck at Low Timeout Register (I2C_SCL_LOW_TIMEOUT)

38.3.27.1 Offset

Register	Offset
I2C_SCL_LOW_TIMEOUT	0x68

38.3.27.2 Function

This register is used to configure SCL stuck at low timeout. This register can be written only when I2C module is disabled.

38.3.27.3 Fields

Field	Access	Description	Reset Value
31:0 SCL_LOW_TIMEOUT	RW	I2C module generates the interrupt to indicate SCL stuck at low if it detects the SCL stuck at low for the I2C_SCL_LOW_TIMEOUT in units of I2C function clock period.	0xFFFFFFFF

38.3.28 I2C SDA Stuck at Low Timeout Register (I2C_SDA_LOW_TIMEOUT)

38.3.28.1 Offset

Register	Offset
I2C_SDA_LOW_TIMEOUT	0x6C

38.3.28.2 Function

This register is used to configure SDA stuck at low timeout. This register can be written only when I2C module is disabled.

38.3.28.3 Fields

Field	Access	Description	Reset Value
31:0 SDA_LOW_TIMEOUT	RW	I2C module initiates the recovery of the SDA line by enabling the SDA_RECOVER_EN (I2C_CONFIG0[3]) register bit if it detects the SDA stuck at low for the I2C_SDA_LOW_TIMEOUT in units of the I2C function clock period.	0xFFFFFFFF

38.3.29 I2C Read Clear Error Status Register

(I2C_RD_CLR_ERR_STATUS)

38.3.29.1 Offset

Register	Offset
I2C_RD_CLR_ERR_STATUS	0x70

38.3.29.2 Function

This register is to read to clear the I2C_ERROR_STATUS register.

38.3.29.3 Fields

Field	Access	Description	Reset Value
0 CLR_ERR	RO	Reading this register will clear all the I2C_ERROR_STATUS bits.	0x0
31:1 RSVD_31_1	RO	Reserved	0x0

38.4 Function description

This section provides the functional information of the Inter-Integrated Circuit (I2C) module.

38.4.1 I2C protocol behavior

38.4.1.1 START and STOP generation

When I2C works as a master, firstly put data into TX FIFO then launch an I2C start condition. Default SDA and SCL lines are high through external pull-up resistors on the I2C bus. When the start condition occurs, SDA changes from high to low when SCL keeps high. If Tx data sets the stop command from I2C_COMMAND_DATA[9] as high, I2C will generate a stop condition after TX FIFO is empty. Otherwise, I2C will hold on busy state and not generate a stop condition even though its TX FIFO is empty.

When the stop condition occurs, the SDA line changes from low to high when the SCL line keeps high.

When I2C works as a slave, I2C does not generate start or stop condition. When an I2C slave read request occurs, SCL will keep low until new data arrives.

Figure 38-2 shows the detailed I2C start and stop condition.

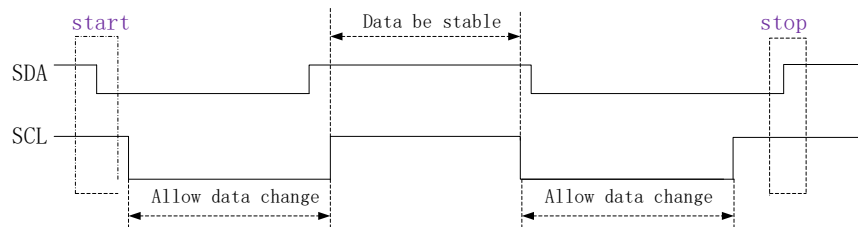


Figure 38-2 I2C start and stop condition

38.4.1.2 7-bit/10-bit slave address formats

There are two I2C slave address formats, 7-bit, and 10-bit.

For the I2C 7-bit address mode, the first MSB seven bits (bit[7:1]) of the first byte are I2C slave addresses. And LSB (bit0) is the transfer direction. See the details in Figure 38-3.

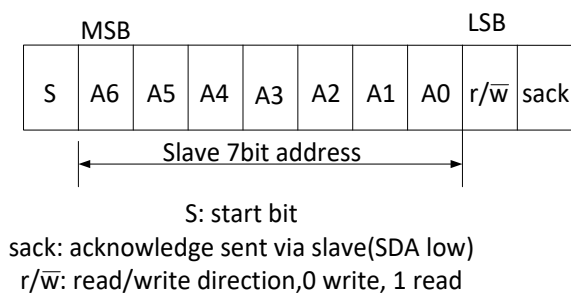


Figure 38-3 7-bit address format

For I2C 10-bit address mode, the master will transfer two bytes, as shown in Figure 38-4.

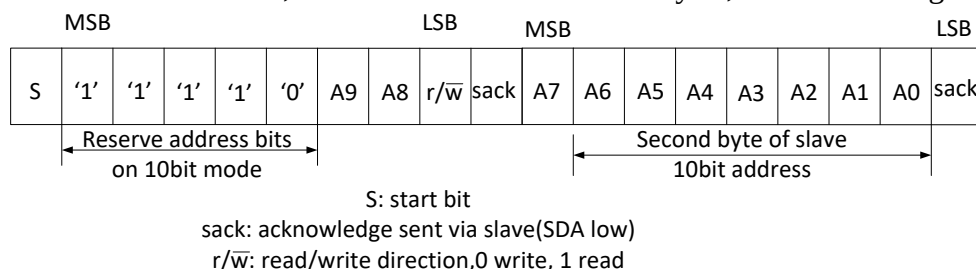


Figure 38-4 10-bit address format

NOTE

I2C also supports mixed transfer like read and write combined formats on either 7-bit or 10-bit address mode. But it could not support a 7-bit address transfer followed by a 10-bit address, or vice versa.

When I2C_CONFIG1[MASTER_RESTART_EN] =1, and I2C works as a master, it will launch a mixed transfer if transfer direction changes and TXFIFO is not empty.

38.4.1.3 Start byte transfer on master mode

If there is no onboard dedicated I2C hardware module in the system, a start byte protocol will be needed. When I2C works as a slave, it always samples the I2C bus with the highest speed supported. Therefore, it never requires a start byte transfer. But when I2C works as a master, if a slave device has such a requirement, it supports transferring a start byte '8'b0000_0001' at the beginning of each transfer. This is because it allows the processor that is polling the bus to under-sample the address phase until SDA bus 0 is detected. Once detecting a 0, it switches to the correct rate of the master from the under-sampling rate.

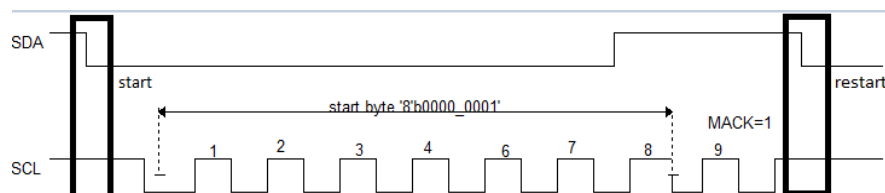


Figure 38-5 Master start byte transfer

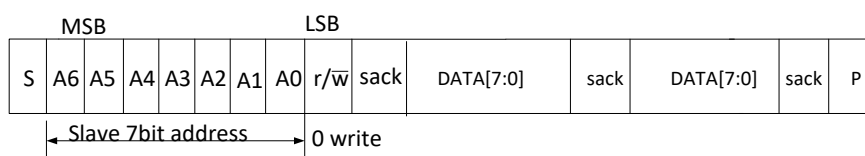
38.4.1.4 Transmission protocol

When I2C works as a master transmitter and transmits data to the slave device, all data is transmitted via byte, and no byte data is limited.

When the I2C master transmits its address and read/write direction bit to the slave device or transmits a data byte to the slave device, the slave device must acknowledge the master. If the slave could not send this acknowledgment, then the master will abort transfer via stop condition. Therefore, it requires the slave to keep SDA high to ensure the master can release the SDA bus and abort the current transfer.

Figure 38-6 shows the I2C 7-bit master transmitter writes data to its slave. The slave will give a slave acknowledge (sack, SDA is low) after each data byte received from the master.

7bit master transmit data to slave



S: start bit

sack: acknowledge sent via slave, SDA low

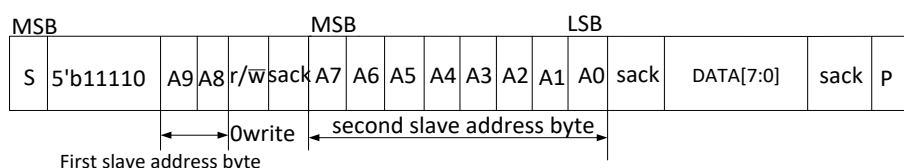
r/w: read/write direction, 0 write, 1 read

P: stop condition

Figure 38-6 7-bit master transmit data to slave

Figure 38-7 shows the I2C 10-bit master transmitter writes data to its slave. A slave will give a slave acknowledge (sack, SDA is low) after each data byte received from the master.

10bit Master transmit data to slave



S: start bit

sack: acknowledge sent via slave, SDA low

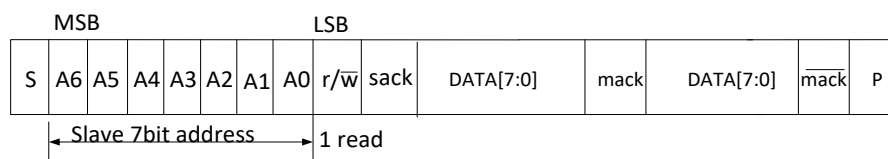
r/w: read/write direction, 0 write, 1 read

P: stop condition

Figure 38-7 10-bit master transmit data to slave

When I2C works as a master receiver, it will read data from the I2C slave device and generate a master acknowledge (mack, SDA is low) to slave after each byte received except for the last byte. This is the way the I2C master receiver tells the slave transmitter that is the last byte now. The slave-transmitter relinquishes the SDA line after detecting the No Acknowledge (NACK) so that the master can issue a STOP condition. See the details in Figure 38-8 and Figure 38-9.

7bit master receive data from slave



S: start bit

sack: acknowledge sent via slave, SDA low

mack: acknowledge sent via master, SDA low

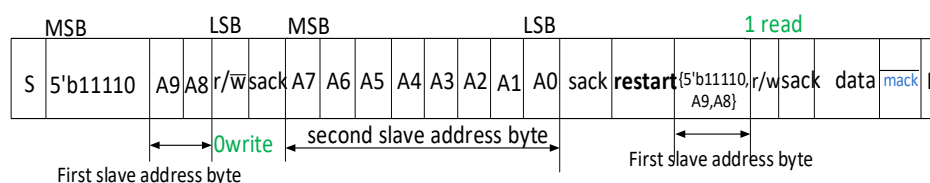
$\overline{\text{mack}}$: acknowledge sent via master, SDA high

r/w: read/write direction, 0 write, 1 read

P: stop condition

Figure 38-8 7-bit master receive data from slave

10bit Master receive data from slave



S: start bit

sack: acknowledge sent via slave, SDA low

$\overline{\text{mack}}$: acknowledge sent via master, SDA high

r/w: read/write direction, 0 write, 1 read

P: stop condition

Figure 38-9 10-bit master receive data from slave

38.4.1.5 I2C stop and restart command function

I2C has a stop and restart command from the I2C_COMMAND_DATA register. A stop condition will occur after transfer if data has a stop command set.

When I2C works as a master transmitter, I2C keeps a busy state and holds SCL as low after its TXFIFO becomes empty if no stop command is set. And it will keep SCL low until there is new data filled into TX FIFO.

Figure 38-10 shows the I2C master transmits data with stop =1 when TX FIFO is empty.

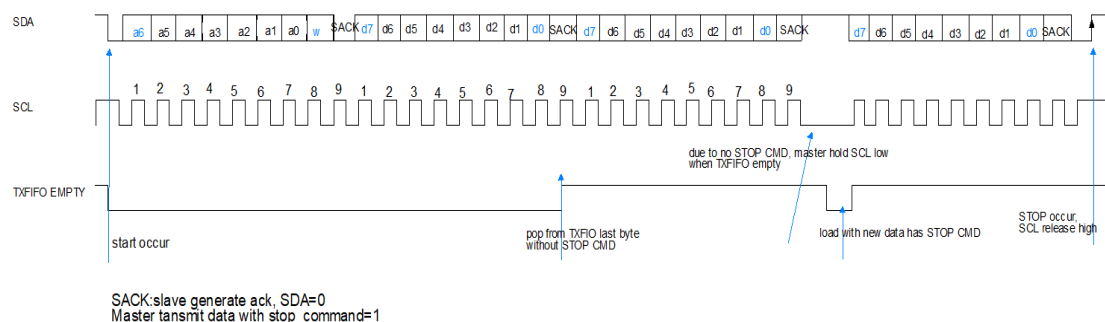


Figure 38-10 Master transmit data with stop = 1, when TXFIFO is empty

Figure 38-11 shows the I2C master receives data from slave with stop =1, when TX FIFO is empty.

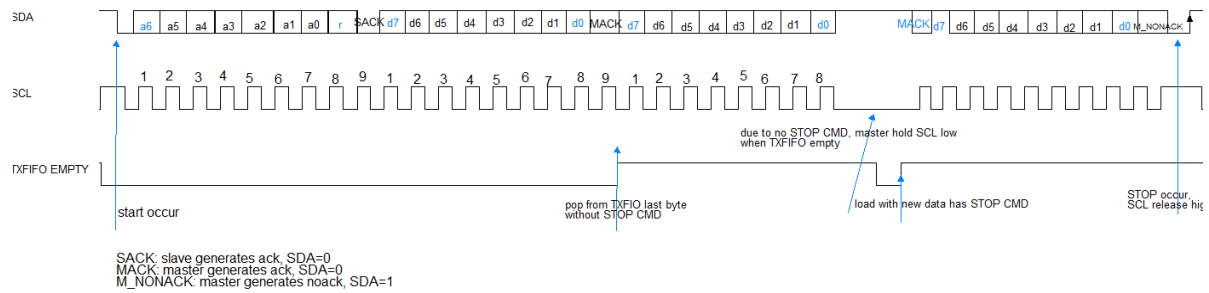


Figure 38-11 Master receive data with stop = 1, when TXFIFO is empty

Figure 38-12 shows I2C master transmits data with re-start command=1, when TXFIFO is not empty.

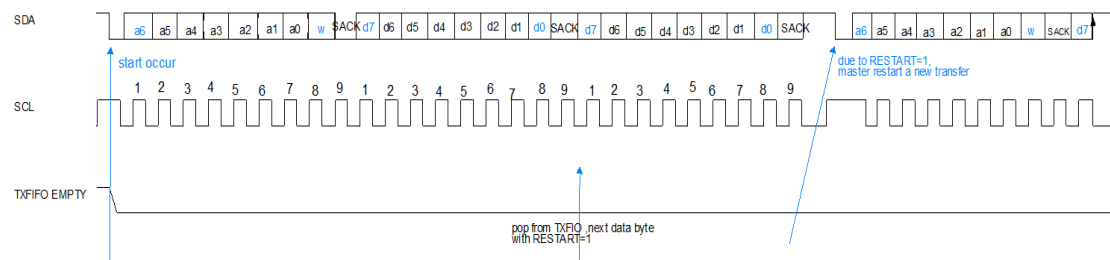


Figure 38-12 Master transmit data with re-start = 1, when TXFIFO is not empty

Figure 38-13 shows I2C master receives data with re-start command=1, when TXFIFO is not empty.

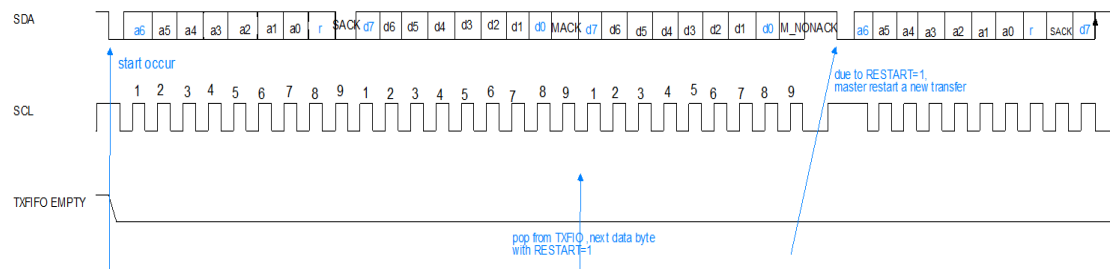


Figure 38-13 Master receive data with re-start = 1, when TXFIFO is not empty

Figure 38-14 shows I2C master transmits data with stop command=1, when TXFIFO is not empty.

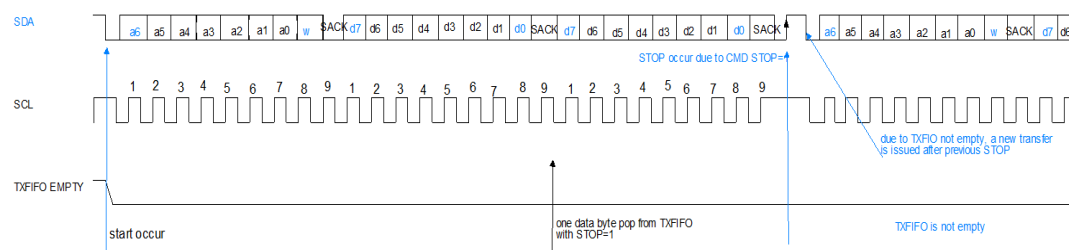


Figure 38-14 Master transmit data with stop = 1, TXFIFO is not empty

Figure 38-15 shows I2C master receives data with stop command=1, when TXFIFO is not empty.

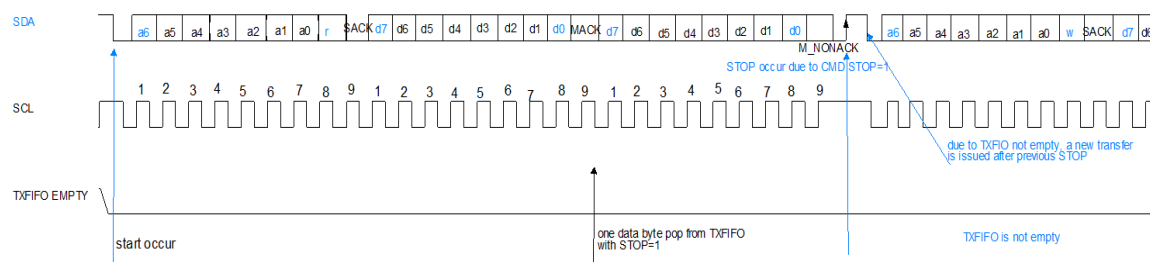


Figure 38-15 Master receive data with stop=1, when TXFIFO is not empty

Figure 38-16 shows I2C master transmits data with re-start command=1, when TXFIFO is empty after last transfer.

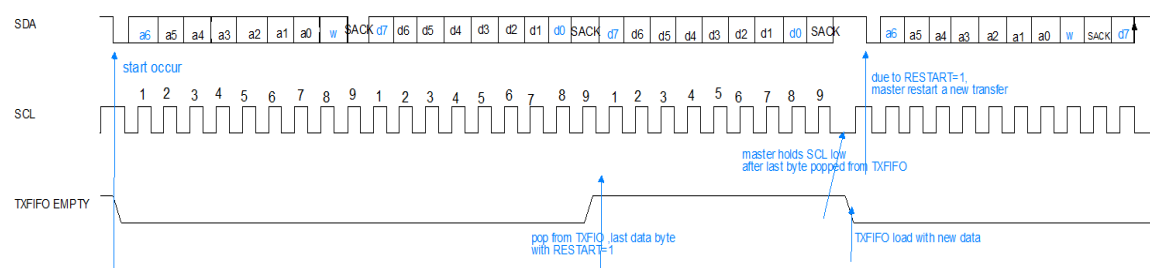


Figure 38-16 Master transmit data with re-start =1, allow TXFIFO is empty

Figure 38-17 shows I2C master receives data with re-start command=1, when TXFIFO is empty after last transfer.

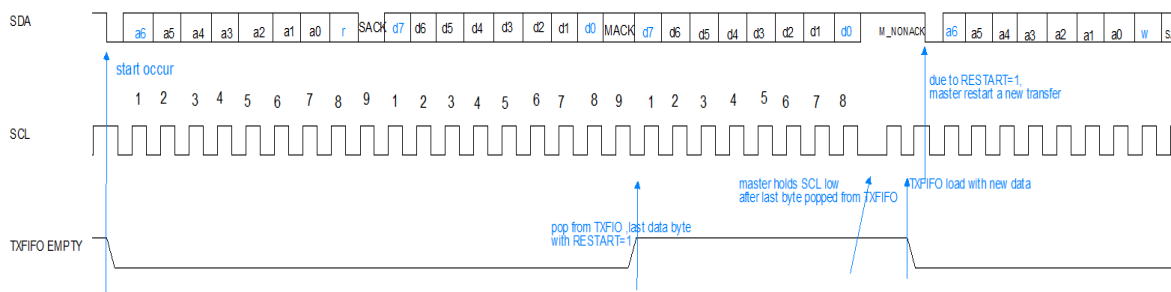


Figure 38-17 Master receive data with re-start =1, allow TXFIFO is empty

38.4.2 Multi-master arbitration and clock synchronization

If there are two or more I2C devices on the same I2C bus, arbitration is needed when these masters all want to control the I2C bus. For example, once a winner master has taken control of the SDA bus, other masters could not control the bus until this winner master sends the stop command and has released the SDA bus to an idle state.

Arbitration occurs on the SDA bus when the SCL line is 1. When another master transmits 0, the master which transmits 1 will lose bus control and turn off its data output stage. The master which loses arbitration can continue generating SCL until the end of the byte transfer. If both masters are communicating with the same slave, the arbitration could go into the data phase. Figure 38-18 shows more details about this multi-master arbitration phase.

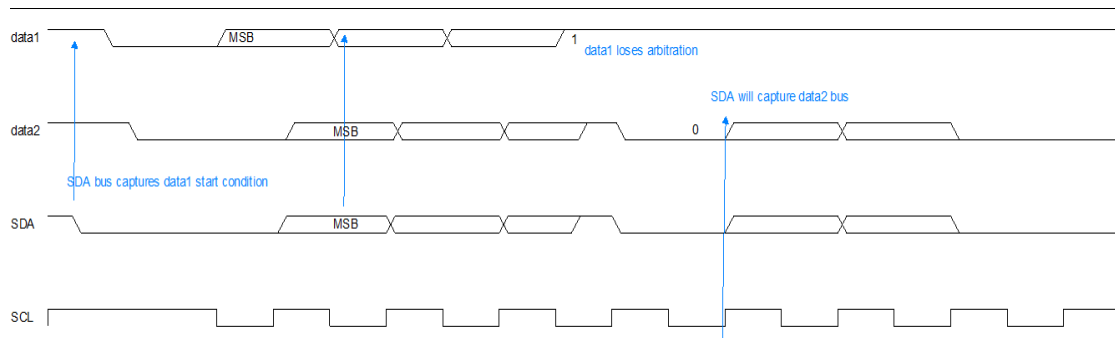


Figure 38-18 Multi-master arbitration

For the I2C SCL bus, if two or more masters attempt to control the SCL line, they must arbitrate and synchronize their SCL clock. All master data is valid only during the high period of SCL clock.

SCL bus is a wire-and connection. Once a master A transitions the SCL clock to 0, the SCL bus will become 0, and this master A starts counting the low time of its SCL clock and transitions the SCL clock signal to 1 at the beginning of the next clock period. Then if another master B is holding the SCL as 0, master A will go to a wait state until the SCL clock line transitions to 1. Figure 38-19 shows the details about this multi-master SCL synchronization.

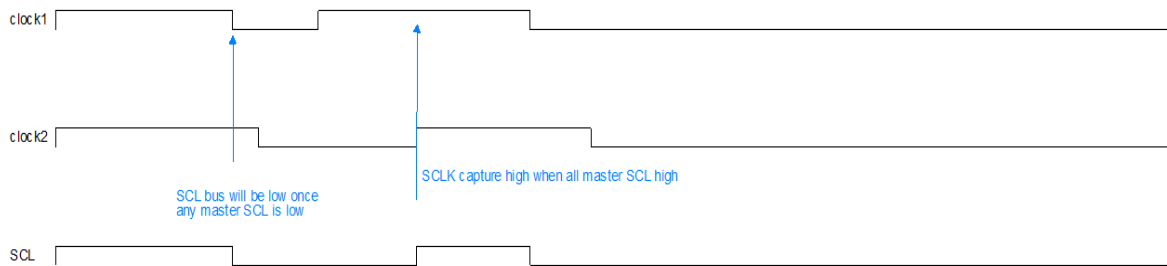


Figure 38-19 multi-master SCL synchronization

38.4.3 Bus recovering feature

This I2C supports recovering features for both SDA and SCL line when unexpected events occur in which either SDA or SCL is stuck at a low phase.

38.4.3.1 SDA recovers from being stuck at low

Considering conditions where the SDA line is stuck at a low phase, the I2C master has following steps to recover, as shown in Figure 38-20.

Master sends a maximum of 9 clock pulses to recover SDA from low to high within those 9 clocks. The number of clock pulses changes with the number of bits that remain to be sent by the slave.

- The master attempts to assert high '1' on the SDA line and check whether SDA bus is recovered. If SDA is not recovered, it will continue sending a maximum of 9 SCL clocks.
- If SDA is recovered within 9 clock pulses, then master will send the STOP command to release the bus.

If SDA is not recovered even after the 9th clock pulse, then I2C needs a hardware reset.

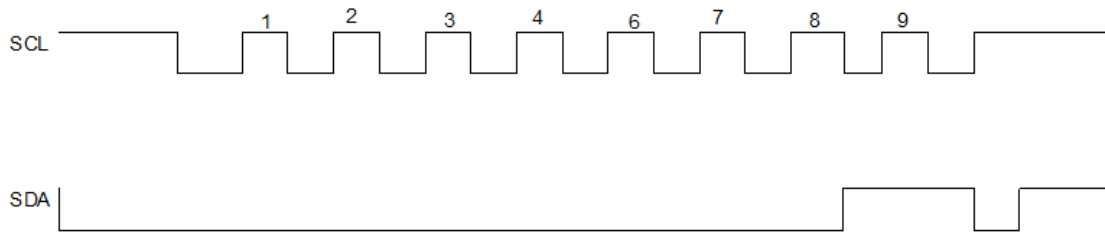


Figure 38-20 I2C SDA recover from stuck at low

For SCL stuck at a low event, there is no effective method but to use system reset to overcome this issue.

Detailed application steps about this SDA or SCL recover feature are like:

1. Default I2C is disabled.
2. Write `I2C_CONFIG0[MASTER_BUS_RECOVER_EN]=1` to enable I2C bus recover feature.
3. Write `I2C_SDA_LOW_TIMEOUT` and `I2C_SCL_LOW_TIMEOUT` registers.
4. Write `I2C_CONFIG0[MODULE_EN] =1` to enable the I2C module.
5. Polling check `I2C_STATUS0[SCL_STUCK_AT_LOW]`. If value=1, it indicates that SCL is stuck at low and needs a system reset to recover SCL.
6. Polling check `I2C_ERROR_STATUS[ERR_SDA_LOW_TIMEOUT]`. If value=1, it indicates SDA stuck at low phase now will generate an interrupt if enabled. Write `I2C_CONFIG0[SDA_RECOVER_EN] =1` to enable the SDA recover feature.
7. Check `I2C_STATUS1[SDA_ERR_RECOVER_STUCK_LOW]`. If value=1, it indicates that SDA was not recovered from low, and it needs reset here. If value=0, it indicates that SDA is recovered from low now. Read `I2C_RD_CLR_ERR_STATUS` to clear the `ERR_SDA_LOW_TIMEOUT` flag.

38.4.4 Spike filter counter

I2C has spike filter counter logic to monitor input bus SDA/SCL signal, checking if they keep stable under programmable counter values of I2C functional clock cycles before they are sampled internally. SDA and SCL have separate spike filter counter registers. Each counter will be started whenever the input SDA or SCL signal changes.

If the input signal keeps stable until the filter counter reaches its count limit value, when this occurs, the internal bus signal is updated with the input value, and the spike filter counter is reset and stopped. The counter will be restarted again once a new change on the input signal is detected.

If the input signal changes again before the spike filter counter reaches its count limit value, the filter counter is reset and stopped, and the internal version of the signal is not updated.

Figure 38-21 shows the above conditions.

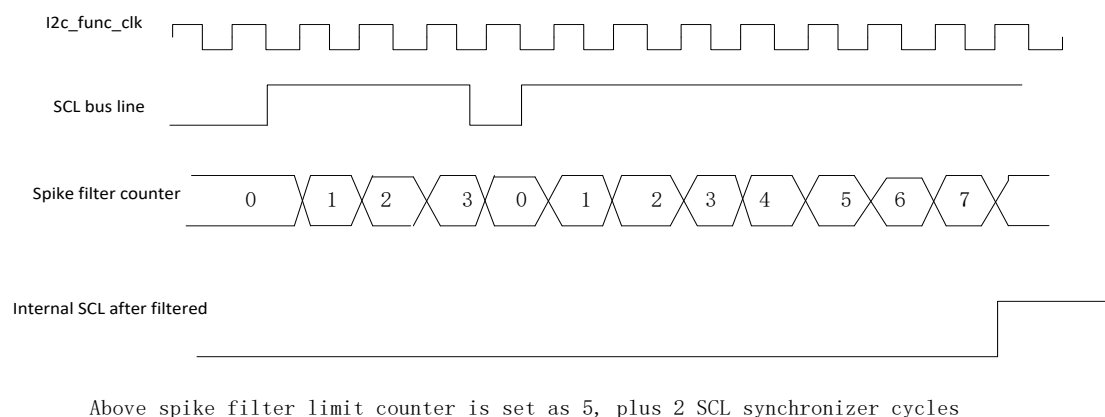


Figure 38-21 Spike filter counter feature

38.4.5 SDA hold timing

According to the I2C protocol specification, it requires a 300ns of hold time delay on the SDA signal for standard mode and fast mode, and a long enough hold time for high-speed mode. I2C has a software programmable register (I2C_SDA_HOLD_TIMING) to configure SDA hold-time when encountering differing board level delays.

The I2C_SDA_HOLD_TIMING[15:0] is used to control the hold time of SDA during transmission (after SCL goes from HIGH to LOW). The bits I2C_SDA_HOLD_TIMING[23:16] are used to extend the SDA transition whenever SCL is high for the I2C receiver.

38.4.5.1 SDA hold timing for receiver mode

When I2C works as a receiver, according to the I2C protocol, the device should internally hold the SDA line to bridge the undefined gap between logic 1 and logic 0 of SCL.

- SDA_RX_HOLD_TIMING stands for counting the I2C function clock cycles. The minimum value of SDA_RX_HOLD_TIMING is 0. This hold time is valid only when SCL is high. The receiver does not extend the SDA after SCL goes LOW internally.
- Figure 38-22 shows I2C works as a receiver with SDA_RX_HOLD_TIMING set greater than or equal to 3

If SDA_RX_HOLD_TIMING is greater than 3, I2C will not hold SDA beyond 3 function cycles, after internal SCL goes low.

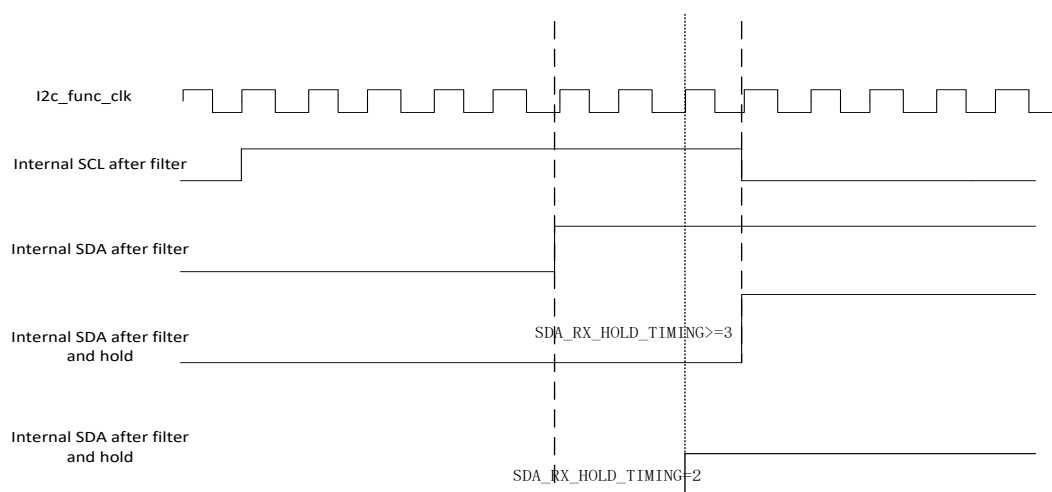


Figure 38-22 I2C SDA hold timing when working as a receiver

The maximum SDA_RX_HOLD_TIMING values on master mode are shown as following:

- Standard mode: I2C_STD_SCL_HCNT - I2C_FSTD_SPKCNT - 3
- Fast mode: I2C_FST_SCL_HCNT - I2C_FSTD_SPKCNT - 3
- High speed(cap=100pF): Min {I2C_FST_SCL_HCNT - I2C_FSTD_SPKCNT - 3, I2C_HS_SCL_LCNT - I2C_HS_SPKCNT - 3}
- High speed(cap=400pF): Min {I2C_FST_SCL_HCNT - I2C_FSTD_SPKCNT - 3, (I2C_HS_SCL_LCNT/2) - I2C_HS_SPKCNT - 3}

For the slave receiver mode, make sure the SDA_RX_HOLD_TIMING does not exceed the maximum SCL fall time.

38.4.5.2 SDA hold timing for transmit mode

When I2C works as a master transmitter, the minimum TX hold timing is one I2C function clock cycle. Even setting SDA_TX_HOLD_TIMING as 0, I2C still keeps one I2C function after internal SCL rising edge. Therefore, the SDA bus will always be driven at SDA_TX_HOLD_TIMING cycles after rising SCL edge.

When I2C works as a slave transmitter, the minimum TX hold timing is *_SPKLEN + 7 I2C function clock. Thus, *_SPKLEN is equal to I2C_FSTD_SPKCNT for standard and fast modes, and *_SPKLEN is equal to I2C_HS_SPKCNT for high-speed mode. Even when SDA_TX_HOLD_TIMING is less than (*_SPKLEN + 7), I2C drives SDA at (*_SPKLEN + 7) I2C function clock cycles after the rising edge of SCL.

38.4.6 I2C SCL configuration

For I2C works as a master, it must meet the following conditions:

- I2C_STD_SCL_LCNT and I2C_FST_SCL_LCNT register values must be larger than (I2C_FSTD_SPKCNT + 7), considering the time required for driving SDA after the falling edge of SCL.
- I2C_STD_SCL_HCNT and I2C_FST_SCL_HCNT register values must be larger than (I2C_FSTD_SPKCNT + 5), considering the time required for sampling SDA after the rising edge of SCL.

If high speed is set, the following conditions must be met:

- I2C_HS_SCL_LCNT register values must be larger than (I2C_HS_SPKCNT + 7).
- I2C_HS_SCL_HCNT register values must be larger than (I2C_HS_SPKCNT + 5).

The number of I2C function clock cycles equals (I2C_*_SCL_LCNT+1) for SCL actual low time phase. And the number of I2C function clock cycles equals (I2C_*_SCL_HCNT+*_SPKLEN + 7) for SCL actual high time phase.

The reasons for SCL high phase periods are:

- Counting logic for SCL high counters (I2C_*_SCL_HCNT + 1).
- Digital filtering for SCL line includes a delay of (*_SPKLEN+2) function clock cycles.
- Whenever SCL changes from 1 to 0, I2C also includes a delay of fixed 3 function cycles to complete the SCL high time.

The following example lists how to calculate this minimum I2C function clock cycle for standard mode, fast mode, and high-speed mode. (The following calculation does not consider SCL rise and fall time).

In the fast mode, the maximum data rate is 400kbps. So, the minimum SCL period is 1/400k = 2500ns.

The minimum number of function cycles about SCL high phase is 14. According to protocol minimum time like min_scl_low_time_fs=1300ns and min_scl_high_time_fs=600ns, $I2C\_FST\_SCL\_LCNT * (2500 / (14 + I2C\_FST\_SCL\_LCNT)) = 1300ns$, I2C_FST_SCL_LCNT = 16.

The I2C function clock frequency is 2500ns / (14+16) = 83.3ns = 12M. For fast speed:

- The actual SCL low time periods are 16. We need to configure

- $I2C\_FST\_SCL\_LCNT=16-1=15$.
The actual SCL high time cycles are 14. We need to configure $I2C\_FST\_SCL\_HCNT=14-8=6$.

Note 1:

- *_SPKLEN is:
 - FSTD_SPKLEN
 - HS_SPKLEN
- I2C_*_SCL_LCNT is:
 - I2C_STD_SCL_LCNT
 - I2C_FST_SCL_LCNT
 - I2C_HS_SCL_LCNT
- I2C_*_SCL_HCNT is:
 - I2C_STD_SCL_HCNT
 - I2C_FST_SCL_HCNT
 - I2C_HS_SCL_HCNT

Note 2:

$SCL\_High\_time = [(I2C\_*_SCL\_HCNT + IC\_*_SPKLEN + 7) * ic\_clk] + SCL\_Fall\_time$

$SCL\_Low\_time = [((I2C\_*_SCL\_LCNT + 1) * ic\_clk) - SCL\_Fall\_time + SCL\_Rise\_time]$

SCL rise and fall time parameters vary depending on external factors such as:

- Characteristics of IO driver
- Pull-up resistor value
- Total capacitance on SCL line, and so on

According to the minimum SCL low time and high time delay from the I2C bus specification, the following table lists the minimum number of I2C function clock cycles.

The following table shows the minimum number of I2C function clock cycles for different speed modes.

Table 38-3 I2C minimum SCL LCNT and HCNT configure

Speed Mode	Standard mode	Fast mode	Fast mode plus	High speed(400pf)	High speed(100pf)
SCL low time	4.7us	1.33us	500ns	333ns	161ns
SCL high time	5.2us	1.16us	500ns	274ns	132ns
I2C function clock frequency/Mhz	2.7M	12M	32M	51M	105.4M
spike filter length	1	1	2	1	1
SCL actual low phase	13	16	16	17	17
SCL actual high phase	14	14	16	14	14
SCL low counter configure val	12	15	15	16	16
SCL high counter configure val	6	6	7	6	6

38.5 I2C DMA feature

I2C has a separate DMA enable bit for TX and RX DMA transfer.

For the I2C transmitter, after enabling I2C_DMA_CTRL[TXFIFO_DMA_EN], TXFIFO DMA request is generated whenever the number of entries in TX FIFO is less than or equal to DMA_TXFIFO_WATERMARK. Then, it will request the DMA controller to fill data into I2C TX FIFO.

For the I2C receiver, after enabling I2C_DMA_CTRL[RXFIFO_DMA_EN], RXFIFO DMA request is generated whenever the number of entries in RX FIFO is greater than or equal to (DMA_RXFIFO_WATERMARK+1). Then, it will request the DMA controller to read data from I2C RX FIFO.

38.6 Initialization application information

In Master Mode:

- Set SLAVE_MODE_DIS to 1. The slave is disabled.
- Configure MASTER_RESTART_EN to enable/disable the restart mode.
- Configure MASTER_10BIT_ADDR_SEL to the 7-bit addressing or 10-bit addressing.
- Set address of target slave by writing it to I2C_DEST_ADDR.
- Configure the speed mode.
- Set MASTER_MODE_EN to 1. Then the master is enabled.
- Write to I2C_RXFIFO_WATER_MARK to set RX FIFO water mark.
- Write to I2C_TXFIFO_WATER_MARK to set TX FIFO water mark.
- Write 1 to MODULE_EN to enable I2C.

In Slave Mode:

- Set MASTER_MODE_EN to 0. Then the master is disabled.
- Configure SLAVE_10BIT_ADDR_SEL to the 7-bit addressing or 10-bit addressing.
- Write to SLV_ADDR to set own slave address.
- Configure the speed mode.
- Set SLAVE_MODE_DIS to 0. Then the slave is enabled.
- Write to I2C_RXFIFO_WATER_MARK to set RX FIFO water mark.
- Write to I2C_TXFIFO_WATER_MARK to set TX FIFO water mark.
- Write 1 to MODULE_EN to enable I2C.

Chapter 39

Inter-IC Sound (I2S)

39.1 Introduction

The Inter-IC Sound (I2S) Bus is developed by Philips. It is a simple three-wire serial bus protocol that transfers stereo audio data. The bus only handles audio data transmission; any control and subcoding signals transferred need to use a different bus protocol (such as I2C).

39.1.1 I2S terminology

Following items can be referred throughout this manual, and they are defined in detail as following:

- SCLK/sclk-----serial/bit clock.
- WS/ws-----word select line.
- SD/sd -----serial data line.
- master-----I2S will generate WS and SCLK under master mode, based on its function clock and master divider settings.
- slave-----I2S will respond to external WS and SCLK signals under slave mode.
- transmitter-----sends out data to SD bus via SCLK and WS clock line. It can work under master or slave mode.
- receiver-----receives data from SD bus via SCLK and WS clock line. It can work under master or slave mode.

39.1.2 I2S overview

I2S bus consists of one sd line, one ws line and one sclk line. The sd line is time multiplexed and allows transferring two data streams like left and right stereo data. This I2S has four data channels for both transmit and receive operations.

The following Figure 39-1 shows simple configuration of I2S.

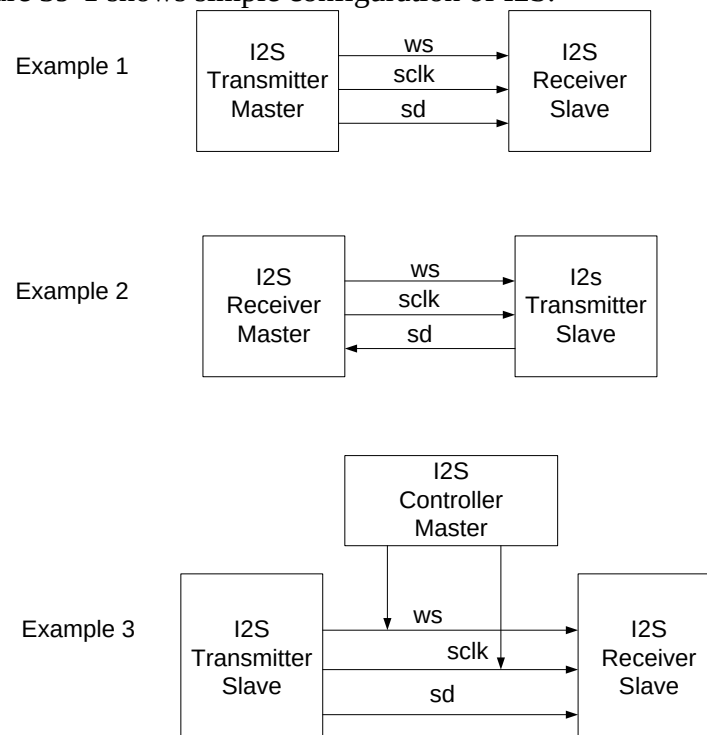


Figure 39-1 Simple configuration of I2S

Example 1 and example 2 show I2S master can work as either transmitter or receiver, which is the same as the I2S slave. The I2S master is responsible for generating sclk and ws line signals. In some conditions, there are several transmitters and receivers, and a separate I2S master that can be used. Example 3 shows I2S master generates shared ws and sclk signals, then other I2S slaves start transmissions based on the shared ws and sclk lines.

For transmitter, sd line is always sent via MSB first. Receiver and transmitter can have different word sizes. If word transferred is too large for receiver, the least significant bits (LSB) will be truncated. Otherwise, if word transferred is too short for receiver, MSB will be padded with zero.

The ws line is used for time multiplexed data transfer. When ws is equal to 0, the left stereo data is transferred. When ws is equal to 1, the right stereo data is transferred. Figure 39-2 shows standard I2S formats, the MSB of a word is sent one sclk cycle after ws changes. Serial data (sd) is sent via negedge of sclk signal. But for receiver, it should use posedge of sclk to latch the sd data.

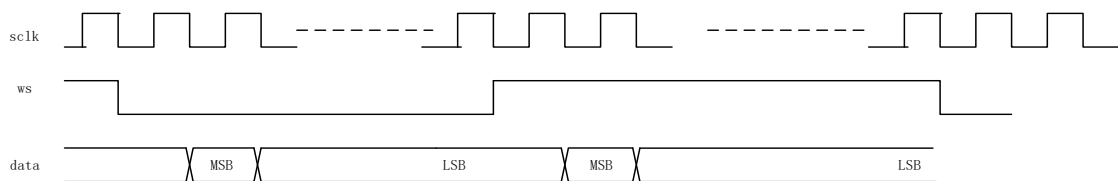


Figure 39-2 Simple configuration of I2S

39.1.3 Features

I2S supports following key features:

- I2S transmitter/receiver based on the Philips I2S serial protocol.
- Support 4 stereo data channels for both transmitter and receivers.
- I2S module only supports half duplex communication.
- Support master or slave option.
- Asynchronous APB clock and I2S function clock.
- Configurable function clock divider to achieve the expected sample rate.
- Configurable audio data resolutions of 12, 16, 20, 24, and 32 bits.
- Transmitter and Receiver FIFO depth is 16 words, word size is determined via data resolutions configuration.
- Configurable FIFO threshold level.
- Support cyclic transmission during all enabled transmission channels.

39.1.4 Block diagram

The following figure is a block diagram of the Inter-IC Sound (I2S) module.

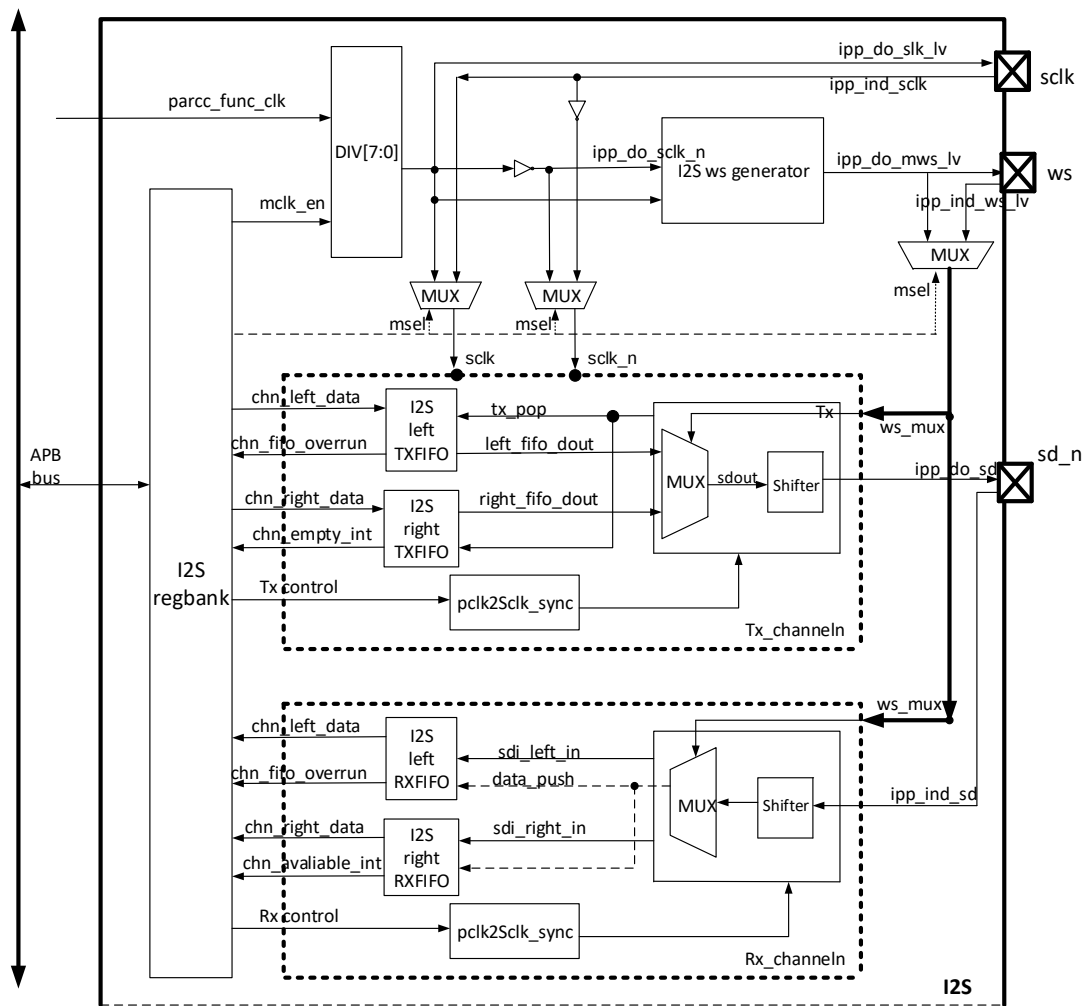


Figure 39-3 I2S diagram

39.1.5 Instantiation information

The following table lists the I2S instantiation information for each chip in the product series.

Configuration	K144MC	K146MC	K148MC
Instance(s)	I2S0	I2S0	I2S0
	-	-	I2S1

39.2 Memory map and register definition

This section provides the memory map and register definitions of the Inter-IC Sound (I2S) module.

39.2.1 I2S memory map

I2S0 base address: 0x40054000
I2S1 base address: 0x40055000

Table 39-1 I2S memory map

Offset	Register	Width (In bits)	Reset Value
0x00	I2S Control Register (I2S_CONTROL)	32	0x00000000
0x04	I2S Master Configuration Register (I2S_MASTER_CONFIG)	32	0x00000004
0x08 + 0x14*n (n=0~3)	I2S Channeln Left Data Register (I2S_CHn_LEFT_DATA)	32	0x00000000
0x0C + 0x14*n (n=0~3)	I2S Channeln Right Data Register (I2S_CHn_RIGHT_DATA)	32	0x00000000
0x10 + 0x14*n (n=0~3)	I2S Channeln Configuration Register (I2S_CHn_CFG)	32	0x000000B7
0x14 + 0x14*n (n=0~3)	I2S Channeln Interrupt Configuration Register (I2S_CHn_INT_CFG)	32	0x00000040
0x18 + 0x14*n (n=0~3)	I2S Channeln Trigger Level Configuration Register (I2S_CHn_TRIGGER_LEVEL_CFG)	32	0x00000033
0x58	I2S Cycle Tx Data (I2S_CYCLE_TX_DATA)	32	0x00000000
0x5C	I2S Cycle Tx Reset (I2S_CYCLE_TX_RESET)	32	0x00000000
0x60	I2S Cycle Rx Data (I2S_CYCLE_RX_DATA)	32	0x00000000
0x64	I2S Cycle Rx Reset (I2S_CYCLE_RX_RESET)	32	0x00000000

39.2.2 I2S Control Register (I2S_CONTROL)

39.2.2.1 Offset

Register	Offset
I2S_CONTROL	0x00

39.2.2.2 Function

This register is used to control the I2S block function.

39.2.2.3 Fields

Field	Access	Description	Reset Value
0 I2S_MODULE_EN	RW	I2S module enable Once clearing this bit, receiver or channel will be disabled and all FIFOs will be flushed. 0b: Disable I2S module. 1b: Enable I2S module.	0x0
1 I2S_TE	RW	I2S transmit enable bit Once clearing this bit to 0, all I2S transmitter channels will be disabled. 0b: Disable I2S transmitter. 1b: Enable I2S transmitter.	0x0
2 I2S_RE	RW	I2S receiver enable bit Once clearing this bit to 0, all I2S receiver channels will be disabled.	0x0

Field	Access	Description	Reset Value
		0b: Disable I2S receiver. 1b: Enable I2S receiver.	
3 I2S_TXFIFO_RESET	WO	I2S TXFIFO reset bit Writing 1 to this bit will reset all I2S Transmit FIFO contents. It can only be written when I2S_TE=0 and will clear itself after any written option to this bit. It is a self-clear bit. 0b: No effect. 1b: Reset all I2S transmit FIFO contents.	0x0
4 I2S_RXFIFO_RESET	WO	I2S RXFIFO reset bit Writing 1 to this bit will reset all I2S Receiver FIFO contents. It can only be written when I2S_RE=0 and will clear itself after any written option to this bit. It is a self-clear bit. 0b: No effect. 1b: Reset all I2S receiver FIFO contents.	0x0
31:5 RSVD_31_5	RO	Reserved	0x0

39.2.3 I2S Master Configuration Register (I2S_MASTER_CONFIG)

39.2.3.1 Offset

Register	Offset
I2S_MASTER_CONFIG	0x04

39.2.3.2 Function

This register is used to control the I2S master configuration.

39.2.3.3 Fields

Field	Access	Description	Reset Value
0 I2S_MASTER_EN	RW	I2S master mode enable bit, default value is 0, slave mode. 0b: Select slave mode 1b: Select master mode	0x0
2:1 I2S_NUM_BCLK	RW	I2S number of bclk cycles during left or right word line. It's valid only during I2S master mode. And any change of this configuration requires priorly setting I2S_MASTER_CFG[I2S_MASTER_EN] = 0. The number of bclk cycles should be equal or greater than tx/rx audio resolutions configured from individual channel configure register. 00b: 16 bclk cycles 01b: 24 bclk cycles 10b: 32 bclk cycles 11b: 32 bclk cycles	0x2
4:3 RSVD_4_3	RO	Reserved	0x0
7:5 I2S_NUM_BCLK_	RW	I2S gated after the number of bclk (bit clock) cycles.	0x0

Field	Access	Description	Reset Value
GATE		It is valid only during I2S master mode. And any change of this configuration requires priorly setting I2S_MASTER_CFG[I2S_MASTER_EN]=0. 0x00: No bclk gating 0x01: Bclk will be gated after 12 bclk cycles 0x02: Bclk will be gated after 16 bclk cycles 0x03: Bclk will be gated after 20 bclk cycles 0x04: Bclk will be gated after 24 bclk cycles 0x05,0x06,0x07: Reserved	
15:8 MCK_DIV	RW	I2S audio master clock divider, it will divide audio master clock to generate the expected I2S bit clock. Actual division value is (MCK_DIV+1)*2.	0x0
31:16 RSVD_31_16	RO	Reserved	0x0

39.2.4 I2S Channeln Left Data Register (I2S_CHn_LEFT_DATA)

39.2.4.1 Offset

Register	Offset
I2S_CHn_LEFT_DATA	0x08 + 0x14*n (n=0~3)

39.2.4.2 Function

The left stereo data received serially from the receive channel input is read through this register.

39.2.4.3 Fields

Field	Access	Description	Reset Value
31:0 LEFT_DATA	RW	When I2S works as a transmitter, it's a write only register. The channeln SD line is serially shifted based on this register content. The left stereo data is firstly written into I2S_CHn_LEFT_DATA, then written into I2S_CHn_RIGHT_DATA. Any data should be written into channeln TXFIFO when it's not full. When I2S works as a receiver, it's a read only register. The right stereo data received from channeln input SD line is read via I2S_CHn_RIGHT_DATA. When RXFIFO is full and CPU has not performed read access like (first read I2S_CHn_LEFT_DATA, then read I2S_CHn_RIGHT_DATA) before a new pair of stereo data arrives, new data will be lost and overrun interrupt will occur.	0x0

39.2.5 I2S Channeln Right Data Register (I2S_CHn_RIGHT_DATA)

39.2.5.1 Offset

Register	Offset
I2S_CHn_RIGHT_DATA	0x0C + 0x14*n (n=0~3)

39.2.5.2 Function

The right stereo data received serially from the receive channel input is read through this register.

39.2.5.3 Fields

Field	Access	Description	Reset Value
31:0 RIGHT_DATA	RW	When I2S works as a transmitter, it's a write only register. The channeln SD line is serially shifted based on this register content. The left stereo data is firstly written into I2S_CHn_LEFT_DATA, then written into I2S_CHn_RIGHT_DATA. Any data should be written into channeln TXFIFO when it's not full. When I2S works as a receiver, it's a read only register. The right stereo data received from channeln input SD line is read via I2S_CHn_RIGHT_DATA. When RXFIFO is full and CPU has not performed read access like (first read I2S_CHn_LEFT_DATA, then read I2S_CHn_RIGHT_DATA) before a new pair of stereo data arrives, new data will be lost and overrun interrupt will occur.	0x0

39.2.6 I2S Channeln Configuration Register (I2S_CHn_CFG)

39.2.6.1 Offset

Register	Offset
I2S_CHn_CFG	0x10 + 0x14*n (n=0~3)

39.2.6.2 Function

This register is used to control the channel function and channel audio resolution.

39.2.6.3 Fields

Field	Access	Description	Reset Value
0 TX_EN	RW	This bit controls channeln transmit feature, independent of other channels. Channeln transmission starts at time after TX_EN, I2S_TE and I2S_MODULE_EN all have been set to 1. 0b: Disable channeln transmit feature. 1b: Enable channeln transmit feature.	0x1
1 RX_EN	RW	This bit controls channeln receive feature, independent of other channels. Channeln receive starts at time after RX_EN, I2S_RE and I2S_MODULE_EN all have been set to 1.	0x1

Field	Access	Description	Reset Value
		0b: Disable channeln receive feature. 1b: Enable channeln receive feature.	
4:2 TX_AUDIO_RESOLUTION	RW	These bits are defined to select actual transmit data resolutions and keep MSB bit transmit first. Any written values greater than 0x05 will cause a written value 0x5 into these bits. These bits can only be changed when any of TX_EN, I2S_TE and I2S_MODULE_EN is 0. 000b: Reserved 001b: Select 12-bit audio resolution 010b: Select 16-bit audio resolution 011b: Select 20-bit audio resolution 100b: Select 24-bit audio resolution 101b: Select 32-bit audio resolution	0x5
7:5 RX_AUDIO_RESOLUTION	RW	These bits are defined to select actual receive data resolutions. Any written values greater than 0x05 will cause a written value 0x5 into these bits. These bits can only be changed when any of RX_EN, I2S_RE and I2S_MODULE_EN is 0. 000b: Reserved 001b: Select 12-bit audio resolution 010b: Select 16-bit audio resolution 011b: Select 20-bit audio resolution 100b: Select 24-bit audio resolution 101b: Select 32-bit audio resolution	0x5
31:8 RSVD_31_8	RO	Reserved	0x0

39.2.7 I2S Channeln Interrupt Configuration (I2S_CHn_INT_CFG)

39.2.7.1 Offset

Register	Offset
I2S_CHn_INT_CFG	0x14 + 0x14*n (n=0~3)

39.2.7.2 Function

This register is used to configure the channel FIFO interrupt and indicate the channel FIFO status.

39.2.7.3 Fields

Field	Access	Description	Reset Value
0 RXFIFO_AVAILABLE_IE	RW	RXFIFO data available interrupt enable. 0b: Disable RX FIFO data available interrupt. 1b: Enable RX FIFO data available interrupt.	0x0
1 RXFIFO_OVERFLOW_IE	RW	RXFIFO overrun interrupt enable. 0b: Disable RX FIFO overrun interrupt. 1b: Enable RX FIFO overrun interrupt.	0x0
2 TXFIFO_EMPTY_IE	RW	TXFIFO empty interrupt enable. 0b: Disable TX FIFO empty interrupt. 1b: Enable TX FIFO empty interrupt.	0x0

Field	Access	Description	Reset Value
3 TXFIFO_OVERFLOW_IE	RW	TXFIFO overrun interrupt enable. 0b: Disable TX FIFO overrun interrupt. 1b: Enable TX FIFO overrun interrupt.	0x0
4 RXFIFO_DAF	RO	RXFIFO data available flag. 0b: RXFIFO full trigger level does not reach. 1b: RXFIFO full trigger level reaches.	0x0
5 RXFIFO_ORF	W1C	RXFIFO overrun flag will be cleared via w1c option (write 1 to clear this flag). 0b: RXFIFO can receive data. 1b: RXFIFO still receives data when it is full already. New coming data will be lost.	0x0
6 TXFIFO_EMPTYF	RO	TXFIFO empty flag. 0b: TXFIFO empty trigger level does not reach. 1b: TXFIFO empty trigger level reaches and generates this empty flag.	0x1
7 TXFIFO_ORF	W1C	TXFIFO overrun flag will be cleared via w1c option (write 1 to clear this flag). 0b: Write to TXFIFO is valid. 1b: Write to TXFIFO when it is full.	0x0
31:8 RSVD_31_8	RO	Reserved	0x0

39.2.8 I2S Channeln Trigger Level Configuration Register (I2S_CHn_TRIGGER_LEVEL_CFG)

39.2.8.1 Offset

Register	Offset
I2S_CHn_TRIGGER_LEVEL_CFG	0x18 + 0x14*n(n=0~3)

39.2.8.2 Function

This register is used to configure the channel FIFO trigger level and reset the FIFO.

39.2.8.3 Fields

Field	Access	Description	Reset Value
3:0 RXFIFO_FULL_TRIG_LEV	RW	Set the channeln RXFIFO full trigger level. These bits program the level at which RXFIFO is full. When valid entries in RXFIFO are greater than or equal to this level, RXFIFO_DAF interrupt will be generated. For any values written in, if greater than 15, they will be equal to 15.	0x3
7:4 TXFIFO_EMPTY_TRIG_LEV	RW	Set the channeln TXFIFO empty trigger level. These bits program the level at which TXFIFO is empty. When valid entries in TXFIFO are less than or equal to this level, TXFIFO_EMPTYF interrupt will be generated. For any values written in, if greater than 15, they will be equal to 15.	0x3

Field	Access	Description	Reset Value
8 CH_TXFIFO_RESET	WO	Channeln TXFIFO channel reset It's a self-clear bit. Writing 1 to this bit will reset transmit channel FIFO.	0x0
9 CH_RXFIFO_RESET	WO	Channeln RXFIFO channel reset It is a self-clear bit. Writing 1 to this bit will reset receive channel FIFO.	0x0
31:10 RSVD_31_10	RO	Reserved	0x0

39.2.9 I2S Cycle Tx Data Register (I2S_CYCLE_TX_DATA)

39.2.9.1 Offset

Register	Offset
I2S_CYCLE_TX_DATA	0x58

39.2.9.2 Function

This register allows access to all enabled Transmit channels via a single point. The received channels are targeted in a cyclical fashion (starting at the lowest numbered enabled channel) and take two writes (left and right stereo data) before the component points to the next channel.

39.2.9.3 Fields

Field	Access	Description	Reset Value
31:0 CYCLE_TX_DATA	WO	Cycle repeatedly transmit data. It can be used to cycle repeatedly through the enabled Tx channels (from lowest number to highest) to allow writing of stereo data pairs. These data could only be written after I2S_TE and I2S_MODULE_EN are both set high. During write cycles, channels can be enabled or disabled. It could not support disabling a channel in the middle of a stereo pair.	0x0

39.2.10 I2S Cycle Tx Reset Register (I2S_CYCLE_TX_RESET)

39.2.10.1 Offset

Register	Offset
I2S_CYCLE_TX_RESET	0x5C

39.2.10.2 Function

The register is used to reset to the lowest enabled transmitter block channel.

39.2.10.3 Fields

Field	Access	Description	Reset Value
0	WO	I2S cycle transmitter reset	0x0

Field	Access	Description	Reset Value
CYCLE_TX_RESET		Writing 1 to this bit will reset I2S_CYCLE_TX_DATA register to the lowest enabled Transmit channel. It's self-clear bit, and has no effect when only left stereo data is written but right stereo data has not been written yet.	
31:1 RSVD_31_1	RO	Reserved	0x0

39.2.11 I2S Cycle Rx Data Register (I2S_CYCLE_RX_DATA)

39.2.11.1 Offset

Register	Offset
I2S_CYCLE_RX_DATA	0x60

39.2.11.2 Function

This register allows access to all enabled Received channels via a single point. The received channels are targeted in a cyclical fashion (starting at the lowest numbered enabled channel) and take two reads (left and right stereo data) before the component points to the next channel.

39.2.11.3 Fields

Field	Access	Description	Reset Value
31:0 CYCLE_RX_DATA	RO	Cycle repeatedly received data It can be used to cycle repeatedly through the enabled Rx channels (from lowest number to highest) to allow reading of stereo data pairs. It could not support disabling a channel in the middle of a stereo pair.	0x0

39.2.12 I2S Cycle Rx Reset Register (I2S_CYCLE_RX_RESET)

39.2.12.1 Offset

Register	Offset
I2S_CYCLE_RX_RESET	0x64

39.2.12.2 Function

The register is used to reset to the lowest enabled receiver block channel.

39.2.12.3 Fields

Field	Access	Description	Reset Value
0 CYCLE_RX_RESET	WO	I2S cycle receiver block reset Writing 1 to this bit will reset I2S_CYCLE_RX_DATA register to the lowest enabled receiver channel. It's self-clear bit, and has no effect when only left stereo data is read	0x0

Field	Access	Description	Reset Value
		but right stereo data has not been read yet.	
31:1 RSVD_31_1	RO	Reserved	0x0

39.3 Function description

This section provides the functional information of the Inter-IC Sound (I2S) module.

39.3.1 I2S module enable

Before I2S data transmission, the user needs to first set `I2S_CONTROL[I2S_MODULE_EN]=1`. When `I2S_MODULE_EN` is set to 0, it acts as a global reset and following events will occur:

- All I2S TX and RX FIFO are reset and read/write pointers are reset.
- Any data under transmission will be lost and each channel will be disabled internally.
- The master clock generation will be disabled.

39.3.2 I2S master clock generator

I2S clock generator is only valid on master mode. Following descriptions in this chapter are all about master mode. When I2S is set as master mode via setting `I2S_MASTER_CONFIG[I2S_MASTER_EN]=1`, it will generate `ws` and `sclk` lines to connected slaves.

When the `I2S_MASTER_EN` bit is disabled, any receiving or transmitting data is lost due to no clock anymore. However, the data already in the TX/RX FIFOs is preserved. After this bit is re-enabled, I2S transmission will recommence at the start of the next stereo data.

There is a master divider on `I2S_MASTER_CONFIG[MCK_DIV]`. It is only valid on I2S master mode.

The default is divider 2 based on its function clock. The divider clock frequency is equal to $(\text{function clock frequency} / 2 * (\text{MCK\_DIV} + 1))$. And it will be output to `sclk` line under master mode.

For above divider clock, there is also gating option like `I2S_MASTER_CONFIG[I2S_NUM_BCLK_GATE]`, `sclk` will be gated off after different data resolution configuration. It is preferred to set `I2S_NUM_BCLK_GATE` when `I2S_MASTER_EN` is disabled.

Once enabling `I2S_MASTER_CONFIG[I2S_MASTER_EN]=1`, `WS` line will start in its left stereo data cycle (that is `WS=0`). After 2-4 `sclk` synchronous cycles, `WS` switches to its right stereo data cycle (that is `WS=1`). It allows half a frame of `sclk` cycles to fill data into its TX FIFO. The data is transmitted after the first falling edge of `WS` since TX data is valid after first `WS` falling edge. If TX FIFO is not empty during first falling edge of `WS` phase, data is transmitted as expected. If TX FIFO is empty during first falling edge of `WS`, default "0" will be transmitted until next available TX FIFO entry.

39.3.3 I2S as transmitter

39.3.3.1 TX enable control

This I2S has four transmit stereo data channels. They can be configured as either master mode or slave mode. By default, it works as slave mode. Each pair of stereo data is firstly filled into each channel TX FIFO, then shifted out to its `sd` line at falling edge of `sclk`.

When working as a transmitter, you need to set `I2S_CONTROL[I2S_TE]` to 1 and make sure to set `I2S_CONTROL[I2S_RE]=0`. It globally controls all the four TX channels on and off. To enable the transmitter, set `I2S_CONTROL[I2S_TE]=1`. To disable the transmitter, set `I2S_CONTROL[I2S_TE]=0`.

Each transmit channel also has its own channel TX enable control from I2S_CHn_CFG[TX_EN], n is the TX channel number.

When I2S transmitter is disabled or any channel transmit is disabled, following events occur:

- Outgoing data will be lost and TX channel output will be held low.
- Data in TX FIFO will be kept and can be written in.
- Any TX channel configuration like its data resolution, trigger level is preserved and can be reprogrammed.

When a TX channel is disabled, you can reset the channel's TX FIFO via setting I2S_CHn_TRIGGER_LEVEL_CFG[CH_TXFIFO_RESET]=1, where n is the channel number. When the TX channel is enabled, if TX FIFO is not empty, the channel resumes transmission on the next left stereo data cycle when the ws line goes low.

39.3.3.2 TX audio data resolution

Each I2S transmit channel has separate transmit audio data resolution configuration. The default four TX data channel resolution is 32 bits. You can program different audio resolution via setting I2S_CHn_CFG[TX_AUDIO_RESOLUTION] when channel is disabled. For example, channel 1 TX audio data resolution is initially set as 32 bits. You can re-program I2S_CHn_CFG[TX_AUDIO_RESOLUTION], n is channel number to choose different resolutions like 12, 16, 20, 24, or 32 bits.

If TX_AUDIO_RESOLUTION is configured with an invalid value like greater than 0x05, it will choose a default 32-bit resolution.

39.3.3.3 TX channel FIFOs

Each I2S transmit data channel has two FIFO banks of left stereo data and right stereo data. The FIFO depth is 16 words. The FIFO word width is 32bits.

You can clear the TX FIFOs and reset the read/write pointers as following ways. You must disable the transmitter or TX channel before the transmitter and individual TX channel can be reset.

- Via reset
- Via disabling I2S_CONTROL[I2S_MODULE_EN]
- Via setting the transmitter FIFO reset of I2S_CONTROL[I2S_TXFIFO_RESET]=1
- Via setting each individual TX channel FIFO reset of I2S_CHn_TRIGGER_LEVEL_CFG[CH_TXFIFO_RESET]

39.3.3.4 TX channel interrupt

Each transmit data channel of I2S has a separate TX FIFO empty trigger level setting bits of I2S_CHn_TRIGGER_LEVEL_CFG[TXFIFO_EMPTY_TRIG_LEV]. This value can be configured from 0 to 15. Default value is 0x03, which means if valid data entries on TX FIFO are less than or equal to 4, it will generate an empty flag on I2S_CHn_INT_CFG[TXFIFO_EMPTYF] (n is channel number). If I2S_CHn_INT_CFG[TXFIFO_EMPTY_IE]=1, then a TX empty interrupt will occur when the empty flag asserts to 1. You need to disable the TX channel before changing this TXFIFO_EMPTY_TRIG_LEV. This empty flag can be cleared via writing data into TXFIFO to bring its level above the empty trigger threshold level.

I2S transmit channel also has an overrun interrupt. When I2S_CHn_INT_CFG[TXFIFO_OVERUN_IE]=1, this flag is generated when writing data to a full TXFIFO. (Any data being written will be lost while FIFO data is still preserved). It will be cleared via w1c option.

39.3.3.5 Example of filling TX FIFO

I2S always transmits a pair of stereo data including left stereo and right stereo data. They can be written via following two steps: (You need to write TX data like this order --- writing left data first, then right data, otherwise the related empty and overrun flag/interrupt may be invalid).

1. Write data into I2S_CHn_LEFT_DATA, n is channel number
2. Write data into I2S_CHn_RIGHT_DATA, n is channel number

I2S also provides a cyclic transmit way of each TX channel. It only requires write data into I2S_CYCLE_TX_DATA, rather than I2S_CHn_LEFT_DATA and I2S_CHn_RIGHT_DATA. Transmitted data is written cyclically through all enabled TX channels. It always starts from the lowest-number index enabled TX channel. After a pair of stereo data is transmitted, it will point to the next enabled channel. The following gives a detailed example of this cyclic transfer. Only TX channel 0, 1, 3 are enabled but channel 2 is disabled.

The transmit data order will be:

1. Channel 0 left data
2. Channel 0 right data
3. Channel 1 left data
4. Channel 1 right data
5. Channel 3 left data
6. Channel 3 right data
7. Repeat above items 1-6

I2S also has a reset bit on I2S_CYCLE_TX_RESET[CYCLE_TX_RESET]. Once it is asserted to 1, it will reset cyclic transfer to lowest enabled TX channel. This bit can be written at any stage of cyclic transmit cycle. It has no effect during the middle of a pair of stereo data. Following example shows the CYCLE_TX_RESET behavior.

The transmit data order will be:

1. Channel 0 left data
2. Channel 0 right data
3. Channel 1 left data
4. Channel 1 right data
5. Setting I2S_CYCLE_TX_RESET[CYCLE_TX_RESET]=1 after completing a pair of stereo data, reset is valid and next channel will be reset to channel 0
6. Channel 0 left data
7. Channel 0 right data
8. Channel 1 left data
9. Setting I2S_CYCLE_TX_RESET[CYCLE_TX_RESET]=1 at the middle of a pair of stereo data, which will have no effect.
10. Channel 1 right data
11. Channel 3 left data
12. Channel 3 right data

39.3.4 I2S as receiver

39.3.4.1 Receiver enable control

This I2S has four receive stereo data channels. They can be configured as either master mode or slave mode. By default, it works as slave mode. Each pair of channel stereo data is

received from sd input line like sd0, sd1, sd2, and sd3. These data words are stored on each channel RX FIFO until they are read via APB bus.

The I2S receiver enable bit of I2S_CONTROL[I2S_RE] controls all four received channels. To enable receiver block, set I2S_CONTROL[I2S_RE]=1 and make sure to set I2S_CONTROL[I2S_TE]=0; to disable receiver block, set I2S_CONTROL[I2S_RE]=0.

Each receive channel also has its own channel RX enable control from I2S_CHn_CFG[RX_EN], n is the RX channel number. When I2S_CHn_CFG[RX_EN]=0, you can also set I2S_CHn_TRIGGER_LEVEL_CFG[CH_RXFIFO_RESET]=1 to reset this channel RX FIFO.

When receiver block is disabled or RX channel is disabled, following items will occur:

- Stop receiving data.
- Data received in RX FIFOs is kept and FIFOs can be read.
- Any RX channel configuration (like RX audio data resolution, channel trigger levels) is preserved.

39.3.4.2 RX audio data resolution

Each I2S receive channel has separate receive audio data resolution configuration. The default resolution of four RX data channels is 32 bits. You can program different audio resolution via setting I2S_CHn_CFG[RX_AUDIO_RESOLUTION] when channel is disabled. For example, channel 1 RX audio data resolution is initially set as 32 bits. You can re-program I2S_CHn_CFG[RX_AUDIO_RESOLUTION], n is channel number to choose different resolutions like 12, 16, 20, 24, or 32 bits.

If RX_AUDIO_RESOLUTION is configured with an invalid value like greater than 0x05, it will choose a default 32bit resolution.

39.3.4.3 RX channel FIFOs

Each I2S receive data channel has two FIFO banks of left stereo data and right stereo data. The FIFO depth is 16 words. The FIFO word width is 32 bits.

You can clear the RX FIFOs and reset the read/write pointers as following ways. You must disable the receiver or RX channel before the receiver and individual RX channel can be reset.

- Via reset
- Via disabling I2S_CONTROL[I2S_MODULE_EN]
- Via setting the receiver FIFO reset of I2S_CONTROL[I2S_RXFIFO_RESET]=1
- Via setting each individual RX channel FIFO reset of I2S_CHn_TRIGGER_LEVEL_CFG[CH_RXFIFO_RESET]

39.3.4.4 RX channel interrupt

Each receive data channel of I2S has a separate RX FIFO available trigger level setting bits of I2S_CHn_TRIGGER_LEVEL_CFG[RXFIFO_FULL_TRIG_LEV]. This value can be configured from 0 to 15. Default value is 0x03, which means if valid data entries on RX FIFO are greater than or equal to 4, it will generate a data available full flag on I2S_CHn_INT_CFG[RXFIFO_DAF](n is channel number). If I2S_CHn_INT_CFG[RXFIFO_AVAILABLE_IE]=1, then an RX data available full interrupt will occur when the available flag asserts to 1. You need to disable the RX channel before changing this RXFIFO_FULL_TRIG_LEV. This available flag can be cleared via reading data from RXFIFO to bring its level lower than the available full trigger threshold level.

I2S receiver channel also has an overrun interrupt. When I2S_CHn_INT_CFG[RXFIFO_OVERUN_IE]=1, this flag is generated when writing data to a full RXFIFO. (Any data being written will be lost while FIFO data is still preserved). It will be cleared via w1c option.

39.3.4.5 Example of reading RXFIFO

The I2S stereo data is received via its RX channel, and stored into its RXFIFO. These RX FIFO can be read via reading I2S_CHn_LEFT_DATA and I2S_CHn_RIGHT_DATA, n is channel number. It requires reading from left channel first, then right channel.

I2S receiver also supports a cyclic reading via reading I2S_CYCLE_RX_DATA rather than I2S_CHn_LEFT_DATA and I2S_CHn_RIGHT_DATA. The reading access is the same as TX cyclic behavior.

Once the reset bit of I2S_CYCLE_RX_RESET[CYCLE_RX_RESET] is asserted to 1, it will reset cyclic receive to lowest enabled RX channel. It is the same with CYCLE_TX_RESET.

39.3.5 I2S configuration example

39.3.5.1 I2S as transmitter on slave mode

When I2S works as a transmitter, it can be configured either as a slave or a master. I2S has two ways of filling TX FIFO, one is normal mode and the other one is cyclic mode.

39.3.5.1.1 Normal mode

The following shows configuring I2S as transmitter on slave mode via its normal configuration steps.

1. Setting I2S_CONTROL[I2S_MODULE_EN]=1 to enable I2S module.
2. Filling TX data into TX FIFO via writing I2S_CHn_LEFT_DATA and I2S_CHn_RIGHT_DATA respectively. Note: keep this write order as write left channel data first, then write right channel data.
3. Setting I2S_CONTROL[I2S_TE]=1 to enable I2S transmitter block.
4. Setting I2S_CHn_CFG[TX_EN]=1 to enable I2S TX channel enable. Default channel enable is 1. I2S will transmit TX data after first falling edge of ws line.

39.3.5.1.2 Cyclic mode

The following shows configuring I2S as transmitter on slave mode via its cyclic configuration steps.

1. Setting I2S_CONTROL[I2S_MODULE_EN]=1 to enable I2S module.
2. Setting I2S_CONTROL[I2S_TE]=1 to enable I2S transmitter block.
3. Filling TX data into TX FIFO via writing I2S_CYCLE_TX_DATA.
4. Setting I2S_CHn_CFG[TX_EN]=1 to enable I2S TX channel enable. Default TX channel enable is 1. I2S will transmit TX data after first falling edge of ws line.

39.3.5.2 I2S as transmitter on master mode

1. Repeat above steps 1-4 in normal mode or cyclic mode first.
2. Enable master clock generation via setting I2S_MASTER_CONFIG[I2S_MASTER_EN].

39.3.5.3 I2S as receiver on slave mode

When I2S works as a receiver, it can be configured either as a slave or a master. The following shows configuring I2S as receiver on slave mode via its normal configuration steps:

1. Setting I2S_CONTROL[I2S_MODULE_EN]=1 to enable I2S module.
2. Setting I2S_CONTROL[I2S_RE]=1 to enable I2S receiver block.
3. Setting I2S_CHn_CFG[RX_EN]=1 to enable I2S RX channel enable. Default RX channel enable is 1.
4. Waiting I2S_CHn_INT_CFG[RXFIFO_DAF]=1, which indicates RX FIFO has reached the full trigger level.
5. Reading data from I2S_CHn_LEFT_DATA and I2S_CHn_RIGHT_DATA to check RX FIFO contents.

RX FIFO on slave mode also supports cyclic reading way to check received data. The only difference is step 5, reading data directly from I2S_CYCLE_RX_DATA instead of separate left and right channel data.

39.3.5.4 I2S as receiver on master mode

The following shows configuring I2S as receiver on master mode via its normal configuration steps:

1. Setting I2S_CONTROL[I2S_MODULE_EN]=1 to enable I2S module.
2. Setting I2S_CONTROL[I2S_RE]=1 to enable I2S receiver block.
3. Setting I2S_CHn_CFG[RX_EN]=1 to enable I2S RX channel enable. Default RX channel enable is 1.
4. Enable master clock generation via setting I2S_MASTER_CONFIG[I2S_MASTER_EN].
5. Waiting I2S_CHn_INT_CFG[RXFIFO_DAF]=1, which indicates RX FIFO has reached the full trigger level.
6. Reading data from I2S_CHn_LEFT_DATA and I2S_CHn_RIGHT_DATA to check RX FIFO contents.

RX FIFO on master mode also supports cyclic reading way to check received data. The only difference is step 6, reading data directly from I2S_CYCLE_RX_DATA instead of separate left and right channel data.

Chapter 40

Controller Area Network (CAN)

40.1 Introduction

This chapter defines Controller Area Network (CAN).

40.1.1 Features

The CAN supports the following key features:

- Full implementation of the CAN with Flexible Data Rate (CAN FD) protocol specification and CAN protocol specification, Version 2.0 B
 - Standard data frames
 - Extended data frames
 - Zero to sixty-four bytes data length
 - Programmable bit rate
 - Content-related addressing
- Compliant with the ISO 11898-1 standard
- Flexible mailboxes configurable to store 0 to 8, 16, 32 or 64 bytes data length
- Each mailbox configurable as receive or transmit, all supporting standard and extended messages
- Individual Rx Mask registers per mailbox
- Full-featured legacy Rx FIFO with storage capacity for up to six frames and automatic internal pointer handling with DMA support
- Transmission abort capability
- Flexible message buffers (MBs), totaling 64/128 message buffers of 8 bytes data length each, configurable as Rx or Tx
- RAM not used by reception or transmission structures can be used as general-purpose RAM space
- Listen-only mode capability
- Programmable Loop-Back mode supporting self-test operation
- Programmable transmission priority scheme: lowest ID, lowest buffer number, or highest priority
- Global network time, synchronized by a specific message
- Maskable interrupts
- Independence from the transmission medium (an external transceiver is assumed)
- Short latency time due to an arbitration scheme for high-priority messages
- Transceiver Delay Compensation feature when transmitting CAN FD messages at faster data rates
- Remote request frames may be handled automatically or by software
- CAN bit time settings and configuration bits can only be written in Freeze mode
- Tx mailbox status (Lowest priority buffer or empty buffer)
- Identifier Acceptance Filter Hit Indicator (IDHIT) register for received frames
- SYNCH bit available in CAN_ESR1 register to inform that the module is synchronous with CAN bus
- CRC status for the transmitted message
- Legacy Rx FIFO Global Mask register
- Selectable priority between mailboxes and Rx FIFO during the matching process
- Powerful legacy Rx FIFO ID filtering, capable of matching incoming IDs against either 128 extended, 256 standard, or 512 partial (8 bit) IDs, with up to 32 individual masking capability
- 100% backward compatibility with previous CAN version
- Supports detection and correction of errors in memory read accesses. Each byte of CAN memory is associated with 5 parity bits, and the error correction mechanism ensures that

in this 13-bit word, errors in one bit can be corrected (correctable errors) and errors in 2 bits can be detected but not corrected (non-correctable errors)

40.1.2 Modes of operation

The CAN module has the following functional modes:

- Normal mode (User or Supervisor):
 - In Normal mode, the module operates receiving and/or transmitting message frames, errors are handled normally, and all CAN Protocol functions are enabled. User Mode differs from Supervisor Mode in the access to some restricted control registers.
- Freeze mode:
 - Freeze mode is enabled when the FRZ bit in CAN_MCR is asserted. If enabled, Freeze mode is entered when CAN_MCR[HALT] is set or when Debug mode is requested at the chip level, and the CAN asserts CAN_MCR[FRZACK] (CAN_MCR[FRZACK] can be asserted successfully only when CAN_MCR[MDIS] is set to 0). In this mode, no transmission or reception of frames is done, and synchronization to the CAN bus is lost.
- Loop-Back mode:
 - The module enters this mode when the LPB field in the Control 1 Register (CAN_CTRL1) is asserted. In this mode, CAN performs an internal loop back that can be used for self-test operation. The bit stream output of the transmitter is internally fed back to the receiver input. The Rx CAN input pin is ignored, and the Tx CAN output goes to the recessive state (logic '1'). CAN behaves as it normally does when transmitting and treats its own transmitted message as a message which is received from a remote node. In this mode, CAN ignores the bit which is sent during the ACK slot in the CAN frame acknowledge field to ensure proper reception of its own message. Both transmit and receive interrupts are generated.
- Listen-Only mode:
 - The module enters this mode when the LOM field in the Control 1 Register (CAN_CTRL1) is asserted. In this mode, transmission is disabled, all error counters are frozen, and the module operates in a CAN Error Passive mode. Only messages acknowledged by another CAN station will be received. If CAN detects a message which has not been acknowledged, it will flag a BIT0 error (without changing the REC), as if it was trying to acknowledge the message.
- CAN FD Active mode:
 - In this mode, CAN can transmit and receive all messages formatted according to the CAN FD Protocol and CAN Protocol 2.0 in an interleaved fashion. In addition, the CPU can set the CAN into CAN FD Active mode by configuring the CAN_MCR[FDEN] bit field in Freeze Mode.

For low-power operation, the CAN module has:

- Module Disable mode:
 - This low-power mode is entered when the MDIS bit in the CAN_MCR register which is asserted by the CPU and the CAN_MCR[LPMACK] is asserted by the CAN. When disabled, the module requests to disable the clocks to the CAN Protocol Engine and Controller Host Interface submodules. Exit from this mode is done by negating the MDIS bit in the CAN_MCR register.
- Doze mode:
 - This low power mode is entered when the DOZE bit is asserted, and Doze mode is requested at the chip level, and the CAN asserts the LPMACK bit in the CAN_MCR register. When in Doze mode, the module requests to disable the clocks to the CAN Protocol Engine and the CAN Controller-Host Interface sub-modules. Exit from this

mode occurs when the DOZE bit in CAN_MCR is negated, or when the chip is removed from Doze mode, or when activity is detected on the CAN bus and the Self Wake Up mechanism is enabled.

- Stop mode:
 - This low power mode is entered when Stop mode is requested at the chip level, and the CAN asserts the LPMACK bit in the CAN_MCR register. When in Stop mode, the module puts itself in an inactive state and then informs the CPU that the clocks can be shut down globally. Exit from this mode occurs when the Stop mode request is removed, or activity is detected on the CAN bus, and the Self Wake Up mechanism is enabled.
- Pretended Network mode:
 - This mode can be selected to operate together with Doze mode or Stop mode. Prior to entering one of these low power modes, the PNET_EN bit in CAN_MCR Register must be asserted. Once in low power mode, CHI sub-block clocks are shut down and CAN_PE sub-block is kept clocked, so that the Rx receive process is still active to filter incoming messages as defined by the configuration registers. Upon detection of a wakeup event, a wake-up interrupt is issued to the system. When the PNET_EN bit in CAN_MCR is asserted, the CPU must disable the Self Wake Up feature by negating the SLFWAK bit in the CAN_MCR register.

40.1.3 Block diagram

The following figure is a block diagram of the CAN module.

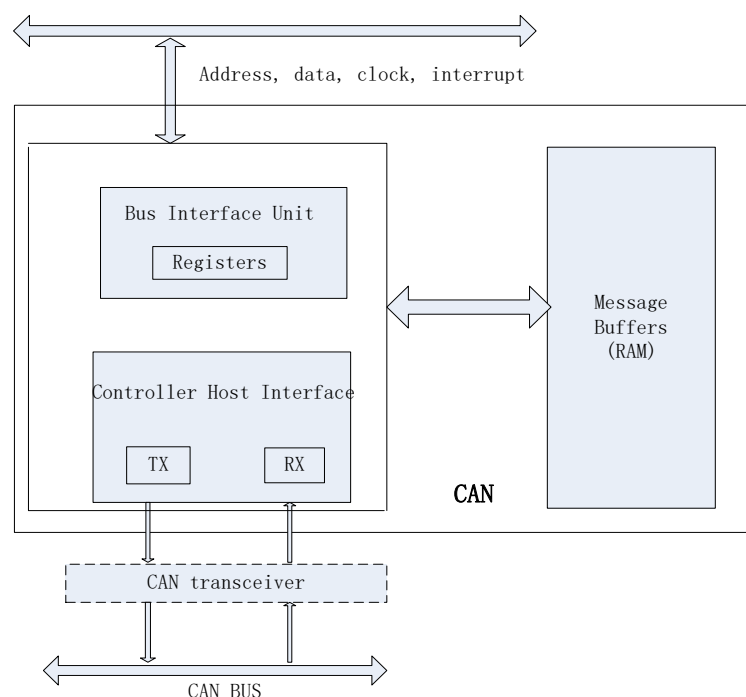


Figure 40-1 CAN block diagram

40.1.4 Instantiation information

The following table lists the CAN instances and features for each chip in the product series.

Table 40-1 CAN instance and features

Chips	Instances	Number of Message Buffers (MB)	CAN FD feature

Chips	Instances	Number of Message Buffers (MB)	CAN FD feature
K144MC	CAN0	64	Yes
	CAN1	64	Yes
	CAN2	64	Yes
	CAN3	64	Yes
K146MC	CAN0	64	Yes
	CAN1	64	Yes
	CAN2	64	Yes
	CAN3	64	Yes
	CAN4	64	Yes
	CAN5	64	Yes
K148MC	CAN0	64	Yes
	CAN1	64	Yes
	CAN2	64	Yes
	CAN3	64	Yes
	CAN4	64	Yes
	CAN5	64	Yes
	CAN6	128	Yes
	CAN7	128	Yes

40.2 Signal description

The CAN module has two I/O signals connected to the external chip pins. These signals are summarized in the following table and described in more details as below.

Table 40-2 CAN signal descriptions

Signal	Description	I/O
CAN Rx	CAN receive pin	Input
CAN Tx	CAN transmit pin	Output

CAN Rx: This pin is the receive pin from the CAN bus transceiver. The dominant state is represented by logic level 0. The recessive state is represented by logic level 1.

CAN Tx: This pin is the transmit pin to the CAN bus transceiver. The dominant state is represented by logic level 0. The recessive state is represented by logic level 1.

40.3 Memory map and register definition

This section provides the memory map and register definitions of the CAN module.

40.3.1 Memory map

CAN0 base address: 0x40024000

CAN1 base address: 0x40025000

CAN2 base address: 0x40038000

CAN3 base address: 0x40039000

CAN4 base address: 0x40044000

CAN5 base address: 0x40045000

CAN6 base address: 0x40056000

CAN7 base address: 0x40057000

Table 40-3 CAN memory map

Offset	Register	Width (In bits)	Reset Value
0x00	CAN Module Configure Register (CAN_MCR)	32	0xd890000f
0x04	CONTROL 1 Register (CAN_CTRL1)	32	0x00000000
0x08	Free Running Timer Register (CAN_TIMER)	32	0x00000000
0x10	Rx Mailbox Global Mask Register (CAN_RXMGMSK)	32	-
0x14	Rx 14 Mask Register (CAN_RX14MASK)	32	-
0x18	Rx 15 Mask Register (CAN_RX15MASK)	32	-
0x1c	Error Counter Register (CAN_ECR)	32	0x00000000
0x20	CAN Error Status Register (CAN_ESR1)	32	0x00000000
0x24	Interrupt Mask 2 Register (CAN_IMASK2)	32	0x00000000
0x28	Interrupt Mask 1 Register (CAN_IMASK1)	32	0x00000000
0x2c	Interrupt Flag 2 Register (CAN_IFLAG2)	32	0x00000000
0x30	Interrupt Flag 1 Register (CAN_IFLAG1)	32	0x00000000
0x34	Control 2 Register (CAN_CTRL2)	32	0x00800000
0x38	Error and Status 2 Register (CAN_ESR2)	32	0x00000000
0x44	CRC Register (CAN_CRCCR)	32	0x00000000
0x48	Legacy FIFO Global Mask Register (CAN_RXFGMASK)	32	-
0x4c	RX FIFO ID Indicator (CAN_RXFIR)	32	-
0x50	CAN Bit Timing Register (CAN_CBT)	32	0x00000000
0x68	Interrupt Mask 4 register (CAN_IMASK4)	32	0x00000000
0x6c	Interrupt Mask 3 Register (CAN_IMASK3)	32	0x00000000
0x70	Interrupt Flag 4 Register (CAN_IFLAG4)	32	0x00000000
0x74	Interrupt Flag 3 Register (CAN_IFLAG3)	32	0x00000000
0x80+16*n CAN0-5: n=0-63 CAN6-7: n=0-127	Message Buffer Control and Status Register (CAN_MBn_CS)	32	-
0x84 +16*n CAN0-5: n=0-63 CAN6-7: n=0-127	Message Buffer ID Register (CAN_MBn_ID)	32	-

Offset	Register	Width (In bits)	Reset Value
0x88+16*n CAN0-5: n=0-63 CAN6-7: n=0-127	Message Buffer Data03 Register (CAN_MBn_DATA03)	32	-
0x8C+16*n CAN0-5: n=0-63 CAN6-7: n=0-127	Message Buffer Data47 Register (CAN_MBn_DATA47)	32	-
0x880+4*n CAN0-5: n=0-63 CAN6-7: n=0-127	RX Individual Mask Register (CAN_RXIMRn)	32	-
0xAE0	Memory Error Control Register (CAN_MECR)	32	0x800c0080
0xAE4	Error Injection Address Register (CAN_ERRIAR)	32	0x00000000
0xAE8	Error Injection Data Pattern Register (CAN_ERRIDPR)	32	0x00000000
0xAEC	Error Injection Parity Pattern Register (CAN_ERRIPPR)	32	0x00000000
0xAF0	Error Report Address Register (CAN_RERRAR)	32	0x00000000
0xAF4	Error Report Data Register (CAN_RERRDR)	32	0x00000000
0xAF8	Error Report Syndrome Register (CAN_RERRSYNR)	32	0x00000000
0xAFC	Error Status Register (CAN_ERRSR)	32	0x00000000
0xB00	Pretend Network Control 1 Register (CAN_CTRL1_PN)	32	0x00000100
0xB04	Pretend Network Control 2 Register (CAN_CTRL2_PN)	32	0x00000000
0xB08	Pretend Network Wakeup Match Register (CAN_WU_MTC)	32	0x00000000
0xB0C	Pretend Network ID Filter 1 Register (CAN_FLT_ID1)	32	0x00000000
0xB10	Pretend Network DLC Filter Register (CAN_FLT_DLC)	32	0x00000008
0xB14	Pretend Network Payload Low Filter 1 register (CAN_PL1_LO)	32	0x00000000
0xB18	Pretend Network Payload High Filter 1 Register (CAN_PL1_HI)	32	0x00000000
0xB1C	Pretended Networking ID Filter 2 Register / ID Mask Register (CAN_FLT_ID2_IDMASK)	32	0x00000000
0xB20	Pretended Networking Payload Low Filter 2 Register / Payload Low Mask Register (CAN_PL2_PLMASK_LO)	32	0x00000000
0xB24	Pretended Networking Payload High Filter 2 low order bits / Payload High Mask Register (CAN_PL2_PLMASK_HI)	32	0x00000000
0xB40+n*16 (n=0 - 3)	Wake Up Message Buffer Register for C/S	32	0x00000000

Offset	Register	Width (In bits)	Reset Value
	(CAN_WMBn_CS)		
0xB44+n*16 (n=0-3)	Wake Up Message Buffer Register for ID (CAN_WMBn_ID)	32	0x00000000
0xB48+n*16 (n=0-3)	Wake Up Message Buffer Register for Data 0-3 (CAN_WMBn_D03)	32	0x00000000
0xB4C+n*16 (n=0-3)	Wake Up Message Buffer Register for Data 4-7 (CAN_WMBn_D47)	32	0x00000000
0xC00	CAN FD Control Register (CAN_FDCTRL)	32	0x80000100
0xC04	CAN FD Bit Timing Control Register (CAN_FDCBT)	32	0x00000000
0xC08	CAN FD CRC Register (CAN_FDCRC)	32	0x00000000

40.3.2 CAN Module Configure Register (CAN_MCR)

40.3.2.1 Offset

Register	Offset
CAN_MCR	0x00

40.3.2.2 Function

This register defines global system configurations, which can be accessed (read or write) only in supervisor mode.

40.3.2.3 Fields

Field	Access	Description	Reset Value
6:0 MAXMB	RW	Number Of The Last Message Buffer This 7-bit field defines the number of the last Message Buffers that will take part in the matching and arbitration processes. The reset value (0x0F) is equivalent to a 16 MB configuration. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Number of the last MB = MAXMB	0xF
7 RSVD_7	RO	Reserved	0x0
9:8 IDAM	RW	ID Acceptance Mode This 2-bit field identifies the format of the Legacy Rx FIFO ID Filter Table elements. Note that all elements of the table are configured at the same time by this field (they are all in the same format). This field can be written only in Freeze mode because it is blocked by hardware in other modes. 00b: Format A. One full ID (standard and extended ID) per ID filter table elements. 01b: Format B. Two full standard IDs or four partial 14 bits (standard or extended) IDs per ID filter table.	0x0

Field	Access	Description	Reset Value
		10b: Format C. Four partial 8 bits standard IDs per ID filter table. 11b: Reject all frames.	
10 RSVD_10	RO	Reserved	0x0
11 FDEN	RW	CAN FD operation enable This bit enables the CAN with Flexible Data rate (CAN FD) operation. This bit can be written in Freeze mode only. 0b: CAN FD is disabled. CAN can receive and transmit messages in CAN 2.0 format. 1b: CAN FD is enabled. CAN can receive and transmit messages in both CAN FD and CAN 2.0 formats. NOTE: The Rx FIFO Enable (RFEN) bit cannot be set if FDEN is asserted.	0x0
12 AEN	RW	Abort Enable This bit enables the Tx abort mechanism. This mechanism guarantees a safe procedure for aborting a pending transmission so that no frame is sent in the CAN bus without notification. It is only written in the FREEZE mode. 0b: Abort disable. 1b: Abort enable. NOTE When Abort Enable, the user must use it only for updating mailboxes configured for transmission. Writing the Abort code into Rx mailboxes can cause unpredictable results.	0x0
13 LPRIOEN	RW	Local Priority Enable It controls whether the local priority feature is enabled or not. It is used to expand the ID used during the arbitration process. With this expanded ID concept, the arbitration process is done based on the full 32-bit word, but the actual transmitted ID still has 11-bit for standard frames and 29-bit for extended frames. It is only written in the FREEZE mode. 0b: Disable local priority feature. 1b: Enable local priority feature.	0x0
14 PNET_EN	RW	Pretended Networking Enable When in stop mode, the PE sub-block is kept operational, making it able to process Rx message filtering as defined by the Pretended Networking configuration registers. It is only written in the FREEZE mode. 0b: Pretended network is disabled. 1b: Pretended network is enabled.	0x0
15 DMAE	RW	DMA Enable DMA feature for CAN RX FIFO is enabled. It is only written in the FREEZE mode. 0b: DMA is disabled. 1b: DMA is enabled. NOTE The DMA feature can only be used in Rx FIFO. Consequently, CAN_MCR[RFEN] must be asserted. When DMAE and RFEN are set, CAN_IFLAG1[BUF5I]	0x0

Field	Access	Description	Reset Value
		generates the DMA request, and no RX FIFO interrupt is generated.	
16 IRMQ	RW	Individual Rx Masking And Queue Enable The Rx matching process will be based either on individual masking and queue or on masking scheme with CAN_RXMGMSK, CAN_RX14MASK, CAN_RX15MASK, and CAN_RXFGMASK. It is only written in the FREEZE mode. 0b: Individual RX mask and queue disable. 1b: Individual RX mask and queue enable.	0x0
17 SRXDIS	RW	Self Reception Disable Frames transmitted by the module will not be stored in any MB, regardless of whether the MB is programmed with an ID that matches the transmitted frame. No interrupt flag or interrupt signal will be generated due to the frame reception. It is only written in the FREEZE mode. 0b: Self reception enable. 1b: Self reception disable.	0x0
18 DOZE	RW	This bit defines whether CAN can enter a low-power mode when Doze mode is requested at the chip level. This bit is automatically reset when CAN wakes up from the Doze mode upon detecting activity on the CAN bus. The self wake-up is enabled. 0b: CAN is not enabled to enter the low-power mode when Doze mode is requested. 1b: CAN is enabled to enter the low-power mode when Doze mode is requested.	0x0
19 WAKSRC	RW	Wake Up Source This bit defines whether the integrated low-pass filter is applied to protect the Rx CAN input from spurious wakeup. This bit is written only in Freeze mode because it is blocked by hardware in other modes. 0b: CAN uses the unfiltered Rx input to detect recessive to dominant edges on the CAN bus. 1b: CAN uses the filtered Rx input to detect recessive to dominant edges on the CAN bus.	0x0
20 LPMACK	RO	Low-Power Mode Acknowledge This read-only bit indicates that CAN is in a low-power mode (Disable mode, Doze mode and Stop mode). A low-power mode cannot be entered until all current transmission or reception processes have finished, so the CPU can poll the LPMACK bit to know when CAN has entered into low power mode. This bit is not affected by soft reset. 0b: CAN is not in the low power mode. 1b: CAN is under the low power mode.	0x1
21 WRNEN	RW	Warning Interrupt Enable When asserted, this bit enables the generation of the TWRNINT and RWRNINT flags in the Error and Status Register 1 (CAN_ESR1). If WRNEN is negated, the TWRNINT and RWRNINT flags will always be zero, independent of the values of the error counters, and no warning interrupt will ever be generated. This bit can be written in Freeze mode only because it is blocked by hardware in other modes.	0x0

Field	Access	Description	Reset Value
		0b: TWRNINT and RWRNINT bits are zero, independent of the values in the error counters. 1b: TWRNINT and RWRNINT bits are set when the respective error counter transitions from less than 96 to greater than or equal to 96.	
22 SLFWAK	RW	Self Wake Up This bit enables the Self Wake Up feature when CAN is in the low-power mode other than the Disable mode. The CAN module monitors the bus for a wake-up event, a recessive-to-dominant transition when this feature is enabled. - If a wake-up event is detected in the Doze mode, CAN requests to resume its clocks. If it is enabled to do so, it generates a Wake-Up interrupt to the CPU. - If a wake-up event is detected during Stop mode, then CAN generates, if enabled to do so, a Wake-Up interrupt to the CPU so that it can exit Stop mode globally and request to resume the clocks. - However, when CAN is in the low-power mode other than the Disable mode, this bit cannot be written because it is blocked by hardware. When the Pretended Networking mode is set, this feature must be disabled. 0b: CAN Self Wake Up feature is disabled. 1b: CAN Self Wake Up feature is enabled.	0x0
23 SUPV	RW	Supervisor Mode This bit configures the CAN to be either in Supervisor or User mode. All the registers except CAM_MCR are affected by this field. CAN_MCR can only be accessed in supervisor mode. The reset value of this bit is 1, so the affected registers start with Supervisor access allowance only. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0b: CAN is in the User mode. Affected registers allow both Supervisor and Unrestricted Accesses. 1b: CAN is in the Supervisor mode. Affected registers allow only Supervisor access. Unrestricted access behaves as though the access was done to an unimplemented register location.	0x1
24 FRZACK	RO	Freeze Mode Acknowledge This read-only bit indicates that CAN is in the Freeze mode, and its prescaler is stopped. The Freeze mode request cannot be granted until the current transmission or reception processes have finished. Therefore, the software can poll the FRZACK bit to know when CAN has entered the Freeze mode. If the Freeze Mode request is negated, this bit is negated after the CAN prescaler is running again. For example, if Freeze mode is requested while CAN is in the low power mode, the FRZACK bit will be set only when the low-power mode is exited. See the "Freeze Mode" section for more details. This bit is not affected by a soft reset.	0x0

Field	Access	Description	Reset Value
		NOTE FRZACK will be asserted within 178 CAN bits from the freeze mode request by the CPU and negated within 2 CAN bits after the freeze mode request removal. 0b: CAN is not in the Freeze mode, and the prescaler is running. 1b: CAN is in the Freeze mode, and the prescaler has stopped.	
25 SOFTRST	RW	Soft Reset CAN resets its internal state machines and some of the memory mapped registers when this bit is asserted. The SOFTRST bit can be asserted directly by the CPU when it writes to the CAN_MCR register, but it is also asserted when a global soft reset is requested at the chip level. Because the soft reset is synchronous and must follow a request/acknowledge procedure across clock domains, it may take some time to fully propagate its effect. The SOFTRST bit remains asserted while the reset is pending and is automatically negated when the reset completes. Therefore, the software can poll this bit to know when the soft reset has been completed. A soft reset cannot be applied while clocks are shut down in a low power mode. The module should be first removed from the low power mode, and then a soft reset can be applied. This bit is not affected by the soft reset. 0b: No reset is requested. 1b: Resets the registers affected by a soft reset.	0x0
26 WAKMSK	RW	Wake Up Interrupt Mask This bit enables the Wake Up Interrupt generation under the Self Wake Up mechanism. 0b: The wake up interrupt is disabled. 1b: The wake up interrupt is enabled.	0x0
27 NOTRDY	RO	CAN Not Ready This read-only bit indicates that CAN is either in Disable mode, Doze mode, Stop mode or Freeze mode. It is negated once CAN has exited these modes. This bit is not affected by soft reset. 0b: CAN module is either in Normal mode, Listen-Only mode or Loop-Back mode. 1b: CAN module is either in Disable mode, Doze mode, Stop mode or Freeze mode.	0x1
28 HALT	RW	Halt CAN Assertion of this bit puts the CAN module into Freeze mode. The CPU should clear it after initializing the Message Buffers and the Control Registers CAN_CTRL1 and CAN_CTRL2. CAN performs no reception or transmission before this bit is cleared. It cannot enter the Freeze mode while CAN is in the low power mode. The HALT bit is set by hardware when a non-correctable error is detected and CAN_MECR[NCEFAFRZ] is asserted. 0b: No Freeze mode is requested. 1b: Enters Freeze mode if the FRZ bit is asserted.	0x1
29	RW	Legacy Rx FIFO Enable	0x0

Field	Access	Description	Reset Value
RFEN		This bit controls whether the Legacy Rx FIFO feature is enabled or not. When RFEN is set, MBs 0 to 5 cannot be used for the normal reception and transmission. Because the corresponding memory region (0x80-0xDC) is used by the FIFO engine as well as additional MBs (up to 32, depending on CAN_CTRL2[RFFN] setting), which are used as Legacy Rx FIFO ID Filter Table elements. RFEN also impacts the definition of the minimum number of peripheral clocks per CAN bit as described in the table "Minimum number of peripheral clocks per can bit for NON-FD format". This bit can be written in the Freeze mode only because it is blocked by hardware in other methods. NOTE This bit cannot be set when CAN FD operation is enabled (see FDEN bit). 0b: Legacy Rx FIFO is not enabled. 1b: Legacy Rx FIFO is enabled.	
30 FRZ	RW	Freeze Enable The FRZ bit specifies the CAN behavior when CAN_MCR[HALT] is set or when Debug mode is requested at the chip level. When FRZ is asserted, CAN is enabled to enter the Freeze mode. Negation of this bit field causes CAN to exit from the Freeze mode. This bit is set by hardware when a non-correctable error is detected CAN_MECCR[NCEFAFRZ] is asserted. 0b: Not enabled to enter the Freeze mode. 1b: Enabled to enter the Freeze mode.	0x1
31 MDIS	RW	Module Disable This bit controls whether CAN is enabled or not. When disabled, CAN disables the clocks to the CAN Protocol Engine and Controller Host Interface sub-modules. Therefore, this bit is not affected by a soft reset. 0b: Enable the CAN module. 1b: Disable the CAN module.	0x1

40.3.3 Control 1 Register (CAN_CTRL1)

40.3.3.1 Offset

Register	Offset
CAN_CTRL1	0x04

40.3.3.2 Function

The whole register is written only in the FREEZE mode, when CAN_CBT[BTF] is set. The register is defined for specific CAN control features related to the CAN bus.

40.3.3.3 Fields

Field	Access	Description	Reset Value
2:0	RW	Propagation Segment	0x0

Field	Access	Description	Reset Value
PROPSEG		This 3-bit field defines the length of the Propagation Segment in the bit time. The valid programmable values are 0–7. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Propagation Segment Time = (PROPSEG + 1) × Time-Quanta. Time-Quantum = one Sclock period.	
3 LOM	RW	Listen-Only Mode This bit configures CAN to operate in Listen-Only mode. In this mode, transmission is disabled, all error counters described in the CAN_ECR register are frozen and the module operates in a CAN Error Passive mode. Only messages acknowledged by another CAN station will be received. If CAN detects a message that has not been acknowledged, it will flag a BIT0 error without changing the receive error counter (CAN_ECR[RXERRCNT]), as if it was trying to acknowledge the message. Listen-Only mode is acknowledged by the state of CAN_ESR1[FLTCONF] field indicating Passive Error. There can be some delay between the Listen-Only mode request and acknowledge. This bit can be written in Freeze mode only because it is blocked by hardware in other modes. 0b: Listen-Only mode is deactivated. 1b: CAN module operates in Listen-Only mode.	0x0
4 LBUF	RW	Lowest Buffer Transmitted First This bit defines the ordering mechanism for Message Buffer transmission. When asserted, CAN_MCR[LPRIOEN] has no effect on the priority arbitration. This bit can be written in the Freeze mode only because it is blocked by hardware in other modes 0b: Buffer with highest priority is transmitted first. 1b: Lowest number buffer is transmitted first.	0x0
5 TSYC	RW	Timer Sync This bit enables a mechanism that resets the free-running timer each time a message is received in Message Buffer 0. This feature provides means to synchronize multiple CAN stations with a special “SYNC” message, that is, global network time. If CAN_MCR[RFEN] is set (Legacy Rx FIFO enabled), the first available Mailbox, according to CAN_CTRL2[RFFN] setting, is used for timer synchronization instead of MB0. This bit can be written in the Freeze mode only because it is blocked by hardware in other modes. 0b: Timer Sync feature is disabled. 1b: Timer Sync feature is enabled.	0x0
6 BOFFREC	RW	This bit defines how CAN recovers from Bus Off state. If this bit is negated, automatic recovering from Bus Off state occurs according to the CAN Specification 2.0B. If the bit is asserted, automatic recovering from Bus Off is disabled, and the module remains in Bus Off state until the user negates the bit. If the negation occurs before 128 sequences of 11 recessive bits are	0x0

Field	Access	Description	Reset Value
		detected on the CAN bus, then Bus Off recovery happens as if the BOFFREC bit had never been asserted. If the negation occurs after 128 sequences of 11 recessive bits occurred, then CAN will re-synchronize to the bus by waiting for 11 recessive bits before joining the bus. After negation, the BOFFREC bit can be re-asserted again during Bus Off, but it will be effective only when the module enters Bus Off. If BOFFREC was negated when the module entered Bus Off, asserting it during Bus Off will not be effective for the current Bus Off recovery. 0b: Automatic recovering from Bus Off state enabled. 1b: Automatic recovering from Bus Off state disabled.	
7 SMP	RW	CAN Bit Sampling This bit defines the sampling mode of CAN bits at the Rx input. It can be written in Freeze mode only because it is blocked by hardware in other modes. NOTE For proper operation, to assert SMP, it is necessary to guarantee a minimum value of 2 TQs in CAN_CTRL1[PSEG1] (or CAN_CBT[EPSEG1]). This bit cannot be asserted when CAN FD is enabled (CAN_MCR[FDEN] = 1). 0b: Just one sample is used to determine the bit value. 1b: Three samples are used to determine the value of the received bit: the regular one (sample point) and 2 preceding samples. A majority rule is used.	0x0
9:8 RSVD_9_8	RO	Reserved	0x0
10 RWRNMSK	RW	Rx Warning Interrupt Mask This bit provides a mask for the Rx Warning Interrupt associated with the RWRNINT flag in the Error and Status Register 1 (CAN_ESR1). This bit is read as zero when CAN_MCR[WRNEN] bit is negated. This bit can be written only if CAN_MCR[WRNEN] bit is asserted. 0b: Rx Warning Interrupt disabled. 1b: Rx Warning Interrupt enabled.	0x0
11 TWRNMSK	RW	Tx Warning Interrupt Mask This bit provides a mask for the Tx Warning Interrupt associated with the TWRNINT flag in the Error and Status Register 1 (CAN_ESR1). This bit is read as zero when CAN_MCR[WRNEN] is negated. This bit can be written only if CAN_MCR[WRNEN] is asserted. 0b: Tx Warning Interrupt disabled. 1b: Tx Warning Interrupt enabled.	0x0
12 LPB	RW	Loop Back Mode This bit configures CAN to operate in Loop-Back mode. In this mode, CAN performs an internal loop back that can be used for the self test operation. The bit stream output of the transmitter is fed back internally to the receiver input. The Rx CAN input pin is ignored and the Tx CAN output goes to the recessive state (logic 1). CAN behaves as it normally does when transmitting, and treats its own transmitted message as a message received from a remote node. In this	0x0

Field	Access	Description	Reset Value
		mode, CAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field, generating an internal acknowledge bit to ensure proper reception of its own message. Both transmit and receive interrupts are generated. This bit can be written only in the Freeze mode because it is blocked by hardware in other modes. NOTE In this mode, CAN_MCR[SRXDIS] cannot be asserted because this will impede the self reception of a transmitted message. CAN_FDCTRL[TDCEN] must be disabled when LPB is asserted. 0b: The loop back mode disabled. 1b: The loop back mode enabled.	
13 RSVD_13	RO	Reserved.	0x0
14 ERRMSK	RW	Error interrupt mask It masks the CAN_ESR1[ERRINT]. 0b: Disable the Error interrupt. 1b: Enable the Error interrupt.	0x0
15 BOFFMSK	RW	Bus off interrupt mask This bit masks the CAN_ESR1[BOFFINT]. 0b: Disable the Bus off interrupt. 1b: Enable the Bus off interrupt.	0x0
18:16 PSEG2	RW	Phase segment 2 This 3-bit field defines the length of Phase Segment 2 in the bit time. The valid programmable values are from 1 to 7. This field can be written only in the Freeze mode because it is blocked by hardware in other modes. Phase Buffer Segment 2 = (PSEG2 + 1) × Time-Quanta.	0x0
21:19 PSEG1	RW	Phase segment 1 This 3-bit field defines the length of Phase Segment 1 in the bit time. The valid programmable values are from 0 to 7. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Phase Buffer Segment 1 = (PSEG1 + 1) × Time-Quanta.	0x0
23:22 RJW	RW	Resync Jump Width This 2-bit field defines the maximum number of time quanta that a bit time can be changed by one resynchronization. One time quantum equals the Sclock period. The valid programmable values are from 0 to 3. This field can be written only in the Freeze mode because it is blocked by hardware in other modes. Resync Jump Width = RJW + 1.	0x0
31:24 PRES DIV	RW	Prescaler Division Factor This 8-bit field defines the ratio between the PE clock frequency and the Serial Clock (Sclock) frequency. The Sclock period defines the time quantum of the CAN protocol. For the reset value, the Sclock frequency is equal to the PE clock frequency. The	0x0

Field	Access	Description	Reset Value
		Maximum value of this field is 0xFF, that gives a minimum Sclock frequency equal to the PE clock frequency divided by 256. This field can be written only in the Freeze mode because it is blocked by hardware in other modes. Sclock frequency = PE clock frequency / (PRESDIV + 1)	

40.3.4 Free Running Timer Register (CAN_TIMER)

40.3.4.1 Offset

Register	Offset
CAN_TIMER	0x08

40.3.4.2 Function

This register represents a 16-bit free running counter that can be read and written by the CPU.

40.3.4.3 Fields

Field	Access	Description	Reset Value
15:0 TIMER	RW	Free running counter value When CAN_CTRL2[TIMER_SRC] is asserted, the timer is continuously incremented by an external time tick. The time tick must be synchronous to the peripheral clock, with a minimum pulse width of one clock cycle. When CAN_CTRL2[TIMER_SRC] is negated, the timer is incremented by the CAN bit clock, which defines the baud rate on the CAN bus. During a message transmission/reception, it increments by one for each bit that is received or transmitted. When there is no message on the bus, it counts by using the previously programmed baud rate. The timer is not incremented during Disable, Stop, Pretended Networking, and Freeze modes. The timer value is captured when the second bit of the identifier field of any frame is on the CAN bus. This captured value is written into the time stamp entry in a message buffer after a successful reception or transmission of a message. If CAN_CTRL1[TSYC] is asserted, the timer is reset whenever a message is received in the first available mailbox, according to CAN_CTRL2[RFFN] setting. The CPU can write to this register anytime. However, if the write occurs at the same time that reception in the first mailbox is resetting the timer, then the write value is discarded.	0x0
31:16 RSVD_31_16	RO	Reserved	0x0

40.3.5 Rx Mailbox Global Mask Register (CAN_RXMGMSK)

40.3.5.1 Offset

Register	Offset
CAN_RXMGMSK	0x10

40.3.5.2 Function

This register is located in RAM, only written in the FREEZE mode and is used to mask the filter fields of all Rx MBs.

40.3.5.3 Fields

Field	Access	Description	Reset Value
31:0 MG	RW	Rx mailbox global mask only valid when CAN_MCR[IRMQ] is low, excluding RX14 and RX15 MG[28:0] : Mask CAN_MBn_ID filter bits MG[29] : Reserved MG[30] : Mask CAN_MBn_CS[IDE] filter bit MG[31] : Mask CAN_MBn_CS[RTR] filter bit 1b: Check the corresponding bit in the filter 0b: Mask the corresponding bit in the filter NOTE: When CAN_MBn_CS[RTR] is negated, the state of CAN_CTRL2[RRS] has no effect on effect on MG. When CAN_MBn_CS[RTR] is asserted, if CAN_CTRL2[RRS] is negated, CAN_MG[31:0] are reserved. Only CAN_CTRL2[EACEN] is asserted, the RTR and IDE bit of mailbox are compared with the RTR and IDE bit of the incoming frame. If CAN_CTRL2[EACEN] is negated, CAN_MG[31:30] are reserved.	-

40.3.6 Rx 14 Mask Register (CAN_RX14MASK)

40.3.6.1 Offset

Register	Offset
CAN_RX14MASK	0x14

40.3.6.2 Function

This register is located in RAM, only written in the FREEZE mode and is used to mask the filter fields of Message Buffer 14.

40.3.6.3 Fields

Field	Access	Description	Reset Value
31:0 RX14M	RW	Rx Buffer 14 Mask Bits Each mask bit masks the corresponding mailbox 14 filter field in the same way that CAN_RXMGMSK	-

Field	Access	Description	Reset Value
		masks other mailboxes' filters. See the description of the CAN_RXMGMSK register. 0b: The corresponding bit in the filter is "don't care". 1b: The corresponding bit in the filter is checked.	

40.3.7 Rx 15 Mask Register (CAN_RX15MASK)

40.3.7.1 Offset

Register	Offset
CAN_RX15MASK	0x18

40.3.7.2 Function

This register is located in RAM, only written in the FREEZE mode and is used to mask the filter fields of Message Buffer 15.

40.3.7.3 Fields

Field	Access	Description	Reset Value
31:0 RX15M	RW	Rx Buffer 15 Mask Bits Each mask bit masks the corresponding mailbox 15 filter field in the same way that CAN_RXMGMSK masks other mailboxes' filters. See the description of the CAN_RXMGMSK register for more details. 0b: The corresponding bit in the filter is "do not care". 1b: The corresponding bit in the filter is checked.	-

40.3.8 Error Counter Register (CAN_ECR)

40.3.8.1 Offset

Register	Offset
CAN_ECR	0x1c

40.3.8.2 Function

This register has four 8-bit fields reflecting the value of the CAN error counters:

- Transmit Error Counter (TXERRCNT field)
- Receive Error Counter (RXERRCNT field)
- Transmit Error Counter for errors detected in the Data Phase of CAN FD messages with the BRS bit set (TX_ERRCNT_FAST field)
- Receive Error Counter for errors detected in the Data Phase of CAN FD messages with the BRS bit set (RX_ERRCNT_FAST field)

40.3.8.3 Fields

Field	Access	Description	Reset Value
7:0 TXERRCNT	RW	Transmit Error Counter Transmit error counter for all errors detected in transmitted messages. The TXERRCNT counter is read-only except in Freeze mode, where the	0x0

Field	Access	Description	Reset Value
		CPU can write it.	
15:8 RXERRCNT	RW	Receive Error Counter Receive error counter for all errors detected in received messages. The RXERRCNT counter is read-only except in Freeze mode, where the CPU can write it.	0x0
23:16 TX_ERRCNT_FAST	RW	Transmit Error Counter for fast bits Transmit error counter for errors detected in the data phase of transmitted CAN FD messages with the BRS bit set. The TX_ERRCNT_FAST counter is read-only except in Freeze mode, where the CPU can only write an 8-bit zero value.	0x0
31:24 RX_ERRCNT_FAST	RW	Receive Error Counter for fast bits Receive error counter for errors detected in the data phase of received CAN FD messages with the BRS bit set. The RX_ERRCNT_FAST counter is read-only except in the Freeze mode, where the CPU can only write an 8-bit zero value.	0x0

40.3.9 CAN Error Status Register (CAN_ESR1)

40.3.9.1 Offset

Register	Offset
CAN_ESR1	0x20

40.3.9.2 Function

This register reports various error conditions detected in the reception and transmission of a CAN frame, some general status of the device and it is the source of some interrupts to the CPU.

40.3.9.3 Fields

Field	Access	Description	Reset Value
0 WAKINT	W1C	Wake Up Interrupt – Applies when the CAN is in low-power mode under Self Wake Up mechanism, either Doze mode or Stop mode: 0b: Indicates no recessive-to-dominant transition was received on the CAN bus. 1b: Indicates a recessive-to-dominant transition was received on the CAN bus.	0x0
1 ERRINT	W1C	Error Interrupt: This bit is cleared by writing it to 1. Writing 0 has no effect. 0b: No such occurrence. 1b: Indicates setting of any error bit in CAN_ESR1.	0x0
2 BOFFINT	W1C	This bit is cleared by writing it to 1. Writing 0 has no effect. 0b: No such occurrence. 1b: CAN module entered Bus Off state.	0x0
3 RX	RO	CAN in Reception This bit indicates if the CAN is receiving a message: 0b: CAN is not receiving a message.	0x0

Field	Access	Description	Reset Value
		1b: CAN is receiving a message.	
5:4 FLTCONF	RO	Fault Confinement State This bit indicates the fault confinement state of the CAN module: 00b: Error Active. 01b: Error Passive. 1xb: Bus Off.	0x0
6 TX	RO	CAN in Transmission This bit indicates if the CAN is transmitting a message: 0b: CAN is not transmitting a message. 1b: CAN is transmitting a message.	0x0
7 IDLE	RO	IDLE This bit indicates when the CAN bus is in the IDLE state: 0b: No such occurrence. 1b: CAN bus is now IDLE.	0x0
8 RXWRN	RO	Rx Error Warning This bit indicates when the repetitive errors are occurring during message reception: 0b: RXERRCNT is lesser than 96. 1b: RXERRCNT is greater than or equal to 96.	0x0
9 TXWRN	RO	TX Error Warning This bit indicates when the repetitive errors are occurring during the message transmission: 0b: TXERRCNT is lesser than 96. 1b: TXERRCNT is greater than or equal to 96.	0x0
10 STFERR	RO	Stuffing Error This register indicates that the receiver node detected a stuffing error in a non-FD message or in a CAN FD message arbitration or Data Phase: 0b: No such occurrence. 1b: A stuffing error occurred since the last read of this register.	0x0
11 FRAMERR	RO	Form Error This bit indicates that a form error was detected by the receiver node in a non-FD message or in a CAN FD message arbitration or Data Phase (that is, a fixed-form bit field contains at least one illegal bit). 0b: No such occurrence. 1b: A Form Error occurred since the last read of this register.	0x0
12 CRCERR	RO	Cyclic Redundancy Check Error This bit indicates that the receiver node detected a CRC error in a non-FD message or in the arbitration or Data Phase of a frame in CAN FD format (that is, the calculated CRC is different from the received CRC). 0b: No such occurrence. 1b: A CRC error occurred since the last read of this register.	0x0
13 ACKERR	RO	Acknowledge Error This bit indicates that an Acknowledge error was detected by the transmitter node (that is, a dominant bit was detected during the ACK SLOT). 0b: No such occurrence. 1b: An ACK error occurred since last read of this register.	0x0

Field	Access	Description	Reset Value
14 BIT0ERR	RO	Bit0 Error This bit indicates when an inconsistency occurs between the transmitted bit and the received bit in a non-CAN FD message or in the arbitration or Data Phase of a CAN FD message: 0b: No such occurrence. 1b: At least one bit sent as dominant is received as recessive.	0x0
15 BIT1ERR	RO	Bit1 Error This bit indicates when an inconsistency occurs between the transmitted bit and the received bit in a non-CAN FD message or in the arbitration or Data Phase of a CAN FD message: 0b: No such occurrence. 1b: At least one bit sent as recessive is received as dominant.	0x0
16 RWRNINT	W1C	Rx Warning Interrupt Flag: 0b: No such occurrence. 1b: The Rx error counter transitioned from less than 96 to greater than or equal to 96.	0x0
17 TWRNINT	W1C	Tx warning interrupt flag: 0b: No such occurrence. 1b: The Tx error counter transitioned from less than 96 to greater than or equal to 96.	0x0
18 SYNCH	RO	Synchronous 0b: No such occurrence. 1b: CAN is synchronized to the CAN bus.	0x0
19 BUSOFFD ONEINT	W1C	Bus off done interrupt BOFFDONEINT is set when the Tx Error Counter (TXERRCNT) has finished counting 128 occurrences of 11 consecutive recessive bits on the CAN bus and is ready to leave Bus Off. If the corresponding mask bit (CAN_CTRL2.BOFFDONEMSK) is set, an interrupt is generated to the CPU. 0b: No such occurrence. 1b: CAN module has completed the Bus Off process.	0x0
20 ERRINT_F AST	W1C	Error Interrupt for Errors Detected in the Data Phase of CAN FD Frames with the BRS Bit Set: This bit is cleared by writing it to 1. Writing 0 has no effect. 0b: No such occurrence. 1b: Indicates setting of any error bit detected in the data phase of CAN FD frames with the BRS bit set.	0x0
21 ERROVR	W1C	Error Overrun This bit indicates that an error condition occurred when any error flag is already set. 0b: Overrun has not occurred. 1b: Overrun has occurred.	0x0
25:22 RSVD_25_ 22	RO	Reserved	0x0
26 STFERR_F AST	RO	Stuffing Error in the Data Phase of CAN FD Frames with the BRS Bit Set 0b: No such occurrence. 1b: A stuffing error occurred since last read of this register.	0x0

Field	Access	Description	Reset Value
27 FRAMERR_FAST	RO	Form Error in the Data Phase of CAN FD Frames with the BRS Bit Set 0b: No such occurrence. 1b: A form error occurred since last read of this register.	0x0
28 CRCERR_FAST	RO	Cyclic Redundancy Check Error in the CRC field of CAN FD Frames with the BRS Bit Set 0b: No such occurrence. 1b: A CRC error occurred since last read of this register.	0x0
29 RSVD_29	RO	Reserved	0x0
30 BIT0ERR_FAST	RO	Bit0 error in data phase of can FD frames with BRS set, useless when CAN_MCR[FDEN]=0 0b: No such occurrence. 1b: At least one bit sent as dominant is received as recessive.	0x0
31 BIT1ERR_FAST	RO	Bit1 error in data phase of can FD frames with BRS set, useless when CAN_MCR[FDEN]=0 0b: No such occurrence. 1b: At least one bit sent as recessive is received as dominant.	0x0

40.3.10 Interrupt Mask 2 Register (CAN_IMASK2)

40.3.10.1 Offset

Register	Offset
CAN_IMASK2	0x24

40.3.10.2 Function

This register allows any number of a range of the 32 Message Buffer Interrupts to be enabled or disabled for MB63 to MB32.

40.3.10.3 Fields

Field	Access	Description	Reset Value
31:0 BUF63TO32M	RW	Buffer MB i Mask Each bit enables or disables the corresponding CAN Message Buffer Interrupt for MB63 to MB32. Thus, enabling the CPU to determine which buffer generates an interrupt after a successful transmission or reception, that is, when the corresponding CAN_IFLAG2 bit is set. 0b: The corresponding buffer interrupt is disabled. 1b: The corresponding buffer interrupt is enabled.	0x0

40.3.11 Interrupt Mask 1 Register (CAN_IMASK1)

40.3.11.1 Offset

Register	Offset

Register	Offset
CAN_IMASK1	0x28

40.3.11.2 Function

This register allows any number of a range of the 32 Message Buffer Interrupts to be enabled or disabled for MB31 to MB0.

40.3.11.3 Fields

Field	Access	Description	Reset Value
31:0 BUF31TO0M	RW	Buffer MB31 -0 Mask Each bit enables or disables the corresponding CAN Message Buffer Interrupt for MB31 to MB0. Thus, enabling the CPU to determine which buffer generates an interrupt after a successful transmission or reception, that is, when the corresponding CAN_IFLAG1 bit is set. 0b: The corresponding buffer interrupt is disabled. 1b: The corresponding buffer interrupt is enabled.	0x0

40.3.12 Interrupt Flag 2 Register (CAN_IFLAG2)

40.3.12.1 Offset

Register	Offset
CAN_IFLAG2	0x2c

40.3.12.2 Function

This register defines the flags for the 32 Message Buffer interrupts for MB63 to MB32.

40.3.12.3 Fields

Field	Access	Description	Reset Value
31:0 BUF63TO32I	W1C	Buffer MB i Interrupt Each bit flags the corresponding CAN Message Buffer Interrupt for MB63 to MB32, i = 31 to 0. 0b: No corresponding buffer MB 32+i has successfully completed transmission or reception. 1b: The corresponding buffer MB 32+i has successfully completed transmission or reception.	0x0

40.3.13 Interrupt Flag 1 Register (CAN_IFLAG1)

40.3.13.1 Offset

Register	Offset
CAN_IFLAG1	0x30

40.3.13.2 Function

This register defines the flags for the 32 Message Buffer interrupts for MB31 to MB0.

40.3.13.3 Fields

Field	Access	Description	Reset Value
0 BUF0I	W1C	Buffer MB0 Interrupt Or Clear FIFO bit When CAN_MCR[RFEN] is cleared (Rx FIFO disabled), this field flags the interrupt for MB0. If the Rx FIFO is enabled, this bit is used to trigger the clear FIFO operation. This operation empties FIFO contents. Before performing this operation, the CPU must service all FIFO-related IFLAGS. When CAN_MCR[DMAE] is enabled, this operation clears the BUF5I flag and, consequently, aborts the DMA request. The clear FIFO operation occurs when the CPU writes 1 in BUF0I. It is only allowed in Freeze mode and is blocked by hardware in other conditions. 0b: The corresponding buffer has no occurrence of completed transmission or reception when CAN_MCR[RFEN]=0. 1b: The corresponding buffer has completed transmission or reception when CAN_MCR[RFEN]=0.	0x0
4:1 BUF4TO1I	W1C	Buffer 1-4 interrupt Buffer MB i Interrupt Or Reserved When CAN_MCR[RFEN] is cleared (Rx FIFO disabled), BUF4TO1I flag the interrupts for MB4 to MB1, i =4,3,2,1 When CAN_MCR[RFEN] is set (Rx FIFO enabled), BUF4TO1I flags are reserved 0b: No interrupt of MBi occurs. 1b: Interrupt of MBi occurs.	0x0
5 BUF5I	W1C	Buffer MB5 Interrupt Or Frames available in Rx FIFO When CAN_MCR[RFEN] is cleared (Rx FIFO disabled), BUF5I flags the interrupt for MB5. When CAN_MCR[RFEN] is set (Rx FIFO enabled), BUF5I represents Frames Available in Rx FIFO and indicates that at least one frame is available to be read from the Rx FIFO. When CAN_MCR[DMAE] is set, BUF5I generates a DMA request and the CPU must not clear BUF5I by writing 1 to it. 0b: No occurrence of MB5 completing transmission/reception when CAN_MCR[RFEN]=0, or of frame(s) available in the FIFO, when CAN_MCR[RFEN]=1. 1b: MB5 completed transmission/reception when CAN_MCR[RFEN]=0, or frame(s) available in the Rx FIFO when CAN_MCR[RFEN]=1. It generates a DMA request in case that CAN_MCR[RFEN] and CAN_MCR[DMAE] are enabled.	0x0
6 BUF6I	W1C	Buffer MB6 Interrupt or Rx FIFO Warning: 0b: No occurrence of MB6 completing transmission/reception when CAN_MCR[RFEN]=0, or of Rx FIFO almost full when CAN_MCR[RFEN]=1. 1b: MB6 completed transmission/reception when CAN_MCR[RFEN]=0, or Rx FIFO almost full when	0x0

Field	Access	Description	Reset Value
		CAN_MCR[RFEN]=1.	
7 BUF7I	W1C	Buffer MB7 Interrupt or Rx FIFO Overflow: 0b: No occurrence of MB7 completing transmission/reception when CAN_MCR[RFEN]=0, or of Rx FIFO overflow when CAN_MCR[RFEN]=1. 1b: MB7 completed transmission/reception when CAN_MCR[RFEN]=0, or Rx FIFO overflow when CAN_MCR[RFEN]=1.	0x0
31:8 BUF31TO8I	W1C	Buffer MBi Interrupt Each bit flags the corresponding CAN Message Buffer Interrupt for MB31 to MB8: 0b: The corresponding buffer has no occurrence of completed transmission or reception. 1b: The corresponding buffer has completed transmission or reception.	0x0

40.3.14 Control 2 Register (CAN_CTRL2)

40.3.14.1 Offset

Register	Offset
CAN_CTRL2	0x34

40.3.14.2 Function

This register complements CAN_CTRL1 providing more control bits.

40.3.14.3 Fields

Field	Access	Description	Reset Value
10:0 RSVD_10_0	RO	Reserved	0x0
11 EDFLTDIS	RW	Edge Filter Disable This bit disables the edge filter used during the bus integration state. When the Edge Filter is enabled, two consecutive nominal time quanta with dominant bus state are required to detect an edge that causes synchronization. When synchronization occurs, the counting of the sequence of eleven consecutive recessive bits is restarted. In addition, the edge filter prevents the dominant pulses that are shorter than a nominal bit time (present during the data phase of an FD frame) from being mistaken for an idle condition. This field is writable only in the Freeze mode. 0b: Edge filter is enabled. 1b: Edge filter is disabled.	0x0
12 ISOCANFDEN	RW	ISO CAN FD Enable This field enables the CAN FD protocol according to ISO specification (ISO 11898-1). This field is writable only in the Freeze mode. 0b: CAN operates using the non-ISO CAN FD protocol.	0x0

Field	Access	Description	Reset Value
		1b: Enables the ISO CAN FD compliant operation.	
13 RSVD_13	RO	Reserved	0x0
14 PREXCEN	RW	Protocol Exception Enable 0b: Disables the protocol exception feature. 1b: Enables the protocol exception feature.	0x0
15 TIMER_SRC	RW	Timer Source This register selects the time tick source used for incrementing the free-running timer counter: 0b: The free-running timer is clocked by the CAN bit clock, which defines the baud rate on the CAN bus. 1b: An external time tick clocks the free-running timer. The period can be either adjusted to be equal to the baud rate on the CAN bus or a different value as required.	0x0
16 EACEN	RW	Entire Frame Arbitration Field Comparison Enable For Rx Mailboxes 0b: Rx mailbox filter's IDE bit is always compared, and RTR is never compared despite mask bits. 1b: Enables the comparison of both the IDE and RTR bits of an Rx mailbox filter with the corresponding bits of the incoming frame. Mask bits do apply.	0x0
17 RRS	RW	Remote Request Storing 0b: Remote Response Frame is generated. 1b: Remote Request Frame is stored.	0x0
18 MRP	RW	Mailboxes Reception Priority: If this bit is set, the matching process starts from the mailboxes, and if no match occurs, the matching continues on the Rx FIFO. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0b: Matching starts from Rx FIFO and continues on mailboxes. 1b: Matching starts from mailboxes and continues on Rx FIFO.	0x0
23:19 TASD	RW	Tx Arbitration Start Delay –This 5-bit field indicates how many CAN bits the Tx arbitration process start point can be delayed from the first bit of CRC field on CAN bus.	0x10
27:24 RFFN	RW	Number of Rx FIFO Filters This bit defines the number of Rx FIFO filters. The maximum selectable number of filters is a function of configuration parameter NUMBER_OF_MB. RFFN must not be programmed with values that cause the number of message buffers occupied by Rx FIFO and ID Filter to exceed the number of mailboxes present, defined by CAN_MCR.MAXMB.	0x0
28 WRMFRZ	RW	Write-Access To Memory In Freeze Mode This bit enables the unrestricted write access to CAN memory in the Freeze mode: 0b: Maintain the write access restrictions. 1b: Enable unrestricted write access to CAN memory.	0x0

Field	Access	Description	Reset Value
29 ECRWRE	RW	Error-Correction Configuration Register Write Enable This bit enables the CAN_MECR register to be updated: 0b: Disable update. 1b: Enable update.	0x0
30 BOFFDONEMSK	RW	Bus Off Done Interrupt Mask – Provides a mask for the Bus Off done interrupt in CAN_ESR1: 0b: Bus Off done interrupt disabled. 1b: Bus Off done interrupt enabled.	0x0
31 ERRMASK_FAST	RW	Error Interrupt Mask for errors detected in the Data Phase of fast CAN FD frames ERRMSK_FAST provides a mask for the ERRINT_FAST Interrupt in CAN_ESR1: 0b: ERRINT_FAST error interrupt disabled. 1b: ERRINT_FAST error interrupt enabled.	0x0

40.3.15 Error and Status 2 Register (CAN_ESR2)

40.3.15.1 Offset

Register	Offset
CAN_ESR2	0x38

40.3.15.2 Function

This register reports some general status information.

40.3.15.3 Fields

Field	Access	Description	Reset Value
12:0 RSVD_12_0	RO	Reserved	0x0
13 IMB	RO	Inactive Mailbox If CAN_ESR2[VPS] is asserted, IMB indicates whether there is any inactive mailbox (CODE field is either 0b1000 or 0b0000).	0x0
14 VPS	RO	Valid Priority Status This bit indicates whether CAN_ESR2[IMB] and CAN_ESR2[LPTM] contents are currently valid or not: 0b: Contents of IMB and LPTM are not valid. 1b: Contents of IMB and LPTM are valid.	0x0
15 RSVD_15	RO	Reserved	0x0
22:16 LPTM	RO	Lowest Priority Tx Mailbox If CAN_ESR2[VPS] is asserted, LPTM indicates the lowest number inactive mailbox (see the CAN_ESR2[IMB] bit description). If there is no inactive mailbox, then the mailbox indicated depends on the CAN_CTRL1[LBUF] bit value.	0x0
31:23 RSVD_31_23	RO	Reserved	0x0

40.3.16 CRC Register (CAN_CRCCR)

40.3.16.1 Offset

Register	Offset
CAN_CRCCR	0x44

40.3.16.2 Function

This register provides information about the CRC of transmitted messages for non FD messages.

40.3.16.3 Fields

Field	Access	Description	Reset Value
14:0 TXCRC	RO	Transmitted CRC Value This bit indicates the CRC value of the last transmitted message for non-FD frames. For CAN FD frames, the CRC value is reported in CAN_FDCRC.	0x0
15 RSVD_15	RO	Reserved	0x0
22:16 MBCRC	RO	CRC Mailbox This bit indicates the number of the mailbox corresponding to the value in the CAN_CRCCR[TXCRC] field.	0x0
31:23 RSVD_31_23	RO	Reserved	0x0

40.3.17 Legacy FIFO Global Mask Register (CAN_RXFGMASK)

40.3.17.1 Offset

Register	Offset
CAN_RXFGMASK	0x48

40.3.17.2 Function

This register is located in RAM, and it is only written in the FREEZE mode.

40.3.17.3 Fields

Field	Access	Description	Reset Value
31:0 FGM	RW	Rx FIFO Global Mask Bits These bits mask the ID Filter Table elements bits in a perfect alignment: 0b: The corresponding bit in the filter is not checked. 1b: The corresponding bit in the filter is checked.	-

40.3.18 RX FIFO ID Indicator (CAN_RXFIR)

40.3.18.1 Offset

Register	Offset
CAN_RXFIR	0x4c

40.3.18.2 Function

RXFIR provides information on Legacy Rx FIFO.

40.3.18.3 Fields

Field	Access	Description	Reset Value
8:0 IDHIT	RO	Identifier Acceptance Filter Hit Indicator This bit indicates which Identifier Acceptance Filter was hit by the received message that is in the output of the Rx FIFO.	-
31:9 RSVD_31_9	RO	Reserved	-

40.3.19 CAN Bit Timing Register (CAN_CBT)

40.3.19.1 Offset

Register	Offset
CAN_CBT	0x50

40.3.19.2 Function

This register is an alternative way to store the CAN bit timing variables described in CAN_CTRL1 register.

40.3.19.3 Fields

Field	Access	Description	Reset Value
4:0 EPSEG2	RW	Extended Phase Segment 2 When CAN_CBT[BTF] is asserted, EPSEG2 defines the length of Phase Segment 2 in the bit time. EPSEG2 extends the CAN_CTRL1.	0x0
9:5 EPSEG1	RW	Extended Phase Segment 1 When CAN_CBT[BTF] is asserted, EPSEG1 defines the length of Phase Segment 1 in the bit time. EPSEG1 extends the CAN_CTRL1.	0x0
15:10 EPROPSEG	RW	Extended Propagation Segment When CAN_CBT[BTF] is asserted, EPROPSEG defines the length of the Propagation Segment in the bit time. EPROPSEG extends the CAN_CTRL1.	0x0
20:16 ERJW	RW	Extended Resync Jump Width When CAN_CBT[BTF] is asserted, ERJW defines the maximum number of time quanta that a bit time can be changed by one re-synchronization.	0x0
30:21 EPRES DIV	RW	Extended Prescaler Division Factor When CAN_CBT[BTF] is asserted, EPRES DIV defines the ratio between the PE clock frequency and the Serial Clock (Sclock) frequency. EPRES DIV	0x0

Field	Access	Description	Reset Value
		extends the CAN_CTRL1[PRES DIV] value range.	
31 BTF	RW	Bit Timing Format Enable This bit enables the use of extended CAN bit timing fields EPRES DIV, EPROPSEG, EPSEG1, EPSEG2, and ERJW, replacing the CAN bit timing variables defined in CAN_CTRL1.	0x0

40.3.20 Interrupt Mask 4 Register (CAN_IMASK4)

40.3.20.1 Offset

Register	Offset
CAN_IMASK4	0x68

40.3.20.2 Function

This register allows any number of a range of the 32 Message Buffer Interrupts to be enabled or disabled for MB127 to MB96.

40.3.20.3 Fields

Field	Access	Description	Reset Value
31:0 BUF127TO96M	RW	Buffer MB 127-96 Mask Each bit enables or disables the corresponding CAN Message Buffer Interrupt for MB127 to MB96.	0x0

40.3.21 Interrupt Mask 3 Register (CAN_IMASK3)

40.3.21.1 Offset

Register	Offset
CAN_IMASK3	0x6c

40.3.21.2 Function

This register allows any number of a range of the 32 Message Buffer Interrupts to be enabled or disabled for MB95 to MB64.

40.3.21.3 Fields

Field	Access	Description	Reset Value
31:0 BUF95TO64M	RW	Buffer MB 95-64 Mask Each bit enables or disables the corresponding CAN Message Buffer Interrupt for MB95 to MB64.	0x0

40.3.22 Interrupt Flag 4 Register (CAN_IFLAG4)

40.3.22.1 Offset

Register	Offset

Register	Offset
CAN_IFLAG4	0x70

40.3.22.2 Function

This register defines the flags for the 32 Message Buffer interrupts for MB127 to MB96.

40.3.22.3 Fields

Field	Access	Description	Reset Value
31:0 BUF127TO96I	W1C	Buffer MB 127-96 Interrupt Each bit flags the corresponding CAN Message Buffer Interrupt for MB127 to MB96.	0x0

40.3.23 Interrupt Flag 3 Register (CAN_IFLAG3)

40.3.23.1 Offset

Register	Offset
CAN_IFLAG3	0x74

40.3.23.2 Function

This register defines the flags for the 32 Message Buffer interrupts for MB95 to MB64.

40.3.23.3 Fields

Field	Access	Description	Reset Value
31:0 BUF95TO64I	W1C	Buffer MB 95-64 Interrupt Each bit flags the corresponding CAN Message Buffer Interrupt for MB95 to MB64, i=31 to 0. 0b: No corresponding buffer MB 64+i has successfully completed transmission or reception. 1b: The corresponding buffer MB 64+i has successfully completed transmission or reception.	0x0

40.3.24 Message Buffer Control and Status Register (CAN_MBn_CS)

40.3.24.1 Offset

Register	Offset
CAN_MBn_CS	0x80+16*n (for CAN0-5, n=0-63; for CAN6-7, n=0-127)

40.3.24.2 Fields

Field	Access	Description	Reset Value
15:0 TIME_STAMP	RW	TIME STAMP — Free-Running Counter Time Stamp This 16-bit field is a copy of the Free-Running Timer, captured for Tx and Rx frames at the time when the beginning of the Identifier field appears on the CAN	-

Field	Access	Description	Reset Value
		bus.	
19:16 DLC	RW	DLC — Length of Data in Bytes This 4-bit field is the length (in bytes) of the Rx or Tx data, which is located in offset 0x8 through 0xF of the MB space. In reception, this field is written by the CAN module, copied from the DLC (Data Length Code) field of the received frame. In transmission, the CPU writes this field and this field corresponds to the DLC field value of the frame to be transmitted. When RTR = 1, the frame to be transmitted is a remote frame and does not include the data field, regardless of the DLC field.	-
20 RTR	RW	RTR — Remote Transmission Request This bit affects the behavior of remote frames and is part of the reception filter. See the description of the RRS field in Control 2 register (CAN_CTRL2) for additional details. If CAN transmits this bit as '1' (recessive) and receives it as '0' (dominant), it is interpreted as an arbitration loss. If this bit is transmitted as '0' (dominant), then the CAN module treats it as a bit error if it is received as '1' (recessive). However, if the value received matches the value transmitted, it is considered a successful bit transmission. 0b: Indicates the current MB has a data frame to be transmitted. For example, in Rx MB, it may be considered in matching processes. 1b: Indicates the current MB may have a remote request frame to be transmitted if MB is Tx. For example, if the MB is Rx, then incoming remote request frames may be stored.	-
21 IDE	RW	IDE — ID Extended Bit This field identifies whether the frame format is standard or extended. 0b: Frame format is standard. 1b: Frame format is extended.	-
22 SRR	RW	SRR — Substitute Remote Request Fixed recessive bit, used only in extended format. It must be set to one by the user for transmission (Tx Buffers) and will be stored with the value received on the CAN bus for Rx receiving buffers. It can be received as either recessive or dominant. If CAN receives this bit as dominant, then it is interpreted as an arbitration loss. 0b: Dominant is not a valid value for transmission in extended format frames. 1b: Recessive value is compulsory for transmission in extended format frames.	-
23 RSVD_23	RO	Reserved	-
27:24 CODE	RW	CODE — Message Buffer Code This 4-bit field can be accessed (read or write) by the CPU and by the CAN module itself, as part of the message buffer matching and arbitration process.	-
28 RSVD_28	RO	Reserved	-

Field	Access	Description	Reset Value
29 ESI	RW	ESI — Error State Indicator This bit indicates if the transmitting node is error active or error passive.	-
30 BRS	RW	BRS — Bit Rate Switch This bit defines whether the bit rate is switched inside a CAN FD format frame.	-
31 EDL	RW	EDL — Extended Data Length This bit distinguishes between CAN format and CAN FD format frames. The EDL bit must not be set for message buffers configured to CAN_RANSWER with code field 0b1010.	-

40.3.25 Message Buffer ID Register (CAN_MBn_ID)

40.3.25.1 Offset

Register	Offset
CAN_MBn_ID	0x84+16*n (for CAN0-5, n=0-63; for CAN6-7, n=0-127)

40.3.25.2 Fields

Field	Access	Description	Reset Value
17:0 ID_EXT	RW	ID extended.	-
28:18 ID_STD_EXT	RW	ID — Frame Identifier In standard frame format, only the 11 most significant bits (28 to 18) are used for frame identification in both receive and transmit cases. The 18 least significant bits (17 to 0) are ignored. In the extended frame format, all bits are used for frame identification in both receive and transmit cases.	-
31:29 PRIO	RO	PRIO — Local priority This 3-bit field is used only when CAN_MCR[LPRIOEN] is set, and it only makes sense for the Tx mailboxes. These bits are not transmitted. They are appended to the regular ID to define the transmission priority.	-

40.3.26 Message Buffer Data03 Register (CAN_MBn_DATA03)

40.3.26.1 Offset

Register	Offset
CAN_MBn_DATA03	0x88+16*n (for CAN0-5, n=0-63; for CAN6-7, n=0-127)

40.3.26.2 Fields

Field	Access	Description	Reset Value
31:0 DATA_BYTE03	RW	DATA BYTE0, Field 31:24 DATA BYTE1, Field 23:16 DATA BYTE2, Field 15:8 DATA BYTE3, Field 7:0	-

40.3.27 Message Buffer Data47 Register (CAN_MBn_DATA47)

40.3.27.1 Offset

Register	Offset
CAN_MBn_DATA47	0x8C+16*n (for CAN0-5, n=0-63; for CAN6-7, n=0-127)

40.3.27.2 Fields

Field	Access	Description	Reset Value
31:0 DATA_BYTE47	RW	DATA BYTE4, Field 31:24 DATA BYTE5, Field 23:16 DATA BYTE6, Field 15:8 DATA BYTE7, Field 7:0	-

40.3.28 RX Individual Mask Register (CAN_RXIMRn)

40.3.28.1 Offset

Register	Offset
CAN_RXIMRn	0x880+4*n (for CAN0-5, n=0-63; for CAN6-7, n=0-127)

40.3.28.2 Function

These registers are used to store the acceptance masks for ID filtering in Rx MBs and the Legacy Rx FIFO.

40.3.28.3 Fields

Field	Access	Description	Reset Value
31:0 MI	RW	Individual Mask Bits Each individual mask bit masks the corresponding bit in both the mailbox filter and Rx FIFO ID filter table element in distinct ways. For mailbox filters, see the CAN_RXMGMSK register description. For Rx FIFO ID filter table elements, see the Legacy Rx FIFO structure description. 0b: The corresponding bit in the filter is "do not care". 1b: The corresponding bit in the filter is checked.	-

40.3.29 Memory Error Control Register (CAN_MECR)

40.3.29.1 Offset

Register	Offset
CAN_MECR	0xAE0

40.3.29.2 Function

This register contains control bits for memory error detection and correction (ECC).

40.3.29.3 Fields

Field	Access	Description	Reset Value
6:0 RSVD_6_0	RO	Reserved	0x0
7 NCEFAFRZ	RW	Non-Correctable Errors In CAN Access Put Device In Freeze Mode Determines the response when a non-correctable error is detected in a memory read performed by CAN internal processes. In this case, entering Freeze mode prevents corrupted data from being treated as valid by CAN internal processes. 0b: Keep normal operation. 1b: Put CAN in the Freeze mode (according to the "Freeze mode" section).	0x1
8 ECCDIS	RW	Error Correction Disable Disables completely the memory detection and correction mechanism. Besides disabling the error report mechanism, it also stops updating the error-related flags and generation of bus transfer errors. The parity bits continue being calculated and written into memory on write transactions. 0b: Enable memory error correction. 1b: Disable memory error correction.	0x0
9 RERRDIS	RW	Error Report Disable Disables the update of the error report registers. The update of error-related flags and the generation of bus transfer errors are still active. 0b: Enable the updates of the error report registers. 1b: Disable the updates of the error report registers. NOTE When reading the report registers, this bit must be set to assure coherence on the consecutive register reads.	0x0
12:10 RSVD_12_10	RO	Reserved	0x0
13 EXTERRIE	RW	Extended Error Injection Enable Memory accesses performed by internal CAN processes are 64-bit. This bit extends the error injection on 32-bit memory accesses to the complementary 32-bit word using the same 32-bit error injection data and parity words. See Error Injection Data Pattern Register (CAN_ERRIDPR) and Error Injection Parity Pattern Register (CAN_ERRIPPR). 0b: The error injection is applied only to the 32-bit word. 1b: The error injection is applied to the 64-bit word.	0x0
14 FAERRIE	RW	CAN Access Error Injection Enable Enables the injection of errors only in memory reads issued by the CAN internal processes. 0b: Injection is disabled. 1b: Injection is enabled.	0x0
15 HAERRI	RW	Host Access Error Injection Enable Enables the injection of errors only in memory reads issued by the host (CPU). 0b: Injection is disabled.	0x0

Field	Access	Description	Reset Value
		1b: Injection is enabled.	
16 CEI_MSK	RW	Correctable Errors Interrupt Mask Enables the interrupt in case of correctable errors detected in memory reads issued by the host or CAN internal processes. 0b: Interrupt is disabled. 1b: Interrupt is enabled.	0x0
17 RSVD_17	RO	Reserved	0x0
18 FANCEI_MSK	RW	CAN Access With Non-Correctable Errors Interrupt Mask Enables the interrupt in case of non-correctable errors detected in memory reads issued by the CAN internal processes. 0b: Interrupt is disabled. 1b: Interrupt is enabled.	0x1
19 HANCEI_MSK	RW	Host Access with Non-Correctable Errors Interrupt Mask Enables the interrupt in case of non-correctable errors detected in memory reads issued by the host (CPU): 0b: Interrupt is disabled. 1b: Interrupt is enabled.	0x1
30:20 RSVD_30_20	RO	Reserved	0x0
31 ECRWRDIS	RW	Error Configuration Register Write Disable Disables writes on this register. This bit is automatically set to 1 (disabled) when CAN_CTRL2[ECRWRE] is enabled. 0b: Write is enabled. 1b: Write is disabled.	0x1

40.3.30 Error Injection Address Register (CAN_ERRIAR)

40.3.30.1 Offset

Register	Offset
CAN_ERRIAR	0xAE4

40.3.30.2 Function

This register holds the address where error is to be injected.

40.3.30.3 Fields

Field	Access	Description	Reset Value
1:0 INJADDR_L	RO	Error Injection Address Low This is a read-only field that defines the two least significant bits of the physical RAM address where a memory read error is to be injected.	0x0
13:2 INJADDR_H	RW	Error Injection Address High This bit defines the twelve most significant bits of the physical RAM address where a memory read error is to be injected.	0x0

Field	Access	Description	Reset Value
31:14 RSVD_31_14	RO	Reserved	0x0

40.3.31 Error Injection Data Pattern Register (CAN_ERRIDPR)

40.3.31.1 Offset

Register	Offset
CAN_ERRIDPR	0xAE8

40.3.31.2 Function

This register is used to hold the error pattern to be injected in the data word read from memory.

40.3.31.3 Fields

Field	Access	Description	Reset Value
31:0 DFLIP	RW	Data Flip Pattern Bits set to 1 in the flip pattern cause the corresponding data bit in the word read from memory to invert.	0x0

40.3.32 Error Injection Parity Pattern Register (CAN_ERRIPPR)

40.3.32.1 Offset

Register	Offset
CAN_ERRIPPR	0xAEC

40.3.32.2 Function

This register is used to hold the error pattern to be injected in parity bits read from memory along with data word.

40.3.32.3 Fields

Field	Access	Description	Reset Value
4:0 PFLIP0	RW	Parity Flip Pattern For Byte 0	0x0
7:5 RSVD_7_5	RO	Reserved	0x0
12:8 PFLIP1	RW	Parity Flip Pattern For Byte 1	0x0
15:13 RSVD_15_13	RO	Reserved	0x0
20:16 PFLIP2	RW	Parity Flip Pattern For Byte 2	0x0
23:21 RSVD_23_21	RO	Reserved	0x0
28:24 PFLIP3	RW	Parity Flip Pattern For Byte 3 (most significant)	0x0

Field	Access	Description	Reset Value
31:29 RSVD_31_29	RO	Reserved	0x0

40.3.33 Error Report Address Register (CAN_RERRAR)

40.3.33.1 Offset

Register	Offset
CAN_RERRAR	0xAF0

40.3.33.2 Function

This register is used to report the address used for an access in which an error was detected, and also report the identification of the source of that access.

40.3.33.3 Fields

Field	Access	Description	Reset Value
13:0 ERRADDR	RO	Address Where the Error was Detected See the description of the Error Injection Address Register (CAN_ERRIAR).	0x0
15:14 RSVD_15_14	RO	Reserved	0x0
18:16 SAID	RO	SAID[2] – Identification of the requestor of the memory read request: 0b: Requested by CAN internal processes 1b: Requested by host (CPU) SAID[1] – Details of CAN operation: 0b: Move 1b: Scanning SAID[0] – Operation that requested the memory read: 0b: Transmission 1b: Reception	0x0
23:19 RSVD_23_19	RO	Reserved	0x0
24 NCE	RO	Non-Correctable Error – Indicates that the report is due to a non-correctable error: 0b: Reporting a correctable error 1b: Reporting a non-correctable error	0x0
31:25 RSVD_31_25	RO	Reserved	0x0

40.3.34 Error Report Data Register (CAN_RERRDR)

40.3.34.1 Offset

Register	Offset
CAN_RERRDR	0xAF4

40.3.34.2 Function

This register is used to report the raw data (unmodified by the correction performed by ECC logic) read from memory with error.

40.3.34.3 Fields

Field	Access	Description	Reset Value
31:0 RDATA	RO	Raw data word read from memory with error	0x0

40.3.35 Error Report Syndrome Register (CAN_RERRSYNR)

40.3.35.1 Offset

Register	Offset
CAN_RERRSYNR	0xAF8

40.3.35.2 Function

This register is used to hold the syndrome detected in a memory read with error. It also reports the bytes which were read in this 32-bit read transaction.

40.3.35.3 Fields

Field	Access	Description	Reset Value
4:0 SYND0	RO	Error Syndrome For Byte 0	0x0
6:5 RSVD_6_5	RO	Reserved	0x0
7 BE0	RO	Byte Enabled For Byte 0 (least significant) 0b: The byte was not read. 1b: The byte was read.	0x0
12:8 SYND1	RO	Error Syndrome For Byte 1	0x0
14:13 RSVD_14_13	RO	Reserved	0x0
15 BE1	RO	Byte Enabled For Byte 1 0b: The byte was not read. 1b: The byte was read.	0x0
20:16 SYND2	RO	Error Syndrome For Byte 2	0x0
22:21 RSVD_22_21	RO	Reserved	0x0
23 BE2	RO	Byte Enabled For Byte 2 0b: The byte was not read. 1b: The byte was read.	0x0
28:24 SYND3	RO	Error Syndrome For Byte 3 (most significant)	0x0
30:29 RSVD_30_29	RO	Reserved	0x0
31 BE3	RO	Byte Enabled For Byte 3 (most significant) 0b: The byte was not read. 1b: The byte was read.	0x0

40.3.36 Error Status Register (CAN_ERRSR)

40.3.36.1 Offset

Register	Offset
CAN_ERRSR	0xAFC

40.3.36.2 Function

This register is used to hold the status bits of the error correction and detection operations.

40.3.36.3 Fields

Field	Access	Description	Reset Value
0 CEIOF	W1C	Correctable Error Interrupt Overrun Flag Indicates that a correctable error was detected in a memory read when CEIF was set. No interrupt is associated with this flag. 0b: No overrun on correctable errors. 1b: Overrun on correctable errors.	0x0
1 RSVD_1	RO	Reserved	0x0
2 FANCEIOF	W1C	CAN Access With Non-Correctable Error Interrupt Overrun Flag Indicates that a non-correctable error was detected in a memory read initiated by CAN internal processes when FANCEIF was set. No interrupt is associated to this flag. 0b: No overrun on non-correctable errors in CAN access. 1b: Overrun on non-correctable errors in CAN access.	0x0
3 HANCEIOF	W1C	Host Access with Non-Correctable Error Interrupt Overrun Flag Indicates that a non-correctable error was detected in a memory read initiated by Host when HANCEIF was set. No interrupt is associated to this flag. 0b: No overrun on non-correctable errors in Host access. 1b: Overrun on non-correctable errors in Host access.	0x0
15:4 RSVD_15_4	RO	Reserved	0x0
16 CEIF	W1C	Correctable Error Interrupt Flag: Indicates that a correctable error was detected in a memory read. If CAN_MECR[CEI_MSK] is set, the interrupt is asserted. 0b: No correctable errors were detected so far. 1b: A correctable error was detected.	0x0
17 RSVD_17	RO	Reserved	0x0
18 FANCEIF	W1C	CAN Access with Non-Correctable Error Interrupt Flag: Indicates that a non-correctable error was detected in a memory read initiated by CAN internal processes. If	0x0

Field	Access	Description	Reset Value
		CAN_MECR[FANCEI_MSK] is set, the interrupt is asserted. 0b: No non-correctable errors were detected in CAN accesses so far. 1b: A non-correctable error was detected in a CAN access.	
19 HANCEIF	W1C	Host Access with Non-Correctable Error Interrupt Flag: Indicates that a non-correctable error was detected in a memory read initiated by the Host. A bus transfer error is asserted for that access. If CAN_MECR[HANCEI_MSK] is set, the interrupt is asserted. 0b: No non-correctable errors were detected in Host accesses so far. 1b: A non-correctable error was detected in a Host access.	0x0
31:20 RSVD_31_20	RO	Reserved	0x0

40.3.37 Pretend Network Control 1 Register (CAN_CTRL1_PN)

40.3.37.1 Offset

Register	Offset
CAN_CTRL1_PN	0xB00

40.3.37.2 Function

This register contains control bits for Pretended Networking mode filtering selection.

40.3.37.3 Fields

Field	Access	Description	Reset Value
1:0 FCS	RW	Filtering Combination Selection This 2-bit field selects the filtering criteria to be applied when CAN is under Pretended Networking mode. 00b: Message ID filtering only. 01b: Message ID filtering and payload filtering. 10b: Message ID filtering occurring a specified number of times. 11b: Message ID filtering and payload filtering a specified number of times.	0x0
3:2 IDFS	RW	ID Filtering Selection This 2-bit field selects the level of ID filtering to be applied when CAN is under Pretended Networking mode. In ID filtering, the IDE and RTR bits are also considered as part of the reception filter if IDE_MSK and RTR_MSK bits in the CAN_FLT_ID2_IDMASK register are set. 00b: Match upon ID contents against an exact target value. 01b: Match upon an ID value greater than or equal to	0x0

Field	Access	Description	Reset Value
		a specified target value. 10b: Match upon an ID value smaller than or equal to a specified target value. 11b: Match upon an ID value inside a range, greater than or equal to a specified lower limit, and smaller than or equal to a specified upper limit.	
5:4 PLFS	RW	Payload Filtering Selection This 2-bit field selects the payload filtering level when CAN is under Pretended Networking mode. Filtering does not accept remote messages (RTR=1) when payload filtering is active. 00b: Match upon payload contents against an exact target value. 01b: Match upon a payload value greater than or equal to a specified target value. 10b: Match upon a payload value smaller than or equal to a specified target value. 11b: Match upon a payload value inside a range, greater than or equal to a specified lower limit, and smaller than or equal to a specified upper limit.	0x0
7:6 RSVD_7_6	RO	Reserved	0x0
15:8 NMATCH	RW	Number of Messages Matching the Same Filtering Criteria This 8-bit field defines the number of times a given message must match the predefined filtering criteria for ID and/or PL before generating a wakeup event. This quantity can be configured in the 1 to 255 range using values from 0x01 to 0xFF, respectively. 00000001b – The received message must match the predefined filtering criteria for ID and/or PL once before generating a wakeup event. 00000010b - The received message must match the predefined filtering criteria for ID and/or PL twice before generating a wakeup event. 11111111b - The received message must match the predefined filtering criteria for ID and/or PL 255 times before generating a wakeup event.	0x1
16 WUMF_MSK	RW	Wake Up by Match Flag Mask Bit Masks the generation of a wakeup event originated by a successful filtered Rx message. 0b: The wakeup match event is disabled. 1b: The wakeup match event is enabled.	0x0
17 WTOF_MSK	RW	Wake Up by Timeout Flag Mask Bit This bit masks the generation of a wakeup event originated by a timeout. 0b: The timeout wakeup event is disabled. 1b: The timeout wakeup event is enabled.	0x0
31:18 RSVD_31_18	RO	Reserved	0x0

40.3.38 Pretend Network Control 2 Register (CAN_CTRL2_PN)

40.3.38.1 Offset

Register	Offset
CAN_CTRL2_PN	0xB04

40.3.38.2 Function

This register contains configuration bits for the timeout value under Pretended Networking mode.

40.3.38.3 Fields

Field	Access	Description	Reset Value
15:0 MATCHTO	RW	Timeout for No Message Matching the Filtering Criteria MATCHTO defines a timeout value that generates a wakeup event if CAN_MCR.PNET_EN is asserted. If the timeout counter reaches the target value when the CAN is under Pretended Networking mode, a wakeup event is generated. The timeout limit can be configured from 1 to 65535 to control an internal 16-bit up-count timer to produce a trigger upon reaching this configured value. The internal timer is incremented based on periodic time ticks, of which the period is 64 times the CAN Bit Time unit. It can be written in the Freeze mode only.	0x0
31:16 RSVD_31_16	RO	Reserved	0x0

40.3.39 Pretend Network Wakeup Match Register (CAN_WU_MTC)

40.3.39.1 Offset

Register	Offset
CAN_WU_MTC	0xB08

40.3.39.2 Function

This register contains wake up information related to the matching processes performed while CAN receives frames under Pretended Networking mode.

40.3.39.3 Fields

Field	Access	Description	Reset Value
7:0 RSVD_7_0	RESERVE	Reserved	0x0
15:8 MCOUNTER	RO	Number of Matches when in Pretended Networking This 8-bit field reports the number of times a given message has matched the predefined filtering criteria for ID and/or PL before a wakeup event. This register is reset by CAN when it enters in Pretended Networking mode, and is not affected by soft reset.	0x0
16	R/W1C	Wake Up by Match Flag Bit	0x0

Field	Access	Description	Reset Value
WUMF		This bit identifies whether the CAN has detected a matching Rx incoming message, which passed the filtering criteria specified in the CAN_CTRL1_PN register. In addition, this flag generates a wake-up event if CAN_CTRL1_PN[WUMF_MSK] is enabled. 0b: No wakeup by match event detected. 1b: Wakeup by match event detected.	
17 WTOF	R/W1C	Wake Up by Timeout Flag Bit This bit identifies whether the CAN has detected a timeout event during a time interval defined by CAN_CTRL2_PN[MATCHTO]. This flag generates a wakeup event if CAN_CTRL1_PN[WTOF_MSK] is enabled. 0b: No wakeup by timeout event detected. 1b: Wakeup by timeout event detected.	0x0
31:18 RSVD_31_18	RO	Reserved	0x0

40.3.40 Pretend Network ID Filter 1 Register (CAN_FLT_ID1)

40.3.40.1 Offset

Register	Offset
CAN_FLT_ID1	0xB0C

40.3.40.2 Function

This register contains FLT_ID1 target value, as well as IDE and RTR target values used to filter incoming message ID.

40.3.40.3 Fields

Field	Access	Description	Reset Value
28:0 FLT_ID1	RW	ID Filter 1 for Pretended Networking Filtering This bit defines either the 29 bits of an extended frame format, considering all bits, or the 11 bits of a standard frame format, considering just the 11 leftmost bits.	0x0
29 FLT_RTR	RW	Remote Transmission Request Filter This bit identifies whether the frame is remote or not. It is used as part of the ID reception filter. 0b: Reject the remote frame (accept data frame). 1b: Accept the remote frame.	0x0
30 FLT_IDE	RW	ID Extended Filter This bit identifies whether the frame format is standard or extended. It is used as part of the ID reception filter. 0b: Accept the standard frame format. 1b: Accept the extended frame format.	0x0
31 RSVD_31	RO	Reserved	0x0

40.3.41 Pretend Network DLC Filter Register (CAN_FLT_DLC)

40.3.41.1 Offset

Register	Offset
CAN_FLT_DLC	0xB10

40.3.41.2 Function

This register contains the DLC inside range target values (FLT_DLC_LO and FLT_DLC_HI) used to filter incoming message.

40.3.41.3 Fields

Field	Access	Description	Reset Value
3:0 FLT_DLC_HI	RW	Upper Limit for Length of Data Bytes Filter – Specifies the upper limit for the number of data bytes considered valid for payload comparison. FLT_DLC_HI is used as part of the payload reception filter.	0x8
15:4 RSVD_15_4	RO	Reserved	0x0
19:16 FLT_DLC_LO	RW	Lower Limit for Length of Data Bytes Filter – Specifies the lower limit for the number of data bytes considered valid for payload comparison. FLT_DLC_LO is used as part of the payload reception filter.	0x0
31:20 RSVD_31_20	RW	Reserved	0x0

40.3.42 Pretend Network Payload Low Filter 1 Register (CAN_PL1_LO)

40.3.42.1 Offset

Register	Offset
CAN_PL1_LO	0xB14

40.3.42.2 Function

This register contains Payload Filter 1 low order bits of the target value used to filter incoming message payload.

40.3.42.3 Fields

Field	Access	Description	Reset Value
7:0 Data_byte_3	RW	Payload Filter 1 low order bits for Pretended Networking payload filtering corresponding to data byte 3	0x0
15:8 Data_byte_2	RW	Payload Filter 1 low order bits for Pretended Networking payload filtering corresponding to data byte 2	0x0
23:16 Data_byte_1	RW	Payload Filter 1 low order bits for Pretended Networking payload filtering corresponding to data byte 1	0x0

Field	Access	Description	Reset Value
31:24 Data_byte_0	RW	Payload Filter 1 low order bits for Pretended Networking payload filtering corresponding to data byte 0	0x0

40.3.43 Pretend Network Payload High Filter 1 Register (CAN_PL1_HI)

40.3.43.1 Offset

Register	Offset
CAN_PL1_HI	0xB18

40.3.43.2 Function

This register contains Payload Filter 1 high order bits of the target value used to filter incoming message payload.

40.3.43.3 Fields

Field	Access	Description	Reset Value
7:0 Data_byte_7	RW	Payload Filter 1 low order bits for Pretended Networking payload filtering corresponding to data byte 7	0x0
15:8 Data_byte_6	RW	Payload Filter 1 low order bits for Pretended Networking payload filtering corresponding to data byte 6	0x0
23:16 Data_byte_5	RW	Payload Filter 1 low order bits for Pretended Networking payload filtering corresponding to data byte 5	0x0
31:24 Data_byte_4	RW	Payload Filter 1 low order bits for Pretended Networking payload filtering corresponding to data byte 4	0x0

40.3.44 Pretended Networking ID Filter 2 Register / ID Mask Register (CAN_FLT_ID2_IDMASK)

40.3.44.1 Offset

Register	Offset
CAN_FLT_ID2_IDMASK	0xB1C

40.3.44.2 Function

This register contains FLT_ID2 target value used only as the upper limit value in ID range detection. Also, when exact ID filtering criteria is selected, this register is used to store the ID mask.

40.3.44.3 Fields

Field	Access	Description	Reset Value

Field	Access	Description	Reset Value
28:0 FLT_ID2_IDMASK	RW	ID Filter 2 for Pretended Networking Filtering / ID Mask Bits for Pretended Networking ID Filtering This register is used to define either the ID filter value in extended frame format (29 bits), considering the CAN_FLT_ID2[28:0], or the ID filter value in standard frame format (11 bits), considering the CAN_FLT_ID2[28:18] (other bits in the [17:0] range have no meaning). It is used only in the range of ID filtering. It can also be used to define either the mask values for the extended frame format (29 bits), considering the IDMASK[28:0], or for the standard frame format (11 bits), considering the IDMASK[28:18] (other bits in the [17:0] range have no meaning). It is used only in exact ID filtering.	0x0
29 RTR_MSK	RW	Remote Transmission Request Mask Bit – Indicates whether the frame type (data/remote) is part of the ID reception filter 0b: The corresponding bit in the filter is “do not care”. 1b: The corresponding bit in the filter is checked.	0x0
30 IDE_MSK	RW	ID Extended Mask Bit This bit indicates whether the frame format (standard/extended) is used as part of the ID reception filter. 0b: The corresponding bit in the filter is “do not care”. 1b: The corresponding bit in the filter is checked.	0x0
31 RSVD_31	RO	Reserved	0x0

40.3.45 Pretended Networking Payload Low Filter 2 Register / Payload Low Mask Register (CAN_PL2_PLMASK_LO)

40.3.45.1 Offset

Register	Offset
CAN_PL2_PLMASK_LO	0xB20

40.3.45.2 Function

This register has two functions. First, it contains the low order bits for the Payload Filter 2 used only as the upper limit value in a payload range detection. Second, when exact payload filtering criteria is selected, this register is used as payload mask.

40.3.45.3 Fields

Field	Access	Description	Reset Value
7:0 Data_byte_3	RW	Payload Filter 2 low order bits / Payload Mask low order bits for Pretended Networking payload filtering corresponding to data byte 3.	0x0
15:8 Data_byte_2	RW	Payload Filter 2 low order bits / Payload Mask low order bits for Pretended Networking payload filtering	0x0

Field	Access	Description	Reset Value
		corresponding to data byte 2.	
23:16 Data_byte_1	RW	Payload Filter 2 low order bits / Payload Mask low order bits for Pretended Networking payload filtering corresponding to data byte 1.	0x0
31:24 Data_byte_0	RW	Payload Filter 2 low order bits / Payload Mask low order bits for Pretended Networking payload filtering corresponding to data byte 0.	0x0

40.3.46 Pretended Networking Payload High Filter 2 Low Order Bits / Payload High Mask Register (CAN_PL2_PLMASK_HI)

40.3.46.1 Offset

Register	Offset
CAN_PL2_PLMASK_HI	0xB24

40.3.46.2 Function

This register has two functions. First, it contains the high order bits for the Payload Filter 2 used only as the upper limit value in a payload range detection. Second, when exact payload filtering criteria is selected, this register is used as payload mask.

40.3.46.3 Fields

Field	Access	Description	Reset Value
7:0 Data_byte_7	RW	Payload Filter 2 high order bits / Payload Mask high order bits for Pretended Networking payload filtering corresponding to data byte 7.	0x0
15:8 Data_byte_6	RW	Payload Filter 2 high order bits / Payload Mask high order bits for Pretended Networking payload filtering corresponding to data byte 6.	0x0
23:16 Data_byte_5	RW	Payload Filter 2 high order bits / Payload Mask high order bits for Pretended Networking payload filtering corresponding to data byte 5.	0x0
31:24 Data_byte_4	RW	Payload Filter 2 high order bits / Payload Mask high order bits for Pretended Networking payload filtering corresponding to data byte 4.	0x0

40.3.47 Wake Up Message Buffer Register for C/S (CAN_WMBn_CS)

40.3.47.1 Offset

Register	Offset
CAN_WMBn_CS	0xB40+n*16 (n=0 - 3)

40.3.47.2 Function

These registers are used to store the Control Status information of an incoming Rx message.

40.3.47.3 Fields

Field	Access	Description	Reset Value
15:0 RSVD_15_0	RO	Reserved	0x0
19:16 DLC	RO	Length of Data in Bytes This 4-bit field is the length (in bytes) of the Rx data received when CAN is in Pretended Networking mode. This 4-bit field is written by the CAN module, copied from the DLC (Data Length Code) field of the received frame. The DLC field indicates which data bytes are valid.	0x0
20 RTR	RO	Remote Transmission Request Bit This bit identifies whether the frame is remote or not. 0b: The frame is data one (not remote). 1b: The frame is a remote one.	0x0
21 IDE	RO	ID Extended Bit This bit identifies whether the frame format is standard or extended. 0b: The frame format is standard. 1b: The frame format is extended.	0x0
22 SRR	RO	Substitute Remote Request This bit can be received either recessive or dominant.	0x0
31:23 RSVD_31_23	RO	Reserved	0x0

40.3.48 Wake Up Message Buffer Register for ID (CAN_WMBn_ID)

40.3.48.1 Offset

Register	Offset
CAN_WMBn_ID	0xB44+n*16 (n=0 - 3)

40.3.48.2 Function

These registers are used to store the ID information of an incoming Rx message.

40.3.48.3 Fields

Field	Access	Description	Reset Value
28:0 ID	RO	Received ID under Pretended Networking mode This register stores either the 29 bits of the extended frame format (considering the ID[28:0] field), or the 11 bits of the standard frame format (considering the ID[28:18] field only. The remaining bits in the ID[17:0] range have no meaning). • The 29 bits of the extended frame format (considering the ID[28:0] field). • The 11 bits of the standard frame format (considering the ID[28:18] field only. The remaining bits in the ID[17:0] range have no meaning).	0x0
31:29 RSVD_31_29	RO	Reserved	0x0

40.3.49 Wake Up Message Buffer Register for Data 0-3 (CAN_WMBn_D03)

40.3.49.1 Offset

Register	Offset
CAN_WMBn_D03	0xB48+n*16 (n=0 - 3)

40.3.49.2 Function

These registers are used to store the data bytes 0 to 3 of the payload information of an incoming Rx message.

40.3.49.3 Fields

Field	Access	Description	Reset Value
7:0 Data_byte_3	RO	The received payload corresponding to data byte 3 under Pretended Networking mode.	0x0
15:8 Data_byte_2	RO	The received payload corresponding to data byte 2 under Pretended Networking mode.	0x0
23:16 Data_byte_1	RO	The received payload corresponding to data byte 1 under Pretended Networking mode.	0x0
31:24 Data_byte_0	RO	The received payload corresponding to data byte 0 under Pretended Networking mode.	0x0

40.3.50 Wake Up Message Buffer Register for Data 4-7 (CAN_WMBn_D47)

40.3.50.1 Offset

Register	Offset
CAN_WMBn_D47	0xB4C+n*16 (n=0 - 3)

40.3.50.2 Function

These registers are used to store the data bytes 4 to 7 of the payload information of an incoming Rx message.

40.3.50.3 Fields

Field	Access	Description	Reset Value
7:0 Data_byte_7	RO	The received payload corresponding to data byte 7 under Pretended Networking mode.	0x0
15:8 Data_byte_6	RO	The received payload corresponding to data byte 6 under Pretended Networking mode.	0x0
23:16 Data_byte_5	RO	The received payload corresponding to data byte 5 under Pretended Networking mode.	0x0
31:24 Data_byte_4	RO	The received payload corresponding to data byte 4 under Pretended Networking mode.	0x0

40.3.51 CAN FD Control Register (CAN_FDCTRL)

40.3.51.1 Offset

Register	Offset
CAN_FDCTRL	0xC00

40.3.51.2 Function

This register contains control bits for the CAN FD operation.

40.3.51.3 Fields

Field	Access	Description	Reset Value
5:0 TDCVAL	RO	Transceiver Delay Compensation Value This register contains the value of the transceiver loop delay measured from the transmitted EDL to R0 transition edge to the respectively received one added to the TDCOFF value specified in the CAN_FDCTRL register. This value is an integer multiple of the Protocol Engine (PE) Clock period (CANCLK).	0x0
7:6 RSVD_7_6	RO	Reserved	0x0
12:8 TDCOFF	RW	Transceiver Delay Compensation Offset This bit field contains the offset value added to the measured transceiver's loop delay to define the position of the delayed comparison point when bit rate switching is active. TDCOFF can be written in the Freeze mode only. Its value can be defined in Protocol Engine (PE) Clock periods (CANCLK), and must be selected to be smaller than the CAN bit duration in the data bit rate for proper operation. NOTE It is not recommended to have TDCOFF equal to zero.	0x1
13 RSVD_13	RO	Reserved	0x0
14 TDCFAIL	W1C	Transceiver Delay Compensation Fail This bit indicates when the Transceiver Delay Compensation (TDC) mechanism is out of range, unable to compensate the transceiver's loop delay and successfully compare the delayed received bits to the transmitted ones. TDCFAIL sets in the first time CAN detects the out of range condition. The CPU needs to write one to clear it. 0b: Measured loop delay is in range. 1b: Measured loop delay is out of range.	0x0
15 TDCEN	RW	Transceiver Delay Compensation Enable This bit can be used to enable and disable the TDC feature. It can be written in the Freeze mode only. NOTE See Transmitter delay compensation in the CAN Protocol standard (ISO 11898-1) for details. NOTE TDC must be disabled when Loop Back mode is enabled (see CAN_CTRL1[LPB]).	0x0

Field	Access	Description	Reset Value
		0b: TDC is disabled. 1b: TDC is enabled.	
17:16 MBDSR0	RW	Message Buffer Data Size for Region 0 This two-bit field selects the data size (8, 16, 32, or 64 bytes) for region R0 of message buffers allocated in RAM. It can be written in the Freeze mode only. 00b: Selects 8 bytes per message buffer. 01b: Selects 16 bytes per message buffer. 10b: Selects 32 bytes per message buffer. 11b: Selects 64 bytes per message buffer	0x0
18 RSVD_18	RO	Reserved	0x0
20:19 MBDSR1	RW	Message Buffer Data Size for Region 1 This two-bit field selects the data size (8, 16, 32, or 64 bytes) for region R1 of message buffers allocated in RAM. It can be written in Freeze mode only. 00b: Selects 8 bytes per message buffer. 01b: Selects 16 bytes per message buffer. 10b: Selects 32 bytes per message buffer. 11b: Selects 64 bytes per message buffer	0x0
21 RSVD_21	RO	Reserved	0x0
23:22 MBDSR2	RW	Message Buffer Data Size for Region 2 This two-bit field selects the data size (8, 16, 32, or 64 bytes) for region R2 of message buffers allocated in RAM. It can be written in Freeze mode only. 00b: Selects 8 bytes per message buffer. 01b: Selects 16 bytes per message buffer. 10b: Selects 32 bytes per message buffer. 11b: Selects 64 bytes per message buffer	0x0
24 RSVD_24	RO	Reserved	0x0
26:25 MBDSR3	RW	Message Buffer Data Size for Region 3 This two-bit field selects the data size (8, 16, 32, or 64 bytes) for region R3 of message buffers allocated in RAM. It can be written in Freeze mode only. 00b: Selects 8 bytes per message buffer. 01b: Selects 16 bytes per message buffer. 10b: Selects 32 bytes per message buffer. 11b: Selects 64 bytes per message buffer	0x0
30:27 RSVD_30_27	RO	Reserved	0x0
31 FD_RATE	RW	Bit Rate Switch Enable This bit enables the Bit Rate Switch (BRS bit) effect during the data phase of Tx messages. The CPU can write this bit any time. However, its effect turns active only when the CAN bus is in Wait for Bus Idle, Bus Idle, or Bus Off state, or when the current frame under reception or transmission reaches the interframe space. By negating CAN_FDCTRL[FD_RATE], the CPU can force all bits in CAN FD messages to be transmitted	0x1

Field	Access	Description	Reset Value
		in nominal bit rate, no matter what the value is in the BRS bit of the Tx MBs. 0b: Transmit a frame in nominal rate. The BRS bit in the Tx MB has no effect. 1b: Transmit a frame with bit rate switching if the BRS bit in the Tx MB is recessive.	

40.3.52 CAN FD Bit Timing Control Register (CAN_FDCBT)

40.3.52.1 Offset

Register	Offset
CAN_FDCBT	0xC04

40.3.52.2 Function

This register stores the CAN bit timing variables used in the data phase of CAN FD messages when the CAN_FDCTRL[FD_RATE] is set.

40.3.52.3 Fields

Field	Access	Description	Reset Value
2:0 FPSEG2	RW	Fast Phase Segment 2 This 3-bit field defines the length of Phase Segment 2 in the data bit rate portion of a CAN FD message with the BRS bit set. This field can be written only in Freeze mode because it is blocked by hardware in other modes. $\text{Phase Segment 2} = (\text{FPSEG2} + 1) \times \text{Time-Quanta}$. Time-Quantum = one Sclock period.	0x0
4:3 RSVD_4_3	RO	Reserved	0x0
7:5 FPSEG1	RW	Fast Phase Segment 1 This 3-bit field defines the length of Phase Segment 1 in the data bit rate portion of a CAN FD message with the BRS bit set. This field can be written only in the Freeze mode because it is blocked by hardware in other modes. $\text{Phase Segment 1} = (\text{FPSEG1} + 1) \times \text{Time-Quanta}$. Time-Quantum = one Sclock period.	0x0
9:8 RSVD_9_8	RO	Reserved	0x0
14:10 FPROPSEG	RW	Fast Propagation Segment This 5-bit field defines the length of the Propagation Segment in the bit time in the data bit rate portion of a CAN FD message with the BRS bit set. This field can be written only in the Freeze mode because it is blocked by hardware in other modes. $\text{Propagation Segment Time} = \text{FPROPSEG} \times \text{Time-Quanta}$. Time-Quantum = one Sclock period.	0x0
15 RSVD_15	RO	Reserved	0x0

Field	Access	Description	Reset Value
18:16 FRJW	RW	Fast Resync Jump Width This 3-bit field defines the maximum number of time quanta that a bit time can be changed by one resynchronization in the data bit rate portion of a CAN FD message with the BRS bit set. One time quantum is equal to the Sclock period. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Resync Jump Width = FSJW + 1.	0x0
19 RSVD_19	RO	Reserved	0x0
29:20 FPRES DIV	RW	Fast Prescaler Division Factor This 10-bit field defines the ratio between the PE clock frequency and the Serial Clock (Sclock) frequency in the data bit rate portion of a CAN FD message with the BRS bit set. The Sclock period defines the time quantum of the CAN FD protocol for the data bit rate. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Sclock frequency = PE clock frequency / (FPRES DIV + 1). NOTE To minimize errors when processing FD frames, use the same value for FPRES DIV and PRES DIV (in CAN_CBT or CAN_CTRL1).	0x0
31:30 RSVD_31_30	RO	Reserved	0x0

40.3.53 CAN FD CRC Register (CAN_FDCRC)

40.3.53.1 Offset

Register	Offset
CAN_FDCRC	0xC08

40.3.53.2 Function

This register provides information about the CRC of transmitted messages.

40.3.53.3 Fields

Field	Access	Description	Reset Value
20:0 FD_TXCRC	RO	Extended Transmitted CRC value This 21-bit field contains the CRC value calculated over the most recent transmitted message. Different CRC polynomials are used for different frame formats. A 15-bit polynomial, CRC_15, is used for all frames in CAN format. The second 17-bit polynomial, CRC_17, is used for frames in CAN FD format with a data field up to sixteen bytes long. The third 21-bit polynomial, CRC_21, is used for frames in CAN FD format with a data field longer than sixteen bytes. For CRC_15 and CRC_17, the 6 most significant bits	0x0

Field	Access	Description	Reset Value
		and the 4 most significant bits are reported as zeros, respectively. For CRC_15, this register has the same content as CRC Register.	
23:21 RSVD_23_21	RO	Reserved	0x0
30:24 FD_MBCRC	RO	CRC Mailbox Number for FD_TXCRC This field indicates the number of the Mailbox corresponding to the value in the FD_TXCRC field, for both FD and non-FD frames. It reports the same information as in the MBCRC bit field in CAN_CRCCR register.	0x0
31 RSVD_31	RO	Reserved	0x0

40.3.54 Message buffer structure

The message buffer structure used by the CAN module is described in Table 40-4. Both Extended (29-bit identifier) and Standard (11-bit identifier) frames used in the CAN specification (Version 2.0 Part B) are described. Each individual MB is formed by 16, 24, 40, or 72 bytes, depending on the quantity of data bytes allocated for the message payload: 8, 16, 32, or 64 data bytes, respectively.

The memory area from 0x80 to 0x47F is used by the mailboxes in CAN0-5. The memory area from 0x80 to 0x87F is used by the mailboxes in CAN6-7. When CAN FD is enabled, the exact address for each MB depends on the size of its payload. See CAN Memory Partition for CAN FD for more detailed information.

Table 40-4 Message buffer structure

MB offset	31	30	29	28	27:24	23	22	21	20	19:16	15:0
0x0	ED L	BR S	ES I		COD E		SR R	ID E	RT R	DLC	TIME STAMP
0x04	PRIO			ID(standard/extended)						ID(extended)	
0x08	data byte0					data byte1				data byte2	data byte3
0x0C	data byte4					data byte5				data byte6	data byte7
0x10	data byte8					data byte9				data byte10	data byte11
0x14	data byte12					data byte13				data byte14	data byte15
0x18	data byte16					data byte17				data byte18	data byte19
0x1c	data byte20					data byte21				data byte22	data byte23
0x20	data byte24					data byte25				data byte26	data byte27
0x24	data byte28					data byte29				data byte30	data byte31
0x28	data byte32					data byte33				data byte34	data byte35
0x2C	data byte36					data byte37				data	data

MB offset	31	30	29	28	27:24	23	22	21	20	19:16	15:0	
											byte38	byte39
0x30	data byte40					data byte41					data byte42	data byte43
0x34	data byte44					data byte45					data byte46	data byte47
0x38	data byte48					data byte49					data byte50	data byte51
0x3C	data byte52					data byte53					data byte54	data byte55
0x40	data byte56					data byte57					data byte58	data byte59
0x44	data byte60					data byte61					data byte62	data byte63

EDL — Extended Data Length

This bit distinguishes between CAN format and CAN FD format frames. The EDL bit must not be set for message buffers configured to ANSWER with code field 0b1010 (See table below).

BRS — Bit Rate Switch

This bit defines whether the bit rate is switched inside a CAN FD format frame.

ESI — Error State Indicator

This bit indicates if the transmitting node is error active or error passive.

CODE — Message Buffer Code

This 4-bit field can be accessed (read or write) by the CPU and by the CAN module itself, as part of the message buffer matching and arbitration process. The encoding is shown in the following table. See Functional description for additional information.

Table 40-5 Message buffer code for Rx buffers

Code description	Rx code before receive new frame	SRV ¹	Rx code after successful reception ²	RRS ³	comment
0b0000: INACTIVE - MB is not active.	inactive	-	-	-	MB does not participate in the matching process.
0b0100: EMPTY - MB is active and empty.	empty	-	full	-	When a frame is received successfully, the CODE field is automatically updated to full.

Code description	Rx code before receive new frame	SRV ¹	Rx code after successful reception ²	RRS ³	comment
4'b0010: full, MB is full	full	yes	full	-	The act of reading the C/S word followed by unlocking the MB (SRV) does not make the code return to empty. It remains full. If a new frame is moved to the MB after the MB was serviced, the code still remains full.
	full	no	overrun	-	If the MB is full and a new frame is moved to this MB before the CPU services it, the CODE field is automatically updated to overrun.
4'b0110: overrun, MB is overwritten into a full MB	overrun	yes	full	-	If the CODE field indicates overrun and CPU has serviced the MB, when a new frame is moved to the MB then the code returns to full.
	overrun	no	overrun	-	If the CODE field already indicates overrun, and another new frame must be moved, the MB will be overwritten again, and the code will remain overrun.
4'b1010: ranswer ⁴ , a frame was configured to recognize a Remote Request Frame and transmit a response Frame in return ⁵ .	ranswer	-	tanswer(4'b1110)	0	A Remote Answer was configured to recognize a remote request frame received. After that, an MB is set to transmit a response frame. The code is automatically changed to tanswer(4'b1110). If CAN_CTRL2[RRS] is negated, transmit a response frame whenever a remote request frame with the same ID is received.
	ranswer	-	-	1	This code is ignored during matching and arbitration process
code[0]=1: busy, CAN is updating the contents of the MB. The CPU must not access the MB.	busy ⁶	-	full	-	Indicates that the MB is being updated. It will be negated automatically and does not interfere with the next CODE.
	busy ⁶	-	overrun	-	

1. SRV: Serviced MB. MB was read and unlocked by reading TIMER or other MB.

2. A frame is considered a successful reception after the frame to be moved to MB (move-in process).
3. Remote Request Stored bit, see "Control 2 Register (CAN_CTRL2)" for details.
4. Code 4'b1010 is not considered Tx and an MB with this code should not be aborted.
5. Code 4'b1010 must be used in Message Buffers configured in CAN FD format, having the EDL bit set.
6. Note that for Tx MBs, the BUSY bit should be ignored upon read, except when AEN bit is set in the MCR register. If this bit is asserted, the corresponding MB does not participate in the matching process.

Table 40-6 Message buffer code for Tx buffers

Code description	Tx code before Tx frame	MB RTR	Tx code after successful transmission	Comment
4'b1000: inactive, mb is not active	inactive	-	-	MB does not participate in arbitration process
4'b1001: abort, mb is aborted	abort	-	-	MB does not participate in arbitration process
4'b1100: data, mb is a Tx Data Frame (MB RTR must be 0)	data	0	inactive	Transmit data frame unconditionally once. After transmission, the MB automatically returns to the inactive state
4'b1100: remote, MB is a Tx Remote Request Frame (MB RTR must be 1)	remote	1	empty	Transmit remote request frame unconditionally once. After transmission, the MB automatically becomes an Rx empty MB with the same ID.
4'b1110: tanswer, MB is a Tx Response Frame from an incoming Remote Request Frame	tanswer	-	ranswer	This is an intermediate code that is automatically written to the MB by the CHI as a result of a match to a remote request frame. The remote response frame will be transmitted unconditionally once, and then the code will automatically return to ranswer (4'b1010). The CPU can also write this code with the same effect. The remote response frame can be either a data frame or another remote request frame depending on the RTR bit value.

40.3.55 CAN MB memory partition

The CAN memory buffers are allocated in memory according to the tables below.

NOTE

For Can0-CAN5 only the memory space with the offset address 0x80-0x47F are available.

Table 40-7 Non-FD 8-byte message buffers

MB address offset (hex)	MBDSR=2'b00 8-byte payload
h0080	MB0
h0090	MB1
h00a0	MB2
h00b0	MB3

MB address offset (hex)	MBDSR=2'b00 8-byte payload
h00c0	MB4
h00d0	MB5
h00e0	MB6
h00f0	MB7
h0100	MB8
h0110	MB9
h0120	MB10
h0130	MB11
h0140	MB12
h0150	MB13
h0160	MB14
h0170	MB15
h0180	MB16
h0190	MB17
h01a0	MB18
h01b0	MB19
h01c0	MB20
h01d0	MB21
h01e0	MB22
h01f0	MB23
h0200	MB24
h0210	MB25
h0220	MB26
h0230	MB27
h0240	MB28
h0250	MB29
h0260	MB30
h0270	MB31
h0280	MB32
h0290	MB33
h02a0	MB34
h02b0	MB35
h02c0	MB36
h02d0	MB37
h02e0	MB38
h02f0	MB39
h0300	MB40
h0310	MB41

MB address offset (hex)	MBDSR=2'b00 8-byte payload
h0320	MB42
h0330	MB43
h0340	MB44
h0350	MB45
h0360	MB46
h0370	MB47
h0380	MB48
h0390	MB49
h03a0	MB50
h03b0	MB51
h03c0	MB52
h03d0	MB53
h03e0	MB54
h03f0	MB55
h0400	MB56
h0410	MB57
h0420	MB58
h0430	MB59
h0440	MB60
h0450	MB61
h0460	MB62
h0470	MB63
h0480	MB64
h0490	MB65
h04a0	MB66
h04b0	MB67
h04c0	MB68
h04d0	MB69
h04e0	MB70
h04f0	MB71
h0500	MB72
h0510	MB73
h0520	MB74
h0530	MB75
h0540	MB76
h0550	MB77
h0560	MB78
h0570	MB79

MB address offset (hex)	MBDSR=2'b00 8-byte payload
h0580	MB80
h0590	MB81
h05a0	MB82
h05b0	MB83
h05c0	MB84
h05d0	MB85
h05e0	MB86
h05f0	MB87
h0600	MB88
h0610	MB89
h0620	MB90
h0630	MB91
h0640	MB92
h0650	MB93
h0660	MB94
h0670	MB95
h0680	MB96
h0690	MB97
h06a0	MB98
h06b0	MB99
h06c0	MB100
h06d0	MB101
h06e0	MB102
h06f0	MB103
h0700	MB104
h0710	MB105
h0720	MB106
h0730	MB107
h0740	MB108
h0750	MB109
h0760	MB110
h0770	MB111
h0780	MB112
h0790	MB113
h07a0	MB114
h07b0	MB115
h07c0	MB116
h07d0	MB117

MB address offset (hex)	MBDSR=2'b00 8-byte payload
h07e0	MB118
h07f0	MB119
h0800	MB120
h0810	MB121
h0820	MB122
h0830	MB123
h0840	MB124
h0850	MB125
h0860	MB126
h0870	MB127

Table 40-8 Non-FD 16-byte message buffer

MB address offset (hex)	MBDSR=2'b01 16-byte payload
h0080	MB0
h0098	MB1
h00b0	MB2
h00c8	MB3
h00e0	MB4
h00f8	MB5
h0110	MB6
h0128	MB7
h0140	MB8
h0158	MB9
h0170	MB10
h0188	MB11
h01a0	MB12
h01b8	MB13
h01d0	MB14
h01e8	MB15
h0200	MB16
h0218	MB17
h0230	MB18
h0248	MB19
h0260	MB20
h0280	MB21
h0298	MB22
h02b0	MB23

MB address offset (hex)	MBDSR=2'b01 16-byte payload
h02c8	MB24
h02e0	MB25
h02f8	MB26
h0310	MB27
h0328	MB28
h0340	MB29
h0358	MB30
h0370	MB31
h0388	MB32
h03a0	MB33
h03b8	MB34
h03d0	MB35
h03e8	MB36
h0400	MB37
h0418	MB38
h0430	MB39
h0448	MB40
h0460	MB41
h0480	MB42
h0498	MB43
h04b0	MB44
h04c8	MB45
h04e0	MB46
h04f8	MB47
h0510	MB48
h0528	MB49
h0540	MB50
h0558	MB51
h0570	MB52
h0588	MB53
h05a0	MB54
h05b8	MB55
h05d0	MB56
h05e8	MB57
h0600	MB58
h0618	MB59
h0630	MB60
h0648	MB61

MB address offset (hex)	MBDSR=2'b01 16-byte payload
h0660	MB62
h0680	MB63
h0698	MB64
h06b0	MB65
h06c8	MB66
h06e0	MB67
h06f8	MB68
h0710	MB69
h0728	MB70
h0740	MB71
h0758	MB72
h0770	MB73
h0788	MB74
h07a0	MB75
h07b8	MB76
h07d0	MB77
h07e8	MB78
h0800	MB79
h0818	MB80
h0830	MB81
h0848	MB82
h0860	MB83

Table 40-9 Non-FD 32-byte message buffer

MB address offset (hex)	MBDSR=2'b10 32-byte payload
h0080	MB0
h00a8	MB1
h00d0	MB2
h00f8	MB3
h0120	MB4
h0148	MB5
h0170	MB6
h0198	MB7
h01c0	MB8
h01e8	MB9
h0210	MB10
h0238	MB11

MB address offset (hex)	MBDSR=2'b10 32-byte payload
h0280	MB12
h02a8	MB13
h02d0	MB14
h02f8	MB15
h0320	MB16
h0348	MB17
h0370	MB18
h0398	MB19
h03c0	MB20
h03e8	MB21
h0410	MB22
h0438	MB23
h0480	MB24
h04a8	MB25
h04d0	MB26
h04f8	MB27
h0520	MB28
h0548	MB29
h0570	MB30
h0598	MB31
h05c0	MB32
h05e8	MB33
h0610	MB34
h0638	MB35
h0680	MB36
h06a8	MB37
h06d0	MB38
h06f8	MB39
h0720	MB40
h0748	MB41
h0770	MB42
h0798	MB43
h07c0	MB44
h07e8	MB45
h0810	MB46
h0838	MB47

Table 40-10 Non-FD 64-byte message buffer

MB address offset (hex)	MBDSR=2'b11 64-byte payload
h0080	MB0
h00c8	MB1
h0110	MB2
h0158	MB3
h01a0	MB4
h01e8	MB5
h0230	MB6
h0280	MB7
h02c8	MB8
h0310	MB9
h0358	MB10
h03a0	MB11
h03e8	MB12
h0430	MB13
h0480	MB14
h04c8	MB15
h0510	MB16
h0558	MB17
h05a0	MB18
h05e8	MB19
h0630	MB20
h0680	MB21
h06c8	MB22
h0710	MB23
h0758	MB24
h07a0	MB25
h07e8	MB26
h0830	MB27

40.3.56 CAN MB memory partition for CAN FD

When CAN FD is enabled, the CAN RAM can be partitioned in blocks of 512 bytes. Each block can accommodate a number of Message Buffers which depends on the configuration provided by CAN_FDCTRL[MBDSRn] bit fields as shown in table below.

NOTE

For CAN0-CAN5, only RAM block 0 and RAM block 1 are available.

Table 40-11 RAM partition

RAM block	Number of MB with 8bytes	Size control bits from CAN_FDCTRL	Number of MBs of different size per block
0	0 to 31	MBDSR0	MBDSR0=00, 32 MBs with 8 bytes payload MBDSR0=01, 21 MBs with 16 bytes payload MBDSR0=10, 12 MBs with 32 bytes payload MBDSR0=11, 7 MBs with 64 bytes payload
1	32 to 63	MBDSR1	MBDSR1=00, 32 MBs with 8 bytes payload MBDSR1=01, 21 MBs with 16 bytes payload MBDSR1=10, 12 MBs with 32 bytes payload MBDSR1=11, 7 MBs with 64 bytes payload
2	64 to 95	MBDSR2	MBDSR2=00, 32 MBs with 8 bytes payload MBDSR2=01, 21 MBs with 16 bytes payload MBDSR2=10, 12 MBs with 32 bytes payload MBDSR2=11, 7 MBs with 64 bytes payload
3	96 to 127	MBDSR3	MBDSR3=00, 32 MBs with 8 bytes payload MBDSR3=01, 21 MBs with 16 bytes payload MBDSR3=10, 12 MBs with 32 bytes payload MBDSR3=11, 7 MBs with 64 bytes payload

When payload sizes of 16, 32, or 64 bytes are configured in some or all RAM blocks 0, 1, 2, and 3, the total number of MBs and its respective number order may be different from the default configuration of 8 bytes. For example, suppose that Block0 is configured to 8 bytes payload, Block1 to 16 bytes, Block2 to 32 bytes and Block3 to 64 bytes, and suppose that CAN_MCR[MAXMB] is configured to 95, then the following table indicates how the Message Buffers will be arranged into RAM.

Table 40-12 RAM partition example

RAM block	Payload size	Num of MBs in the RAM block	Message buffer range
0	CAN_FDCTRL[MBDSR0]=2'b00, 8 bytes payload	32	0 to 31
1	CAN_FDCTRL[MBDSR1]=2'b01, 16 bytes payload	21	32 to 52
2	CAN_FDCTRL[MBDSR2]=2'b10, 32 bytes payload	12	53 to 64
3	CAN_FDCTRL[MBDSR3]=2'b11, 64 bytes payload	7	65 to 71

40.3.57 Legacy Rx FIFO structure

When the CAN_MCR[RFEN] bit is set, the memory area from 0x80 to 0xDC (which is normally occupied by MBs 0–5) is used by the reception Legacy FIFO engine.

The region 0x80-0x8C contains the output of the Legacy Rx FIFO that must be read by the CPU as a message buffer. This output contains the oldest message which has been received but not read yet. The region 0x90-0xDC is reserved for internal use of the Legacy Rx FIFO engine.

An additional memory area, which starts at 0xE0 and may extend up to 0x2DC (normally occupied by MBs 6–37) depending on the CAN_CTRL2[RFFN] field setting, contains the ID filter table (configurable from 8 to 128 table elements) which specifies filtering criteria for accepting frames into the Legacy Rx FIFO.

Out of reset, the ID filter table flexible memory area defaults to 0xE0 and extends only to 0xFC, which corresponds to MBs 6 to 7 for RFFN = 0, for backward compatibility with previous versions of CAN.

Table 40-13 Rx FIFO structure

Offset	31	2928 24	23 22	21	20	19 18	16	15	8	7	0
0x80	IDHIT[31:23]		SRR [22]	IDE [21]	RTR[20]	DLC[19:16]		TIME STAMP[15:0]			
0x84	Reserv ed [31:29]	ID standard[28:18]				ID extended[17:0]					
0x88	data byte0		data byte1					data byte2		data byte 3	
0x8C	data byte4		data byte5					data byte6		data byte 7	
0x90 to 0xDC	reserved										
0xE0	ID filter table element 0										
0xE4	ID filter table element 1										
0xE8 to 0x2D4	ID filter table element 2 to ID filter table element 125										
0x2D8	ID filter table element 126										
0x2DC	ID filter table element 127										

Each ID filter table element occupies an entire 32-bit word and can be compounded by one, two, or four Identifier Acceptance Filters (IDAF) depending on the CAN_MCR[IDAM] field setting. The IDAF indexation is shown in the following figures.

The following table illustrates the three different formats of the ID table elements. Note that all elements of the table must have the same format.

Table 40-14 RX filter ID table structure

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A	R	I	RXIDA (standard = 29-19, extended = 29-1)																													
B	R	I	RXIDB_0 (standard = 29-19, extended = 29-16)										R	I	RXIDB_1 (standard = 13-3, extended = 13-0)																	
C	RXIDC_0 (std/ext = 31-24)				RXIDC_1 (std/ext = 23-16)				RXIDC_2 (std/ext = 15-8)				RXIDC_3 (std/ext = 7-0)																			

- RTR — Remote Frame

- This bit specifies if Remote Frames are accepted into the Legacy FIFO if they match the target ID.
- 1 = Remote Frames can be accepted and data frames are rejected

- 0 = Remote Frames are rejected and data frames can be accepted
- IDE — Extended Frame
 - Specifies whether extended or standard frames are accepted into the Legacy FIFO if they match the target ID.
 - 1 = Extended frames can be accepted and standard frames are rejected
 - 0 = Extended frames are rejected and standard frames can be accepted
- RXIDA — Rx Frame Identifier (Format A)
 - Specifies an ID to be used as acceptance criteria for the Legacy FIFO. In the standard frame format, only the 11 most significant bits (19 to 29) are used for frame identification. In the extended frame format, all bits are used.
- RXIDB_0, RXIDB_1 — Rx Frame Identifier (Format B)
 - Specifies an ID to be used as acceptance criteria for the Legacy FIFO. In the standard frame format, the 11 most significant bits (a full standard ID) (3 to 13 and 19 to 29) are used for frame identification. In the extended frame format, all 14 bits of the field are compared to the 14 most significant bits of the received ID.
- RXIDC_0, RXIDC_1, RXIDC_2, RXIDC_3 — Rx Frame Identifier (Format C)
 - Specifies an ID to be used as acceptance criteria for the Legacy FIFO. In both standard and extended frame formats, all 8 bits of the field are compared to the 8 most significant bits of the received ID.
- IDHIT — Identifier Acceptance Filter Hit Indicator
 - This 9-bit field indicates which Identifier Acceptance Filter was hit by the received message which is in the output of the Legacy Rx FIFO.

Following table shows how the Legacy Rx FIFO filter structure is determined by the value of CAN_CTRL2[RFFN].

Table 40-15 Legacy Rx FIFO filter: possible structures

RFFN[3:0]	Number of Legacy Rx FIFO filter elements	Message Buffers occupied by Legacy Rx FIFO and ID Filter Table	Remaining Available Mailboxes	Legacy Rx FIFO ID Filter Table Elements Affected by Rx Individual Masks	Legacy Rx FIFO ID Filter Table Elements Affected by Legacy Rx FIFO Global Mask
0x0	8	MB0 to 7	MB8 to 127	element0 to 7	none
0x1	16	MB0 to 9	MB10 to 127	element0 to 9	element 10 to 15
0x2	24	MB0 to 11	MB12 to 127	element0 to 11	element 12 to 23
0x3	32	MB0 to 13	MB14 to 127	element0 to 13	element 14 to 31
0x4	40	MB0 to 15	MB16 to 127	element0 to 15	element 16 to 39
0x5	48	MB0 to 17	MB18 to 127	element0 to 17	element 18 to 47
0x6	56	MB0 to 19	MB20 to 127	element0 to 19	element 20 to 55
0x7	64	MB0 to 21	MB22 to 127	element0 to 21	element 22 to 63
0x8	72	MB0 to 23	MB24 to 127	element0 to 23	element 24 to 71
0x9	80	MB0 to 25	MB26 to 127	element0 to 25	element 26 to 79
0xa	88	MB0 to 27	MB28 to 127	element0 to 27	element 28 to 87
0xb	96	MB0 to 29	MB30 to 127	element0 to 29	element 30 to 95
0xc	104	MB0 to 31	MB32 to 127	element0 to 31	element 32 to 103
0xd	112	MB0 to 33	MB34 to 127	element0 to 31	element 32 to 111
0xe	120	MB0 to 35	MB36 to 127	element0 to 31	element 32 to 119

RFFN[3:0]	Number of Legacy Rx FIFO filter elements	Message Buffers occupied by Legacy Rx FIFO and ID Filter Table	Remaining Available Mailboxes	Legacy Rx FIFO ID Filter Table Elements Affected by Rx Individual Masks	Legacy Rx FIFO ID Filter Table Elements Affected by Legacy Rx FIFO Global Mask
0xf	128	MB0 to 37	MB38 to 127	element0 to 31	element 32 to 127

40.4 Function description

The CAN module is a CAN protocol engine with a very flexible mailbox system for transmitting and receiving CAN frames. The mailbox system consists of Message Buffers (MB) that store configuration and control data, time stamp, message ID and data. In addition, the memory corresponding to the first 38 MBs can be configured to support a Legacy FIFO reception scheme with a powerful ID filtering mechanism, capable of checking incoming frames against a table of IDs for up to 128 ID Filter Table elements.

For Classical CAN frames, simultaneous reception through Legacy FIFO and mailbox is supported. For CAN FD frames, reception is supported through mailboxes. For mailbox reception, a matching algorithm makes it possible to store received frames only into MBs that have the same ID programmed on its ID field. A masking scheme makes it possible to match the ID programmed on the MB with a range of IDs on received CAN frames. For transmission, an arbitration algorithm decides the prioritization of MBs to be transmitted based on the message ID (optionally augmented by 3 local priority bits) or the MB ordering.

40.4.1 Transmit process

To transmit a CAN frame, the CPU must prepare a Message Buffer for transmission by running the following procedures:

1. Check if the respective interrupt bit is set and clear it.
2. If the MB is active (meaning the transmission is pending), write the ABORT code (1001b) to the CODE field of the Control and Status word to request an abort of the transmission.
3. Wait for the corresponding IFLAG bit to be asserted by polling the CAN_IFLAG register, or by the interrupt request if enabled by the respective IMASK bit.
4. Read back the CODE field to check whether the transmission was aborted or transmitted.
5. Clear the corresponding interrupt flag.
6. Write the ID register (containing the local priority if enabled via CAN_MCR[LPRIOEN]).
7. Write payload data bytes.
8. Configure the Control and Status word with the desired configuration (MB_CS is CAN_MBn_CS)
 - a) Set ID type via MB_CS[IDE].
 - b) Set Remote Transmission Request (if needed) via MB_CS[RTR].
 - c) If CAN_MCR[FDEN] is enabled, also configure the MB_CS[EDL] and MB_CS[BRS] fields. For details about the relationship between the written value and transmitted value of the MB_CS[ESI] field, see Table 40-18.
 - d) Activate the message buffer to transmit the CAN frame by setting MB_CS[CODE] to 0xC.
 - e) Set Data Length Code in bytes via MB_CS[DLC].

When the MB is activated, it takes part in the arbitration process and is eventually transmitted according to its priority. When the DLC value stored in the MB selected for transmission is greater than the respective MB payload size, CAN adds the necessary number of bytes with constant 0xCC pattern to match the DLC value.

At the end of the successful transmission:

- The value of the Free Running Timer is written into the Time Stamp field.
- The CODE field in the Control and Status word is updated.
- Both CAN_CRCR and CAN_FDCRC registers are updated.
- A status flag is set in the Interrupt Flag register.
- An interrupt is generated if allowed by the corresponding Interrupt Mask Register bit.

When the Abort feature is enabled (CAN_MCR[AEN] is asserted), after the Interrupt Flag is asserted for a Mailbox configured as transmit buffer, the Mailbox is blocked. Therefore, the CPU cannot update it until the CPU negates the Interrupt Flag. This means that the CPU must clear the corresponding IFLAG bit prior to preparing this MB for a new transmission or reception.

40.4.2 Arbitration process

The arbitration process scans the Mailboxes to search for the Tx one that holds the message to be sent in the next opportunity. This Mailbox is called the arbitration winner.

The scan starts from the lowest number Mailbox and runs toward the higher ones.

The arbitration process is triggered in the following events:

- From the CRC field of the CAN frame. The start point depends on the CAN_CTRL2[TASD] field value.
- During the Error Delimiter field of a CAN frame.
- During the Overload Delimiter field of a CAN frame.
- When the winner is inactivated and the CAN bus has still not reached the first bit of the Intermission field.
- When there is CPU write to the C/S word of a winner MB and the CAN bus has still not reached the first bit of the Intermission field.
- When CHI is in Idle state and the CPU writes to the C/S word of any MB.
- When CAN exits Bus Off state.
- Upon leaving Freeze mode or Low Power mode.

If the arbitration process does not manage to evaluate all Mailboxes before the CAN bus has reached the first bit of the intermission field, the temporary arbitration winner is invalidated and the CAN will not compete for the CAN bus in the next opportunity. The arbitration process selects the winner among the active Tx Mailboxes at the end of the scan according to the settings of both CAN_CTRL1[LBUF] and CAN_MCR[LPRIOEN] bits.

If CAN_CTRL1[LBUF] bit is asserted, the first (lowest number) active Tx Mailbox found is the arbitration winner. CAN_MCR[LPRIOEN] bit has no effect when CAN_CTRL1[LBUF] is asserted.

If CAN_CTRL1[LBUF] bit is negated, then the arbitration process searches the active Tx Mailbox with the highest priority. This means that this Mailbox's frame would have a higher probability in winning the arbitration on the CAN bus when multiple external nodes compete for the bus at the same time.

The sequence of bits considered for this arbitration is called the arbitration value of the Mailbox. Thus, the highest-priority Tx Mailbox is the one that has the lowest arbitration value among all Tx Mailboxes.

If two or more Mailboxes have equivalent arbitration values, the Mailbox with the lowest number is the arbitration winner. The composition of the arbitration value depends on the setting of CAN_MCR[LPRIOEN] bit.

40.4.3 Arbitration process (continued)

After the arbitration winner is found, its content is copied to a hidden auxiliary MB called Tx

Serial Message Buffer (Tx SMB), which has the same structure as a normal MB but is not user accessible. This operation is called move-out and after it is done, write access to the C/S word of the corresponding MB is blocked (if the AEN bit in CAN_MCR register is asserted). The write access is restored in the following events:

- After the MB is transmitted and the corresponding IFLAG bit is cleared by the CPU
- CAN enters in Freeze mode or Bus Off
- CAN loses the bus arbitration or there is an error during the transmission

At the first opportunity window on the CAN bus, the message on the Tx SMB is transmitted according to the CAN protocol rules.

Arbitration process can be triggered in the following situations:

- During Rx and Tx frames from CAN CRC field to end of the frame. CAN_CTRL2[TASD] bit value may be changed to optimize the arbitration start point.
- During CAN Bus Off state from TX_ERR_CNT=124 to 128. CAN_CTRL2[TASD] bit value may be changed to optimize the arbitration start point.
- During C/S write by CPU in Bus Idle. First C/S write starts the arbitration process and a second C/S write during this same arbitration restarts the process. If other C/S writes are performed, Tx arbitration process is pending. If there is no arbitration winner after the arbitration process has finished, then the TX arbitration machine begins a new arbitration process. If there is a pending arbitration and Bus Idle state starts, an arbitration process is triggered. In this case, the first and second C/S write in Bus Idle will not restart the arbitration process. It is possible that there is not enough time to finish arbitration in Wait For Bus Idle state and the next state is Idle. In this case, the scan is not interrupted, and it is completed during Bus Idle state. During this arbitration, C/S write does not cause arbitration restart.
- Arbitration winner deactivation during a valid arbitration window.
- Upon exiting Freeze mode (first bit of the WaitForBusIdle state). Or if there is a re-synchronization during WaitForBusIdle, the arbitration process is restarted.

Arbitration process stops in the following situations:

- All Mailboxes were scanned.
- A Tx active Mailbox is found in case of Lowest Buffer feature enabled.
- Arbitration winner inactivation or abort during any arbitration process.
- There was not enough time to finish Tx arbitration process (for example, when a deactivation was performed near the end of the frame). In this case, arbitration process is pending.
- Error or Overload flag in the bus.
- Low Power or Freeze mode request in Idle state.

Arbitration is considered pending as described below:

- It was not possible to finish arbitration process in time.
- C/S write during arbitration if write is performed in a MB of which number is lower than the Tx arbitration pointer.
- Any C/S write if there is no Tx Arbitration process in progress.
- Rx Match has just updated an Rx Code to Tx Code.
- Entering Bus off state.

C/S write during arbitration has the following effect:

- If C/S write is performed in the arbitration winner, a new process is restarted immediately.
- If C/S write is performed in an MB with a number higher than the Tx arbitration pointer, the ongoing arbitration process will scan this MB as normal.

40.4.4 Receive process

To be able to receive CAN frames into a Mailbox, the CPU must prepare it for reception by executing the following steps:

1. If the Mailbox is active (either Tx or Rx) inactivate the Mailbox
2. Write the ID word
3. Write the EMPTY code (0100b) to the CODE field of the Control and Status word to activate the Mailbox. No setup is required for EDL, BRS and ESI bits. They are overwritten by the respective bit fields in the received message.

After the MB is activated, it will be able to receive frames that match the programmed filter. At the end of a successful reception, the Mailbox is updated by the move-in process as follows:

1. Store the received Data field (8 bytes at most for Classical CAN message format and up to 64 bytes for CAN FD message format).
2. Store the received Identifier field.
3. Write the value of the Free Running Timer at the time of the second bit of frame's Identifier field into the Mailbox's Time Stamp field.
4. Store the received SRR, IDE, RTR, EDL, BRS, ESI and DLC fields.
5. Update the CODE field in the Control and Status word.
6. Set a status flag in the Interrupt Flag Register and generate an interrupt if allowed by the corresponding Interrupt Mask Register bit.

The recommended way for CPU to service (read) the frame received in a Mailbox is to use the following procedures:

1. Read the Control and Status word of that Mailbox.
2. Check if the BUSY bit is deasserted, which indicates that the Mailbox is locked. Repeat step 1 while it is asserted.
3. Read the contents of the Mailbox. Once Mailbox is locked now, its contents will not be modified by CAN Move-in processes.
4. Acknowledge the proper flag at IFLAG registers.
5. Read the Free Running Timer. It is optional but recommended to unlock Mailbox as soon as possible and make it available for reception.

The CPU should poll for frame reception by the status flag bit for the specific Mailbox in one of the IFLAG Registers and not by the CODE field of that Mailbox. The reason why polling the CODE field does not work is that once a frame was received and the CPU services the Mailbox (by reading the C/S word followed by unlocking the Mailbox), the CODE field will not return to EMPTY. It will remain FULL. If the CPU tries to work around this behavior by writing to the C/S word to force an EMPTY code after reading the Mailbox without a prior safe inactivation, a newly received frame matching the filter of that Mailbox may be lost.

Note that the received frame's Identifier field is always stored in the matching Mailbox. The contents of the ID field in a Mailbox may change if the match was due to masking. When CAN_MCR[SRXDIS] bit is asserted, CAN will not store frames transmitted by itself in any MB, even if it contains a matching Rx Mailbox, and no interrupt flag or interrupt signal will be generated. Otherwise, when CAN_MCR[SRXDIS] bit is deasserted, CAN can receive frames transmitted by itself if a matching Rx Mailbox exists.

To receive CAN frames through the Legacy Rx FIFO, the CPU must enable and configure the Legacy Rx FIFO during Freeze mode. In addition, upon reception of the Frames Available in Legacy Rx FIFO interrupt, the CPU should service the received frame by using the following procedures:

Read the Control and Status word (optional: needed only if a mask was used for IDE and RTR bits)

1. Read the ID field (optional: needed only if a mask was used)

2. Read the Data field
3. Read the CAN_RXFIR register (optional)
4. Clear the Frames Available in Legacy Rx FIFO interrupt by writing 1 to CAN_IFLAG1[BUF5I] bit (mandatory: releases the MB and allows the CPU to read the next Rx FIFO entry)

When CAN_MCR[DMAE] is asserted, upon reception of a frame in Legacy FIFO, CAN_IFLAG1[BUF5I] generates a DMA request and does not generate a CPU interrupt. The CAN_IMASK1 bits in Legacy Rx FIFO region are not used.

The DMA controller must service the received frame by using the following procedures:

1. Read the Control and Status word (read 0x80 address, optional)
2. Read the ID field (read 0x84 address, optional)
3. Read all Data Bytes (start read at 0x88 address, optional)
4. Read the last Data Bytes (read 0x8C address is mandatory)

40.4.5 Match process

The matching process scans the MB memory looking for Rx MBs programmed with the same ID as the one received from the CAN bus. If the Legacy Rx FIFO is enabled, the priority of scanning can be selected between mailboxes and FIFO filters. The matching starts from the lowest number Message Buffer toward the higher ones. If no match is found within the first structure, then the other is scanned subsequently. When the FIFO is full, the matching algorithm always looks for a matching MB outside the FIFO region.

As the frame is being received, it is stored in a hidden auxiliary MB called Rx Serial Message Buffer (Rx SMB).

The matching process start point depends on the following conditions:

- If the received frame is a remote frame, the start point is the CRC field of the frame
- If the received frame is a data frame with a DLC field that is equal to zero, the start point is the CRC field of the frame
- If the received frame is a data frame with a DLC field differing from zero, the start point is the DATA field of the frame

If a matching ID is found in the FIFO table or in one of the mailboxes, the contents of the Rx SMB are transferred to the FIFO or to the matched mailbox by the move-in process. However, if any CAN protocol error is detected, then no match results are transferred to the FIFO or to the matched mailbox at the end of reception.

The matching process scans all matching elements of both Rx FIFO (if enabled) and the active Rx Mailboxes (CODE is EMPTY, FULL, OVERRUN or RANSWER). The matching process searches for a successful comparison with the matching elements of the Rx SMB that is receiving the frame on the CAN bus. The Rx SMB has the same structure as a mailbox. The reception structures (Rx FIFO or Mailboxes) associated with the matching elements that had a successful comparison are the matched structures. The matching winner is selected at the end of the scan among those matched structures and depends on conditions described above. See the details in the following table.

1. For mailbox structure, if SMB[IDE] is asserted, the ID is 29 bits (ID Standard + ID Extended). If SMB[IDE] is negated, the ID is only 11 bits (ID Standard). For Legacy Rx FIFO structure, the ID depends on IDAM.
2. cmp: Compares the Rx SMB contents with the MB contents regardless of the masks.
3. no_cmp: The Rx SMB contents are not compared with the MB contents.
4. cmp_msk: Compares the Rx SMB contents with MB contents considering the masks.
5. SMB[IDE] and SMB[RTR] are not considered when IDAM is type C.

Table 40-16 Matching architecture

Structure	SMB [RTR]	CTRL2 [RRS]	CTRL2 [EACEN]	MB [IDE]	MB [RTR]	MB [ID1]	MB [CODE]
Mailbox	0	-	0	cmp2	no_cmp3	Cmp_msk4	EMPTY or FULL or OVERRUN
Mailbox	0	-	1	cmp_msk	Cmp_msk	Cmp_msk	EMPTY or FULL or OVERRUN
Mailbox	1	0	-	cmp	No_cmp	cmp	RANSWER
Mailbox	1	1	0	cmp	No_cmp	Cmp_msk	EMPTY or FULL or OVERRUN
Mailbox	1	1	1	cmp_msk	Cmp_msk	Cmp_msk	EMPTY or FULL or OVERRUN
Legacy FIFO	-	-	-	cmp_msk	Cmp_msk	Cmp_msk	-

A reception structure is free-to-receive when any of the following conditions is satisfied:

- The CODE field of the mailbox is EMPTY
- The CODE field of the mailbox is either FULL or OVERRUN and it has already been serviced
- The CODE field of the Mailbox is either FULL or OVERRUN and an inactivation is performed
- The Legacy Rx FIFO is not full

The scan order for mailboxes and Legacy Rx FIFO is from the matching element with lowest number to the higher ones. The matching winner search for mailboxes is affected by the CAN_MCR[IRMQ] bit. If it is negated, the matching winner is the first matched mailbox regardless of whether it is free-to-receive or not. If it is asserted, the matching winner is selected according to the priority below:

1. The first free-to-receive matched mailbox;
2. The last non free-to-receive matched mailbox.

It is possible to select the priority of scan between mailboxes and Legacy Rx FIFO by the CAN_CTRL2[MRP] bit.

If the selected priority is Rx FIFO first:

- If the Rx FIFO is a matched structure and is free-to-receive, then the Rx FIFO is the matching winner regardless of the scan for mailboxes
- Otherwise (the Rx FIFO is not a matched structure or is not free-to-receive), then the matching winner is searched among mailboxes as described above

If the selected priority is mailboxes first:

- If a free-to-receive matched mailbox is found, it is the matching winner regardless of the scan for Rx FIFO
- If no matched mailbox is found, then the matching winner is searched in the scan for the Rx FIFO
- If both conditions above are not satisfied and a non free-to-receive matched mailbox is found, then the matching winner determination is conditioned by the CAN_MCR[IRMQ] bit:
 - If CAN_MCR[IRMQ] bit is negated, the matching winner is the first matched mailbox
 - If CAN_MCR[IRMQ] bit is asserted, the matching winner is the Rx FIFO if it is a free-to-receive matched structure; otherwise, the matching winner is the last non free-to-receive matched mailbox.

For example, if the FIFO is disabled, IRMQ is enabled and there are two MBs with the same ID, and CAN starts receiving messages with that ID.

Assume that these MBs are the second and the fifth in the array.

- When the first message arrives, the matching algorithm finds the first match in MB number 2. The code of this MB is EMPTY, so the message is stored there.
- When the second message arrives, the matching algorithm finds MB number 2 again, but it is not "free-to-receive", so it keeps looking, finds MB number 5, and stores the message there. If yet another message with the same ID arrives, the matching algorithm finds out that there are no matching MBs that are "free-to-receive", so it decides to overwrite the last matched MB, which is number 5. In doing so, it sets the CODE field of the MB to indicate OVERRUN.
- The ability to match the same ID in more than one MB can be exploited to implement a reception queue (in addition to the full featured FIFO) to allow more time for the CPU to service the MBs.
- By programming more than one MB with the same ID, received messages are queued into the MBs. The CPU can examine the Time Stamp field of the MBs to determine the order in which the messages arrived.
- Matching to a range of IDs is possible by using ID Acceptance Masks. CAN supports individual masking per MB.

See the description of the Rx Individual Mask Registers (CAN_RXIMRn). During the matching algorithm, if a mask bit is asserted, then the corresponding ID bit is compared. If the mask bit is negated, the corresponding ID bit is a "do not care" bit.

NOTE

The Individual Mask Registers are implemented in RAM, so they are not initialized out of reset. Besides, they can only be programmed while the module is in Freeze mode. Otherwise, they are blocked by hardware.

- CAN also supports an alternate masking scheme with only four mask registers (CAN_RXFGMASK, CAN_RXMGMSK, CAN_RX14MASK and CAN_RX15MASK) for backwards compatibility with legacy applications.
- This alternate masking scheme is enabled when the IRMQ bit in the CAN_MCR Register is negated.

40.4.6 Move process

40.4.6.1 Move in

The move-in process is the copy of a message received by an Rx SMB to an Rx mailbox or FIFO that has matched it. If the move destination is the Legacy Rx FIFO, attributes of the message are also copied to the CAN_RXFIR FIFO. Each Rx SMB has its own move-in process, but only one is performed at a given time as described above. The move-in starts only when the message held by the Rx SMB has a corresponding matching winner and all of the following conditions are true:

- The CAN bus has reached or past either:
- The second bit of Intermission field next to the frame that carried the message that is in the Rx SMB
- The first bit of an overload frame next to the frame that carried the message that is in the Rx SMB
- There is no ongoing matching process
- The destination mailbox is not locked by the CPU
- There is no ongoing move-in process from another Rx SMB. If more than one move-in process is to be started simultaneously, both are performed and the newest substitutes the oldest

The term pending move-in is used throughout the documentation and stands for a move-to-be that still does not satisfy all the conditions mentioned above.

The move-in is canceled and the Rx SMB can receive another message if any of the following conditions is satisfied:

- The destination mailbox is inactivated after the CAN bus has reached the first bit of Intermission field next to the frame that carried the message and its matching process has finished
- There is a previous pending move-in to the same destination mailbox
- The Rx SMB is receiving a frame transmitted by the CAN itself and the self-reception is disabled (CAN_MCR[SRXDIS] bit is asserted)
- Any CAN protocol error is detected

NOTE

The pending move-in is not canceled if the module enters the Freeze or Low-Power mode. It only stays on hold waiting for exiting the Freeze and Low-Power mode and unlocking the module. If an MB is unlocked during Freeze mode, the move-in occurs immediately.

Perform the following steps to run the move-in process by the CAN:

- Push IDHIT into the RXFIR FIFO if the message is destined to the Legacy Rx FIFO.
- Read all data words from the Rx SMB according to the selected payload size for the Rx storage element.
- Write all data words to the Rx mailbox according to the selected payload size for the Rx storage element. If the data size of the storage element is smaller than the original payload size described in the message's DLC field, the payload is truncated and the high order bytes that do not fit the destination size are lost.
- Read the Control/Status and ID words from the Rx SMB.
- Write Control/Status and ID words to the Rx mailbox, and update the CODE field.

The move-in process is not atomic. In such a way, the inactivation of the destination Mailbox immediately cancels it. In this case, the mailbox may be partially updated, thus incoherent. The exception is if the move-in destination is a Legacy Rx FIFO Message Buffer, then the process cannot be canceled.

The BUSY Bit (least significant bit of the CODE field) of the destination Message Buffer is asserted while the move-in is being performed to alert the CPU that the Message Buffer content is temporarily incoherent.

40.4.6.2 Move out

The move-out process is the copy of the content from a Tx Mailbox to the Tx SMB when a message for transmission is available. See more details in the "Arbitration process" section. The move-out occurs when the following conditions are met:

- The first bit of Intermission field
- During the Bus Off state when TX Error Counter is in the 124 to 128 range
- During the Bus Idle state
- During the Wait For Bus Idle state

The move-out process is not atomic. Only the CPU has priority to access the memory concurrently out of Bus Idle state. In Bus Idle, the move-out has the lowest priority to the concurrent memory accesses.

40.4.7 Data coherence

40.4.7.1 Transmission abort mechanism

The abort mechanism provides a safe way to request the abortion of a pending transmission. A feedback mechanism is provided to inform the CPU if the transmission was aborted or if the frame could not be aborted and was transmitted instead.

Two primary conditions must be fulfilled to abort a transmission:

- CAN_MCR[AEN] bit must be asserted

- The first CPU action must be the writing of abort code (1001b) into the CODE field of the Control and Status word.

Active MBs configured for transmission must be aborted first before they can be updated. Suppose the abort code is written to a mailbox which is currently being transmitted or to a mailbox that was already loaded into the Tx SMB for transmission, in this case, the write operation is blocked and the transmission is not disturbed. However, the abort request is captured and kept pending until one of the following conditions is satisfied:

- The module loses the bus arbitration
- There is an error during the transmission
- The module is put into the Freeze mode
- The module enters the Bus Off state
- There is an overload frame

If none of the conditions above are met, the MB is transmitted correctly, the interrupt flag is set in the IFLAG register, and an interrupt to the CPU is generated (if enabled). The abort request is automatically cleared when the interrupt flag is set. On the other hand, if one of the above conditions is met, the frame is not transmitted. Therefore, the abort code is written into the CODE field, the interrupt flag is set in the IFLAG, and an interrupt is (optionally) generated to the CPU.

- If the CPU writes the abort code before the transmission begins internally, then the write operation is not blocked; therefore, the MB is updated and the interrupt flag is set. In this way, the CPU just needs to read the abort code to ensure the active MB was safely inactivated.
- Although the AEN bit is asserted and the CPU wrote the abort code, the MB is inactivated and not aborted, because the transmission did not start yet.
- One mailbox is aborted when the abort request is captured and kept pending until one of the previous conditions is satisfied.

40.4.7.2 Mailbox inactivation

Inactivation is a mechanism provided to protect the Mailbox against updates by the CAN internal processes, allowing the CPU to rely on Mailbox data coherence after updating it, even in Normal mode.

Inactivation of transmission mailboxes must be performed just when MCR[AEN] bit is deasserted.

If a mailbox is inactivated, it takes part in neither the arbitration process nor the matching process until it is reactivated.

To inactivate a mailbox, the CPU must update its CODE field to INACTIVE (either 0000b or 1000b).

Because the user cannot synchronize the CODE field update with the CAN internal processes, an inactivation can have the following consequences:

- A frame in the bus that matches the filtering of the inactivated Rx mailbox may be lost without notice, even if there are other mailboxes with the same filter
- A frame containing the message within the inactivated Tx mailbox may be transmitted without setting the respective IFLAG.

NOTE

Message Buffers that are part of the Legacy Rx FIFO cannot be inactivated.

There is no write protection on the Legacy FIFO region by CAN. Therefore, CPU must maintain data coherency in the Legacy FIFO region when RFEN is asserted.

40.4.7.3 Mailbox lock mechanism

In addition to mailbox inactivation, CAN has another data coherence mechanism for the receiving process. When the CPU reads the Control and Status word of an Rx MB with codes FULL or OVERRUN, CAN assumes that the CPU wants to read the whole MB in an atomic operation, and therefore it sets an internal lock flag for that MB. The lock is released when

the CPU reads the Free Running Timer (global unlock operation), or when it reads the Control and Status word of another MB regardless of its code.

A CPU write into the C/S word also unlocks the MB. However, this procedure is not recommended for the normal unlock use because it cancels a pending-move and potentially may lose a received message. The MB locking prevents a new frame from being written into the MB while the CPU is reading it.

Note that the locking mechanism applies only to Rx MBs that are not part of the Legacy Rx FIFO and have a code different than INACTIVE (0000b) or EMPTY[1] (0100b). In addition, Tx MBs cannot be locked.

- For example, suppose that the Legacy Rx FIFO is disabled and the second and the fifth MBs of the array are programmed with the same ID, and CAN has already received and stored messages into these two MBs.
- Suppose now that the CPU decides to read MB number 5 and at the same time another message with the same ID is arriving. When the CPU reads the Control and Status word of MB number 5, this MB is locked. The new message arrives and the matching algorithm finds out that there are no "free-to-receive" MBs, so it decides to override MB number 5. However, this MB is locked, so the new message cannot be written there. It will remain in the Rx SMB waiting for the MB to be unlocked, and only then will be written to the MB.

Suppose that the MB is not unlocked in time, and yet another new message with the same ID arrives, in that case, the new message overwrites the one on the Rx SMB, and there will be no indication of lost messages either in the CODE field of the MB or in the Error and Status Register.

- While the message is being moved-in from the Rx SMB to the MB, the BUSY bit on the CODE field is asserted. If the CPU reads the Control and Status word and finds out that the BUSY bit is set, it should defer accessing the MB until the BUSY bit is negated.
- Inactivation takes precedence over locking. If the CPU inactivates a locked Rx MB, then its lock status is negated and the MB is marked as invalid for the current matching round. Any pending message on the Rx SMB will not be transferred anymore to the MB. An MB is unlocked when the CPU reads the Free Running Timer Register (see the section "Free Running Timer Register (CAN_TIMER)" for more details), or the C/S word of another MB. Lock and unlock mechanisms have the same functionality in both the Normal and Freeze modes.

40.4.8 Legacy RX FIFO

The Legacy Rx FIFO is receive-only and is enabled by asserting the CAN_MCR[RFEN] bit. The reset value of this bit is zero to maintain software backward compatibility with previous versions of the module that did not have the Legacy FIFO feature.

- The Legacy FIFO is 6-message deep. The CPU can read the received messages sequentially, in the order they were received, by repeatedly reading a Message Buffer structure at the output of the Legacy FIFO.
- The CAN_IFLAG1[BUF5I] (Frames available in Legacy Rx FIFO) is asserted when at least one frame is available to be read from the Legacy FIFO. An interrupt is generated if it is enabled by the corresponding mask bit. Upon reception of the interrupt, the CPU can read the message (accessing the output of the Legacy FIFO as a Message Buffer) and the CAN_RXFIR register and then clear the interrupt. If there are more messages in the Legacy FIFO, the act of clearing the interrupt updates the output of the Legacy FIFO with the next message and updates the CAN_RXFIR with the attributes of that message, reissuing the interrupt to the CPU. Otherwise, the flag remains negated. The output of the Legacy FIFO is only valid whilst the CAN_IFLAG1[BUF5I] is asserted.
- The CAN_IFLAG1[BUF6I] (Legacy Rx FIFO Warning) is asserted when the number of unread messages within the Legacy Rx FIFO is increased to 5 from 4 due to the reception of a new one, which means that the Legacy Rx FIFO is almost full. The flag remains asserted until the CPU clears it.
- The CAN_IFLAG1[BUF7I] (Legacy Rx FIFO Overflow) is asserted when an incoming

message was lost for the Legacy Rx FIFO is full. Note that the flag will not be asserted when the Legacy Rx FIFO is full and the message was captured by a mailbox. The flag remains asserted until the CPU clears it.

- Clearing one of those three flags does not affect the state of the other two.
- An interrupt is generated if an IFLAG bit is asserted and the corresponding mask bit is asserted too.

NOTE

Legacy Rx FIFO must not be enabled when CAN FD feature is enabled.

A powerful filtering scheme is provided to accept only frames intended for the target application, reducing the interrupt servicing work load. The filtering criteria is specified by programming a table of up to 128 32-bit registers, according to CAN_CTRL2[RFFN] setting, that can be configured to one of the following formats:

- Format A: 128 IDAFs (extended or standard IDs including IDE and RTR)
- Format B: 256 IDAFs (standard IDs or extended 14-bit ID slices including IDE and RTR)
- Format C: 512 IDAFs (standard or extended 8-bit ID slices)

NOTE

A chosen format is applied to all entries of the filter table. It is not possible to mix formats within the table.

Every frame available in the Legacy Rx FIFO has a corresponding IDHIT (Identifier Acceptance Filter Hit Indicator) that can be read in the IDHIT field from C/S word, as shown in the Legacy Rx FIFO Structure description. Another way in which the CPU can obtain this information is by accessing the CAN_RXFIR register.

The CAN_RXFIR[IDHIT] field refers to the message at the output of the Legacy FIFO and is valid while the CAN_IFLAG1[BUF5I] flag is asserted. Therefore, the CAN_RXFIR register must be read only prior to clearing the flag, which ensures that the information refers to the correct frame within the Legacy FIFO.

Up to 32 elements of the filter table are individually affected by the Individual Mask Registers (CAN_RXIMRn), according to the setting of CAN_CTRL2[RFFN], allowing very powerful filtering criteria to be defined. If the CAN_MCR[IRMQ] bit is negated, then the Legacy Rx FIFO filter table is affected by CAN_RXFGMASK.

40.4.8.1 Legacy RX FIFO under DMA operation

The receive-only Legacy FIFO can support DMA. This feature is enabled by asserting both the CAN_MCR[RFEN] and CAN_MCR[DMA] bits. The reset value of CAN_MCR[DMA] bit is zero to maintain backward compatibility with previous module versions that did not have the DMA feature.

The DMA controller can read the received message by reading a Message Buffer structure at the Legacy FIFO output port at the 0x80-0x8C address range.

When CAN_MCR[DMAE] is asserted, the CPU must not access the Legacy FIFO output port address range. Prior to enabling the CAN_MCR[DMAE], the CPU must service the IFLAGs asserted in the Legacy Rx FIFO region. Otherwise, these IFLAGs may show that the FIFO has data to be serviced, and generate a DMA request in a mistaken way. Before disabling the CAN_MCR[DMAE], the CPU must perform a clear Legacy FIFO operation.

The CAN_IFLAG1[BUF5I] (Frames available in Legacy Rx FIFO) is asserted when there is at least one frame that is available to be read from the FIFO. Consequently, a DMA request is generated simultaneously. Upon reception of the request, the DMA controller can read the message (accessing the output of the Legacy FIFO as a Message Buffer). The DMA reading process must end by reading address 0x8C, which clears the CAN_IFLAG1[BUF5I] and updates both the FIFO output with the next message (if FIFO is not empty) and the CAN_RXFIR register with the attributes of the new message. If more messages are stored in the FIFO, the CAN_IFLAG1[BUF5I] will be re-asserted, and another DMA request is issued. Otherwise, the flag remains negated.

NOTE

CAN_RXFIR register contents cannot be read after DMA completes the Legacy FIFO read. The

IDHIT information is also available in the C/S word at address 0x080.

The CAN_IFLAG1[BUF6I] and CAN_IFLAG1[BUF7I] are not used when the DMA feature is enabled.

When CAN is working with DMA, the CPU does not receive any Legacy Rx FIFO interruption and must not clear the related IFLAGs. Besides, the related IMASKs are not used to mask the generation of DMA requests.

40.4.8.2 Clear Legacy RX FIFO operation

When CAN_MCR[RFEN] is asserted, the clear Legacy FIFO operation is a feature that is used to empty Legacy FIFO contents.

With CAN_MCR[RFEN] asserted, the Clear FIFO occurs when the CPU writes 1 in CAN_IFLAG1[BUF0I]. This operation can only be performed in Freeze Mode and is blocked by hardware in other modes. This operation does not clear the FIFO IFLAGs. Consequently, the CPU must service all FIFO IFLAGs before the clear FIFO task is executed.

When Legacy Rx FIFO is working with DMA, the clear FIFO operation clears the CAN_IFLAG1[BUF5I] and the DMA request is canceled.

NOTE

Clear Legacy FIFO operation does not clear IFLAGs, except when CAN_MCR[DMAE] is asserted. In this case, only the CAN_IFLAG1[BUF5I] is cleared.

40.4.9 CAN protocol related features

40.4.9.1 CAN FD ISO compliance

The CAN FD protocol has been improved to increase the failure detection capability in the original CAN FD protocol, which is also called non-ISO CAN FD by CAN in Automation (CiA). A three-bit stuff counter and a parity bit have been introduced in the improved CAN FD protocol, now called ISO CAN FD. The CRC calculation has also been modified. All these improvements make the ISO CAN FD protocol incompatible with the non-ISO CAN FD protocol. The non-ISO CAN FD is still supported by CAN so that it can be used mainly during an intermediate phase, for evaluation and development purposes.

It is recommended to configure CAN to the ISO CAN FD protocol by setting the ISOCANFDEN field in the CAN_CTRL2 register.

40.4.9.2 CAN FD frames

The ISO 11898-1 standard specifies the Classical Frame format compliant to ISO 11898-1 (2003) and introduces the CAN Flexible Data Rate Frame format. The Classical Frame format allows bit rates up to 1 Mbit/s and payloads up to 8 bytes per frame. The Flexible Data Rate Frame format allows bit rates to be higher than 1 Mbit/s and payloads to be longer than 8 bytes per frame. CAN can receive and transmit CAN FD messages interleaved with Classical CAN messages.

There are three additional control bits in the CAN FD frame. The Extended Data Length (EDL) bit enables a longer data payload with different data length coding. The Bit Rate Switch (BRS) bit decides if the bit rate is switched inside a CAN FD format frame. The Error State Indicator (ESI) flag is transmitted dominant by error active nodes, and recessive by error passive nodes. There are no Remote Frames in the CAN FD format. A message configured to transmit a Remote Frame is always sent out in the Classical CAN format. When an FD frame is received and matches a mailbox, the RTR bit in the receiving message buffer is negated. The RTR bit must be taken into account in classical frames only.

CAN FD messages may be formatted as long frames where the data field exceeds 8 bytes, and may range from 12 up to 64 bytes. They can also be configured to support bit rate switching, where the control field, the data field, and the CRC field of a CAN frame are transmitted with a higher bit rate than the beginning and the end of the frame. Messages in

Classical CAN format are limited to transport a maximum payload of 8 bytes at a nominal rate. The following figure shows the message formats for Classical and FD frames with either standard or extended ID.

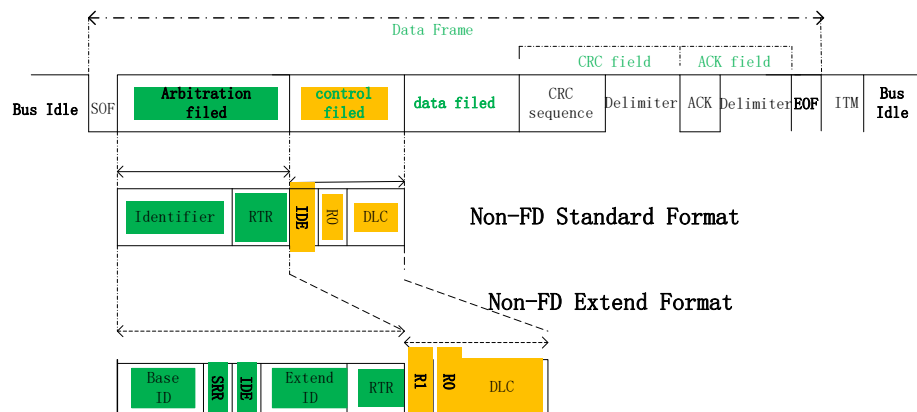


Figure 40-2 CAN Non-FD frame structure

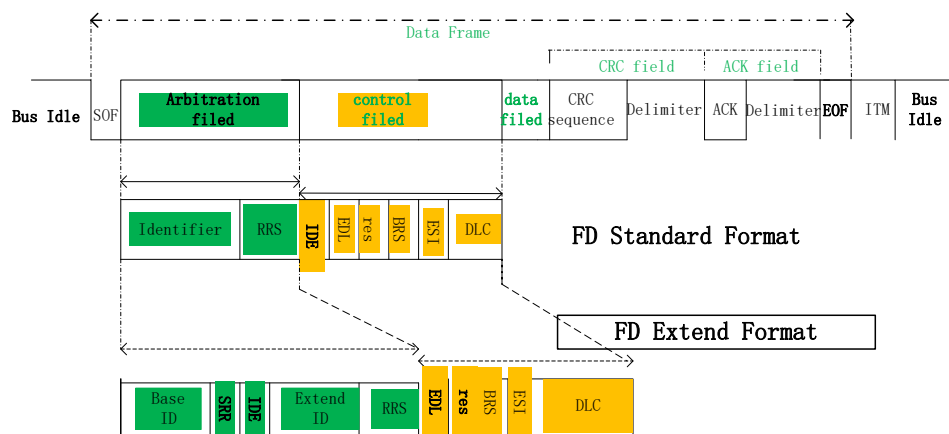


Figure 40-3 CAN FD frame structure

Table 40-17 Naming Convention

Name	Descriptions
SOF	start of frame
RTR	remote transmit request
IDE	ID extend bit
DLC	length of data
SRR	substitute remote request
RRS	remote request storing
EDL	extend data length
res	recessive bit
BRS	bit rate switching
ESI	Error state indicator

The ability to receive and transmit CAN FD messages is enabled by the CAN_MCR[FDEN] bit. Either a recessive R0 bit in CAN frames with 11-bit identifiers or a recessive R1 bit in CAN frames with 29-bit identifiers are decoded as an EDL bit (not a reserved one). A CAN

FD frame is recognized by a recessive EDL bit, while a Classical CAN frame is recognized by a dominant EDL bit. The BRS bit specifies whether this frame switches the bit rate in its data phase. A long frame is decoded according to the DLC field value.

CAN FD messages can be transmitted with two different bit rates.

- The first part of a CAN FD frame, from the Start Of Frame (SOF) bit until the Bit Rate Switch (BRS) bit, also called the arbitration phase, is transmitted with the nominal bit rate based on a set of nominal CAN bit timing configuration values.
- The second part, from the BRS bit until the CRC Delimiter bit, also called the data phase, is transmitted with the data bit rate defined by a second set of CAN data bit timing configuration values.
- Finally, from the CRC Delimiter until the Intermission bits, the transmission resumes to a nominal bit rate. In CAN FD frames with bit rate switching, the bit timing is changed inside the frame at the sample point of the BRS bit if this bit is recessive.

Before the BRS bit, in the CAN FD arbitration phase, the nominal CAN bit timing is used as defined by the CAN_CBT register (also by CAN_CTRL1 register for backward compatibility). Upon detection of a recessive BRS bit, the CAN data bit timing is used as defined by the CAN_FDCBT register.

CAN_FDCTRL[FD_RATE] enables the transmission of all frames with bit rate switching if the BRS bit in the selected Tx MB is set. If FD_RATE is negated, the transmission is performed at a nominal rate regardless of the BRS bit value. The CAN_FDCTRL[FD_RATE] bit can be written any time but takes effect only for the next message transmitted or received. The nominal bit timing is resumed at either the sample point of the CRC Delimiter bit or when an error is detected, whichever occurs first. The following figure shows the mechanism for entering and leaving the data phase when the BRS bit is recessive.

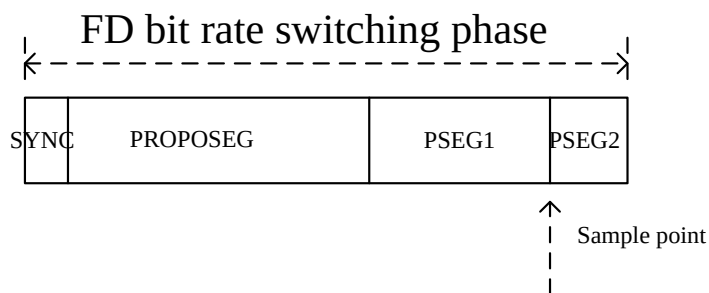


Figure 40-4 CAN FD bit rate switching phase

The maximum configurable bit rate in the CAN FD data phase depends on the clock frequency of the CAN_PE sub-block. For instance, with a CAN_PE clock frequency of 80MHz and the shortest configurable bit time of 10 time quanta, the bit rate in the data phase is 8 Mbit/s.

The value of the ESI bit is determined either by one of the followings:

- The transmitter's error state at the start of the transmission, if the frame is originated in the CAN node
- The original transmitting node in case CAN is acting as a gateway for the message.

If the transmitter is error passive, ESI is transmitted recessive. Otherwise, it is transmitted dominant. The permutations of the relationship between the written value and the transmitted value of the ESI are shown in this table.

Table 40-18 Written versus transmitted values of ESI field

CAN fault confinement status at start of frame	ESI bit of TX MB	transmitted ESI
Error Active	0	0(error active)
Error Passive	0	1(error passive)
Error active	1	1(error passive)

CAN fault confinement status at start of frame	ESI bit of TX MB	transmitted ESI
error passive	1	1(error passive)

There are different CRC polynomials for other CAN frame formats. CRC_15, the first polynomial, is used for all frames in Classical CAN format. CRC_17, the second, is used for frames in CAN FD format with a data field up to sixteen bytes long. CRC_21, the third, is used for frames in CAN FD format with a data field longer than sixteen bytes. Each polynomial results in a Hamming Distance of 6. At the start of the frame, all three CRC polynomials are calculated concurrently. The CRC sequence to be transmitted is selected by the values of the EDL bit and the DLC bit field. CAN decodes EDL and DLC to select the adequate CRC polynomial to check for a CRC error when receiving a message.

In CAN FD format frames, stuff bits are included in the bit stream for the CRC calculation. In Classical CAN format frames, stuff bits are not included. After the transmission of the last bit relevant to the CRC calculation, the CAN_FDCRC register stores the calculated CRC for the transmitted message, with the adequate length according to the type of message, for both CAN FD and non-FD messages. The CAN_CRCCR register reports a valid CRC for Classical CAN messages only.

In CAN FD format frames, the CAN bit stuffing method is changed for the CRC sequence so that the stuff bits are inserted at fixed positions. When CAN transmits a CAN FD frame, a fixed stuff bit is inserted just before the first bit of the CRC sequence, even if the last bits of the preceding field do not fulfill the CAN stuff condition. Additional stuff bits are inserted after each fourth bit of the CRC sequence. The value of any fixed stuff bit is the inverse value of its preceding bit. When CAN receives a CAN FD frame, it discards the fixed stuff bits from the bit stream for the CRC check. A Stuff Error is detected if the fixed stuff bit has the same value as its preceding bit.

CAN detects errors in CAN FD frames the same way as in Classical CAN frames. The error counters RXERRCNT and TXERRCNT in the CAN_ECR register accumulate the counts of Rx and Tx errors, respectively, for both FD and non-FD frames indistinctly.

In addition, there are two extra error counters (RX_ERRCNT_FAST and TX_ERRCNT_FAST) that accumulate Rx and Tx errors happening in the data phase of CAN FD frames with the BRS bit set only. The rules for updating the error counters are the same for both CAN FD and non-FD frames (see CAN_ECR register).

Error Flags BITERR1, BITERR0, ACKERR, CRCERR, FRMERR, and STFERR in the ESR1 register report errors in both CAN FD and non-FD frames. They also generate the ERRINT interrupt if CAN_CTRL1[ERRMSK] is asserted. In addition, the CAN_ESR1 register has additional error flags (BIT1ERR_FAST, BIT0ERR_FAST, CRCERR_FAST, FRAMERR_FAST and STFERR_FAST) to individually indicate the occurrence of errors in the data phase of CAN FD frames with the BRS bit set.

There is no ACKERR detected in the data phase of a CAN FD frame. Fault confinement status reported in CAN_ESR1[FLTCONF] is the same for both CAN FD and Classical CAN frames. It is based on RXERRCNT and TXERRCNT error counters only. Information in RX_ERRCNT_FAST and TX_ERRCNT_FAST counters may be considered as status to help detect the error nature related to the bit rate value.

When CAN is in the data phase, either transmitting or receiving a CAN FD message, and detects an error, it immediately switches back to the arbitration phase and to the nominal rate to start an Error Flag.

Resynchronization and Hard Synchronization occur in CAN FD frames in the same way as in Classical CAN ones. Moreover, a Hard Synchronization is also performed at the recessive to dominant edge from EDL to R0 in CAN FD format frames. CAN does not resynchronize while transmitting in the CAN FD data phase.

40.4.10 Transceiver delay compensation

The CAN FD protocol allows the transmission and reception of data at a bit rate higher than the nominal rate used in the arbitration phase when the message's BRS bit is set. This feature enables the use of rates up to 8 Mbps.

During the data phase of a CAN FD frame, the Transmitter detects a bit error if it cannot

receive its own latest transmitted bit at the sample point of that bit. When bit rate switching is enabled (BRS bit is asserted), the length of the CAN bit time in the data phase can become shorter than the transceiver's loop delay, which thus impedes the correct comparison between the transmitted bit and the received bit within the current CAN bit time interval.

Note that the TDC process defines a secondary sample point at which the transmitted bit is correctly compared with the received bit to check for bit errors. The TDC mechanism can be enabled by the CAN_FDCTRL[TDCEN] bit and is effective only during the data phase of FD frames having the BRS bit set. It has no effect either on non-FD frames, or on FD frames transmitted at a normal bit rate.

The TDC is active from the sample point of the BRS bit until the sample point of the CRC Delimiter bit, if the respective message under transmission has the BRS bit set. When it is active, a comparison is made between the real received bit and the delayed transmitted bit, where the delay is calculated based on the measured transceiver loop delay.

For every transmitted FD frame having the BRS bit set, the delay measurement is triggered by the transition from the recessive EDL bit to the dominant R0 bit. The loop delay is measured in Protocol Engine (PE) clock periods, from the transmitted EDL-R0 edge to the received EDL-R0 edge.

The position of the secondary sample point is defined by the measured loop delay time added to an offset value specified in CAN_FDCTRL[TDCOFF]. CAN_FDCTRL[TDCVAL] bit field stores the result of this calculation. The TDCVAL value saturates at its maximum value of 63 CANCLK when the delay measurement is too long.

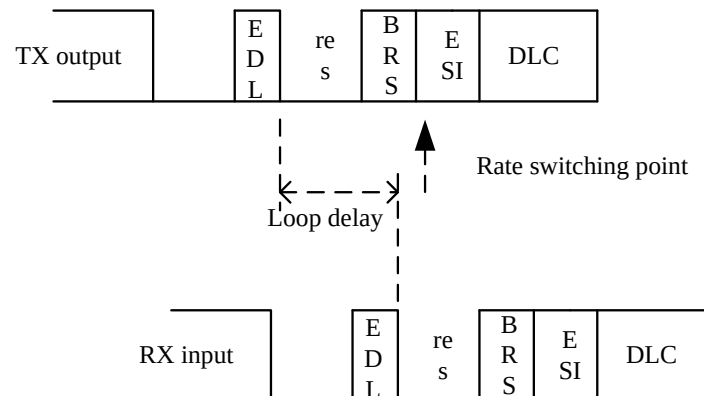


Figure 40-5 transceiver loop measurement

The measured loop delay is not enough to define the secondary sample point for it relates to the CAN bit edges. The transceiver delay compensation offset TDCOFF is used to shift the secondary sample point from the edge to an intermediate point inside the bit time, far away from its edges. Thus, the TDCOFF value cannot be larger than the CAN bit duration in the data phase.

If the secondary sample point (SSP) is set near the CAN bit edge (SYNC field), problems may happen during the bit sampling in the data phase. For the TDC to work reliably, the user must use the optional settings for the offset. To be sure that the bit sampling is performed in the best region, the TDC offset should be configured as shown in this equation:

$$\text{offset} = \text{PSEG1} + \text{PROPSEG} + 2$$

The following figure illustrates the SSP position when these settings are used.

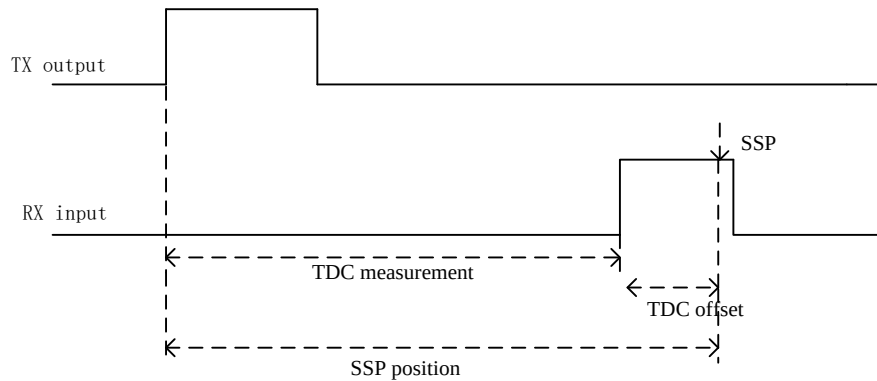


Figure 40-6 SSP position with optimal value

During the data phase of CAN FD frames with bit rate switching enabled, at the onset of every Tx CAN bit, the transmitted Tx bit value is temporarily stored in a buffer and a time countdown based on TDCVAL is started which ends with the comparison of the received Rx bit (delayed by the external loop delay plus the specified offset) with the stored Tx bit. If a bit error is detected at the secondary sample point, the CAN issues an error flag to the CAN bus at the next sample point.

During the arbitration phase, the delay compensation is always disabled. Therefore, the maximum delay for which the CAN's transceiver delay compensation can compensate during the data phase is 3 CAN bit times - 2 T_q. Beyond this limit, the CAN_FDCtrl[TDCFAIL] flag is set to indicate when the Transceiver Delay Compensation mechanism is out of range, thus unable to compensate the transceiver loop delay.

40.4.11 Remote frame

The remote frame is a special kind of frame. The user can program a mailbox to be a Remote Request Frame by configuring the mailbox as Transmit with the RTR bit set to '1'. After the remote request frame is transmitted successfully, the mailbox becomes a Receive Message Buffer, with the same ID as before.

When a remote request frame is received by CAN, it can be treated in two ways, depending on Remote Request Storing (CAN_CTRL2[RRS]) and Rx FIFO Enable (CAN_MCR[RFEN]) bits:

- If RRS is negated the frame's ID is compared to the IDs of the Transmit Message Buffers with the CODE field 0b1010. If there is a matching ID, then this mailbox frame will be transmitted. Note that if the matching mailbox has the RTR bit set, then CAN will transmit a remote frame as a response. The received remote request frame is not stored in a receive buffer. It is only used to trigger a transmission of a frame in response. The mask registers are not used in remote frame matching, and all ID bits (except for RTR) of the incoming received frame should match. In the case that a remote request frame is received and matches a mailbox, this message buffer immediately enters the internal arbitration process. It is still considered as a normal Tx mailbox, with no higher priority. The data length of this frame is independent of the DLC field in the remote frame that initiated its transmission.
- If RFEN is asserted, CAN will not automatically respond to remote request frames that match the Legacy FIFO filtering criteria. If the remote frame matches one of the target IDs, it will be stored in the Legacy FIFO and presented to the CPU. Note that for filtering formats A and B, it is possible to select whether remote frames are accepted or not. For format C, remote frames are always accepted (if they match the ID). Remote Request Frames are considered normal frames, generating a Legacy FIFO overflow when a successful reception occurs, and the Legacy FIFO is already full.

NOTE

Since there is no remote frame in the CAN FD format, and the RTR bit is replaced by a fixed dominant RRS bit, CAN receives and transmits remote frames in the Classical CAN format.

40.4.12 Overload frames

CAN does transmit overload frames due to the detection of the following conditions on CAN bus:

- Detection of a dominant bit in the first/second bit of Intermission.
- Detection of a dominant bit at the 7th bit (last) of End of Frame field (Rx frames)
- Detection of a dominant bit at the 8th bit (last) of Error Frame Delimiter or Overload Frame Delimiter

40.4.13 Message buffer time stamp

The value of the Free Running Timer is sampled at the beginning of the Identifier field on the CAN bus. It is stored at the end of "move-in" in the TIME STAMP field of the message buffer, providing network behavior concerning time.

When the TIMER_SRC bit in CAN_CTRL2 register is asserted, the Free Running Timer is continuously clocked by an external time tick.

When the TIMER_SRC bit in CAN_CTRL2 register is negated, the Free Running Timer is clocked by the CAN bit-clock, which defines the baud rate on the CAN bus. During a message transmission/reception, it increments by one for each bit received or transmitted. When there is no message on the bus, it counts by using the previously programmed baud rate.

The Free Running Timer is not incremented during Disable, Doze, Stop, and Freeze modes. It can be reset upon a specific frame reception, enabling network time synchronization. See more details about the TSYC description in Control 1 Register (CAN_CTRL1).

40.4.14 Protocol timing

The CAN module supports various means to set up bit timing parameters required by the CAN protocol. The Control 1 Register (CAN_CTRL1) has various fields used to control bit timing parameters: PRESDIV, PROPSEG, PSEG1, PSEG2, and RJW.

The CAN Bit Timing register (CAN_CBT) extends the range of the CAN bit timing variables in CAN_CTRL1. The CAN FD Bit Timing register (CAN_FDCBT) provides a second set of CAN bit timing variables to be applied at the data phase of CAN FD frames with the Bit Rate Switch (BRS) set.

NOTE

When the CAN FD feature is enabled, always set CAN_CBT[BTF] and configure the CAN bit timing variables in CAN_CBT.

The PRESDIV field (as well as its extended range EPRESDIV and FDPRESDIV for the data phase bits of CAN FD messages) defines the Prescaler Value (see the equation below) that generates the Serial Clock (Sclock), whose period defines the 'time quantum' used to compose the CAN waveform. A time quantum (Tq) is the atomic unit of time handled by the CAN engine.

$$Tq = (PRESDIV + 1) / fpe\_clk$$

For nominal CAN baud rate:

baud_rate for nominal CAN =

$$fpe\_clk / ((PRESDIV + 1) * (PSEG1 + PROPSEG + PSEG2 + 4))$$

For CAN FD baud rate:

baud_rate for FD CAN =

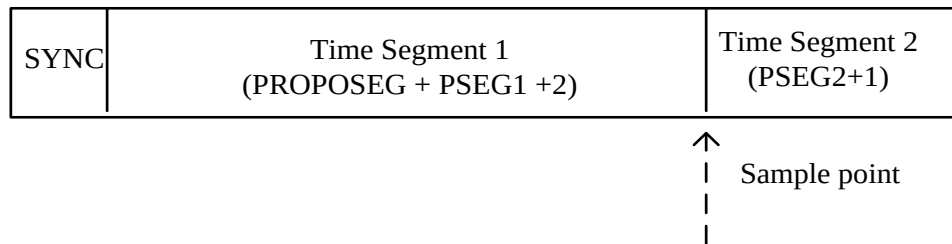
$$fpe\_clk / ((PRESDIV + 1) * (FPSEG1 + FPROPSEG + FPSEG2 + 3))$$

A bit time is subdivided into three segments:

- SYNC_SEG: This segment has a fixed length of one-time quantum. Signal edges are expected to occur within this section.
- Time Segment 1: This segment contains the Propagation Segment and Phase Segment 1 of the CAN standard. It can be programmed by setting the PROPSEG and the PSEG1 fields of the CAN_CTRL1 Register so that their sum (plus 2) is in the range of 2 to 16 time quanta. When CAN_CBT[BTF] bit is asserted, CAN uses EPROPSEG and

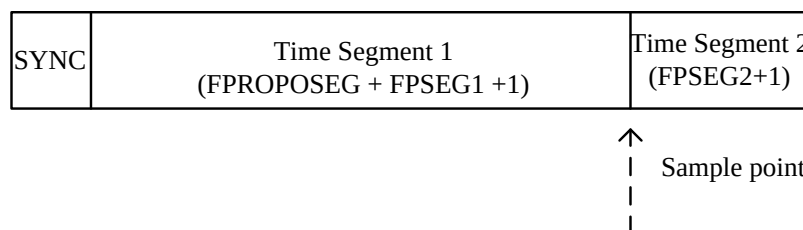
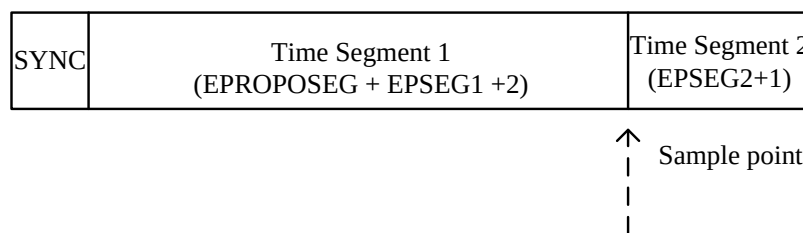
EPSEG1 fields from the CAN_CBT register so that their sum (plus 2) is in the range of 2 to 96 time quanta. For messages in CAN FD format with the BRS bit set, CAN uses FPROPSEG and FPSEG1 from CAN_FDCBT instead, so that their sum (plus 1) is in the range of 2 to 39 time quanta.

- Time Segment 2: This segment represents the Phase Segment 2 of the CAN standard. It can be programmed by setting the PSEG2 field of the CAN_CTRL1 Register (plus 1) to be 2 to 8 time quanta long. When CAN_CBT[BTF] bit is asserted, CAN uses EPSEG2 fields of CAN_CBT register so that its value (plus 1) is in the range of 2 to 32 time quanta. For messages in CAN FD format with the BRS bit set, CAN uses FPSEG2 from CAN_FDCBT instead, so that its value (plus 1) is in the range of 2 to 8 time quanta. The Time Segment 2 cannot be smaller than the Information Processing Time (IPT), which value is 2 time quanta in CAN.



CAN segment within bit time(example with CAN_CTRL1 for non-fd)

Figure 40-7 CAN segment within bit time (NON-FD)



CAN segment within bit time(example with CAN_CBT and CAN_FDCBT for FD)

Figure 40-8 CAN segment with bit time (FD)

The following table shows examples of the CAN compliant segment settings for Classical CAN format (Bosch CAN 2.0B) (non-FD) messages.

Table 40-19 Bosch 2.0B standard CAN bit time segment setting

Time Segment1 (PROPSEG+PSEG1+2)	Time Segment2 (PSEG2 +1)	Re-synchronization jump width
5...10	2	1...2
4...11	3	1...3
5...12	4	1...4

Time Segment1 (PROPSEG+PSEG1+2)	Time Segment2 (PSEG2 +1)	Re-synchronization jump width
6...13	5	1...4
7...14	6	1...4
8...15	7	1...4
9...16	8	1...4

40.4.15 Arbitration and matching timing

During normal reception and transmission, the matching, arbitration, move-in, and move-out processes are executed during specific time windows inside the CAN frame, as shown in the following figures.

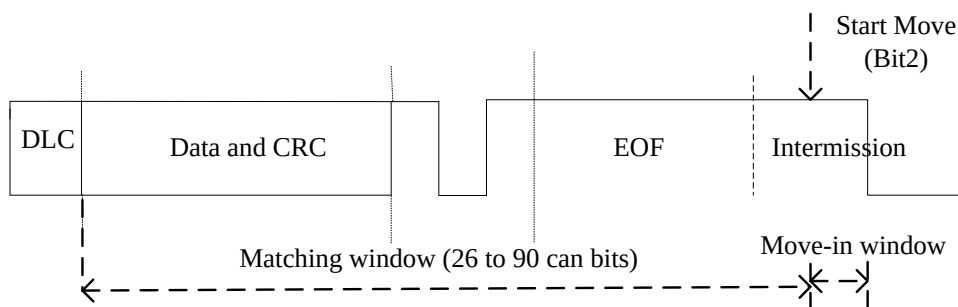


Figure 40-9 Matching and move-in time windows

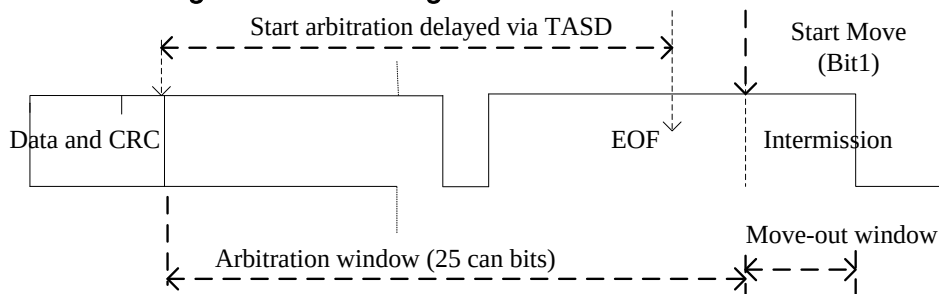


Figure 40-10 Arbitration and move-out time windows

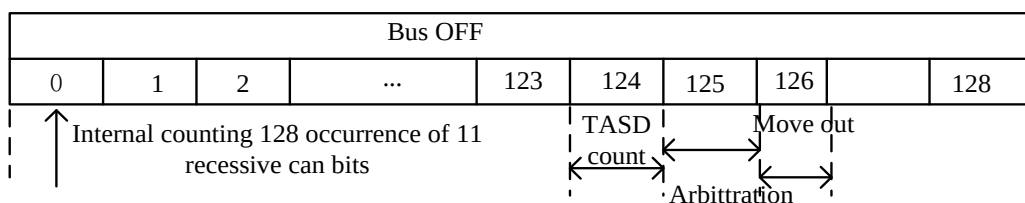


Figure 40-11 Arbitration at end of bus off and move-out windows

40.4.16 TX arbitration start delay

The Tx Arbitration Start Delay (TASD) bit field in Control 2 register (CAN_CTRL2[TASD]) is a variable that indicates the number of CAN bits used by CAN to delay the Tx Arbitration process start point from the first bit of CRC field of the current frame. This variable can be written only in Freeze mode for it is blocked by hardware in other modes.

The transmission performance is impacted by the ability of the CPU to reconfigure Message Buffers (MBs) for transmission after the end of the internal Arbitration process, where CAN finds the winner MB for transmission. If the Arbitration ends too early prior to the first bit of

the Intermission field, then there is a chance that the CPU reconfigures some Tx MBs and the winner MB is no longer the best candidate to be transmitted.

TASD is useful to optimize the transmission performance by defining the Arbitration start point, as shown in the next following figure, based on factors such as:

- The peripheral-to-oscillator clock ratio
- CAN bit timing variables which determine the CAN bit rate
- The number of Message Buffers (MBs) in use by the Matching and Arbitration processes.

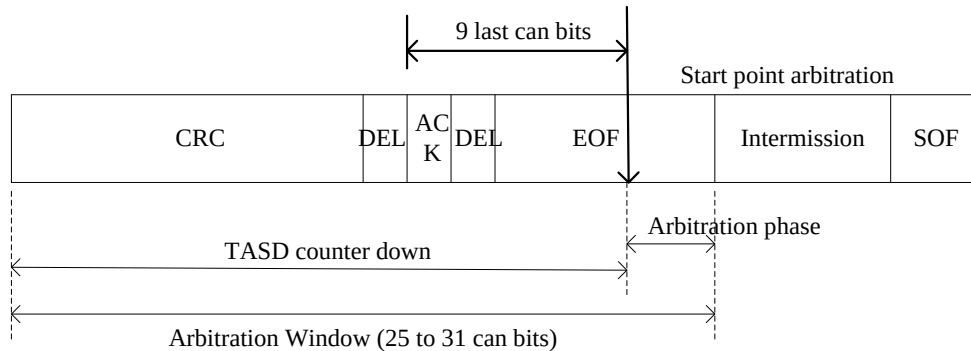


Figure 40-12 Optimal TX Arbitration start point

In terms of CAN bits, the duration of an Arbitration process, is directly proportional to the number of available MBs and to the CAN bit rate, and inversely proportional to the peripheral clock frequency.

The optimal Arbitration timing is when the last MB is scanned right before the first bit of the Intermission field of a CAN frame. For example, if there are few MBs and the peripheral/oscillator clock ratio is high and the CAN baud rate is low, then the Arbitration can be placed closer to the frame's end, adding more delay to its start point, and vice versa.

If TASD is set to 0 then the Arbitration start is not delayed, and more time is reserved for Arbitration. On the other hand, if TASD is close to 24, the CPU can configure a Tx MB later and less time is reserved for Arbitration. If too little time is reserved for Arbitration, the CAN may not be able to find a winner MB in time to be transmitted with the best chance to win the bus arbitration against external nodes on the CAN bus.

The optimal TASD value can be calculated as follows:

- For CAN FD frames and $(MAXMB + 1) < NMBend$,

$$TASD = 31 - (2 * (MAXMB + 1) + 4) / (CPCBn)$$
- For CAN FD frames and $(MAXMB + 1) > NMBend$,

$$TASD = 22 - (2 * (MAXMB + 1) - NMBend) / (CPCBf)$$
- For NON-FD frames, $TASD = 25 - (2 * (MAXMB + 1) + 4) / (CPCB)$

NOTE

- $NMBend = ((9 * CPCBn) - 4) / 2$
- nominal baud rate $BAUDRATE_n = f_{pe\_clk} / \{(1 + (EPROPSEG + 1) + (EPSEG1 + 1) + (EPSEG2 + 1)) * (EPRES DIV + 1)\}$
- FD baud rate $BAUDRATE_f = f_{pe\_clk} / \{(1 + (FPROPSEG + 1) + (FPSEG1 + 1) + (FPSEG2 + 1)) * (FPRES DIV + 1)\}$
- number of bus clocks per can bit in nominal data rate $CPCB_n = f_{bus\_clk} / BAUDRATE_n$
- number of bus clocks per can bit in FD data rate $CPCB_f = f_{bus\_clk} / BAUDRATE_f$
- f_{bus_clk} is the frequency of CHI bus clock.
- $EPROPSEG$, $EPSEG1$, $EPSEG2$ are from CAN_CBT and CAN_CTRL1 registers
- $FPROPSEG$, $FPSEG1$, $FPSEG2$ are from CAN_FDCBT register.

The following tables give the TASD value calculated for some configuration cases.

- clock ratio=1:1 (for example, CHI bus clock 40M Hz, PE clock 40M Hz)
- baud rate in arbitration phase=1M baud

Table 40-20 TASD values when 1:1

Number of MB	TASD values	Maximum baud rate on data phase Mbaud
16	24	Invalid
32	23	6.67
54	22	5
64	21	3.33
96	20	1.6
128	18	1.05

- Clock ratio=1:1 (for example, CHI bus clock 40M Hz, PE clock 20M Hz)
- Baud rate in arbitration phase=1M baud

Table 40-21 TASD values when 2:1

Number of MB	TASD values	Maximum baud rate on data phase Mbaud
16	24	Invalid
32	23	4
54	22	4
64	21	3.33
96	20	1.54
128	18	1.05

40.4.17 Detection and correction of memory errors

The CAN module supports detection and correction of errors in memory read accesses. The error correction mechanism ensures that in this 13-bit word, errors in one bit can be corrected (correctable errors) and errors in 2 bits can be detected but not corrected (non-correctable errors). Errors in more than 2 bits may not be detected. In case of non-correctable errors, the corrupted data is not changed by the error correction logic. When a read access is performed, the parity bits are used to calculate a syndrome, which indicates the error in each byte.

Memory errors are indicated to the Host through Error Status Register (CAN_ERRSR) and bus transfer errors, and reported through report registers including Error Report Address Register (CAN_RERRAR), Error Report Data Register (CAN_RERRDR) and Error Report Syndrome Register (CAN_RERRSYNR).

The error detection and correction mechanism can be controlled by the CAN_MECCR[ECCDIS] bit. When disabled, updates on indications and reporting registers are stopped, but the parity bits are still calculated and written along with data in memory write operations to ensure that memory has consistent parity bits associated to the data.

Note: All CAN memory must be initialized before starting its operation in order to have the parity bits in memory properly updated. CAN_CTRL2[WRMFRZ] grants write access to all memory positions that require initialization. The CAN_RXMGMSK, CAN_RX14MASK, CAN_RX15MASK, and CAN_RXFGMASK registers need to be initialized as well. CAN_MCR[RFEN] shall not be set during memory initialization.

To avoid accidentally changing the critical error correction configuration, this protocol must be followed to enable the update of CAN_MECCR register:

1. By default, CAN_CTRL2[ECRWRE]= 0 and CAN_MECCR[ECRWRDIS]= 1
2. Set CAN_CTRL2[ECRWRE] bit
3. Clear CAN_MECCR[ECRWRDIS] bit

4. All writes to CAN_MECR shall keep CAN_MECR[ECRWRDIS] cleared
5. After configuration is done, lock the Memory Error Control Register (CAN_MECR) by either setting CAN_MECR[ECRWRDIS] or clearing CAN_CTRL2[ECRWRE].

40.4.17.1 Sources of the memory access

The CAN memory can be accessed by two major sources (or requestors):

- by Host (CPU): the largest word accessed is 32-bit wide
- by CAN internals processes (Rx Matching, Tx Arbitration, Move-in on reception, Move-out on transmission): the largest word accessed is 64-bit wide

The way that non-correctable errors are indicated and reported depends on the source of access.

40.4.17.2 Error indication

Non-correctable errors detected in memory reads requested by Host are indicated separately than the ones detected in requests by CAN internal processes. CAN makes no distinction of the source of the access when correctable errors are detected. Memory errors are indicated by flags HANCEIF, FANCEIF, CEIF in CAN_ERRSR, and each flag will raise an interrupt unless it is masked by mask bits in CAN_MECR. If both non-correctable and correctable errors are found in different bytes in the same read operation, both flags are set.

A non-correctable error detected in Host access is also indicated as a bus transfer error. A bus wait request may be asserted to extend the memory transaction to the moment the report registers are updated. This indication cannot be masked. If the flag bit CAN_ERRSR[HANCEIF] is not masked, the same non-correctable error will raise a bus transfer error and an interrupt request.

Note: The bus transfer may trigger hardfault on reading the vector for the original exception. More details refer to ARMv7-M Architecture Reference Manual.

Each indication flag has one Overrun flag in CAN_ERRSR. They do not request interrupts, and for non-correctable errors just indicate that other errors of the same nature were detected after current error being treated, while for correctable errors indicate that other errors of the same nature were detected before the current error being treated. The recommended handling sequence for error indication is as following:

1. Get error report information from report registers
2. Use this information to take proper measures in the application
3. Clear the HANCEIF, FANCEIF, CEIF flags
4. If the Overrun flag is active:
 - a. Alert application that at least one error could not be handled
 - b. Clear the Overrun flag

The CAN internal processes can access memory in transactions larger than 32-bits. For the indication, this kind of access is considered a consecutive sequence of 32-bit accesses. If errors are found in 2 or more 32-bit words the Interrupt and Overrun flags are set simultaneously.

40.4.17.3 Error reporting

The address, data and syndrome information are updated in CAN_RERRAR, CAN_RERRDR and CAN_RERRSYNR registers respectively simultaneously along with the error flags, according to these rules:

1. If any of the 2 non-correctable error flags is currently set, the report registers are not updated (the previous non correctable error reporting is preserved)
2. Otherwise (either no error flag is currently set or only the correctable error flag is currently set), the report registers are updated according to the new error; or according to the most severe of new errors if non-correctable and correctable errors are simultaneously detected.

The address reported in CAN_RERRAR and defined in CAN_ERRIAR are not the same listed in the module memory map. It is recommended to set CAN_MECR[RERRDIS] bit to assure coherence of the error report registers.

40.4.17.4 Response to errors

Correctable errors have no consequence on CAN operation because affected data is corrected before its use by the host or CAN internal processes.

For host-initiated reads, a non-correctable error may affect the host, but does not affect CAN operation.

Non-correctable errors detected on memory reads requested by the CAN internal processes may result in incorrect operation depending on the state of the CAN_MECR[NCEFAFRZ] bit, as follows:

- During reception (either Matching or Move-in processes), when a non-correctable error occurs, an incorrect destination may be selected to store the incoming frame, a corrupted frame may be stored in the correct destination, or both. In case CAN_MECR[NCEFAFRZ] is set, CAN stops operation automatically and enters in Freeze mode to prevent corrupted data from being treated as valid by CAN internal processes. When CAN_MECR[NCEFAFRZ] is negated, CAN continues working and a corrupted frame is received.
- During Arbitration process, when a non-correctable error occurs, either a non-highest priority Tx Message Buffer may be mistakenly selected for transmission or its data may be corrupted. In case CAN_MECR[NCEFAFRZ] is set, CAN stops operation automatically and enters in Freeze mode before starting the Move-out. When CAN_MECR[NCEFAFRZ] is negated, CAN proceeds to Move-out with a corrupted frame that will be transmitted on the CAN bus.
- During Move-out process, when a non-correctable error occurs, a corrupted frame is copied from the selected Tx MB that won the Arbitration to the Tx SMB for transmission. In case CAN_MECR[NCEFAFRZ] is set, CAN stops operation automatically and enters in Freeze mode before starting the transmission. When NCEFAFRZ is negated, the corrupted frame is transferred from the Tx SMB to the Protocol Engine (PE) sub-block and is transmitted on the CAN bus.
- A non-correctable error can also be detected beyond the Move-out process, when Tx data is read from Tx SMB (buffer located in RAM) to be transferred to the PE sub-block for transmission. In this case, a frame with corrupted ID and/or data is transmitted on the CAN bus. To prevent the frame from being successfully received by the external nodes, CAN inverts all bits in the CRC field (CRC sequence plus CRC Delimiter), and transmits an Error Flag just after CRC Delimiter as a result of self-detecting a Bit1 Error and a Form Error due to the CRC field inversion. When CAN_MECR[NCEFAFRZ] is set, CAN stops operation automatically and enters in Freeze mode just after the Error Frame. When CAN_MECR[NCEFAFRZ] is negated, CAN may attempt to re-transmit the same frame, as long as no other higher priority Tx MB is subsequently configured for transmission. In the event the non-correctable error persists, CAN eventually reaches the Bus Off state because of consecutive error detection. The CAN_ECR[TXERRCNT] is updated every time the CAN inverts the CRC field causing errors as described above.

When CAN_MECR[NCEFAFRZ] is set and CAN enters in Freeze Mode, only the CPU can cause CAN to exit Freeze mode and resume Normal Mode. The assertion of the CAN_MECR[NCEFAFRZ] bit is the only way to prevent corrupted frames from being transmitted on the CAN bus up to the Move-out internal process.

40.4.17.5 Error injection

The error injection registers CAN_ERRIAR, CAN_ERRIDPR, and CAN_ERRIPPR are used to inject errors in memory reads in order to force errors and consequently update the indication and reporting registers. The relation between the error injection addresses and the respective ones in the module memory map is shown in Table 1-1 and Table 1-2. The recommended injecting sequence is as following:

1. Set CAN module into freeze mode(if read need)
2. Set CAN_CTRL2[ECRWRE] and [WRMFRZ] both as 1
3. Write CAN_ERRIAR register via column of "ECC Injection address"
4. Write CAN_ERRIDPR and CAN_ERRIPPR registers to hold error pattern
5. Memory read via column of "Memory map".

Table 40-22 ECC injection and RAM address mapping when Number_of_MB =64

CAN register	Memory map	ECC Injection address
MBs	0x0080-0x047C	0x0000-0x03FC
RXIMRs	0x0880-0x0A7C	0x0400-0x04FC
RXFIR	0x0A80-0x0A94	0x0500-0x0514
RXMGMSK	0x0AA0	0x0520
RXFGMASK	0x0AA4	0x0524
RX14MASK	0x0AA8	0x0528
RX15MASK	0x0AAC	0x052C
Tx SMB (FDEN=0)	0x0AB0-0x0ABC	0x0530-0x053C
Rx SMB0 (FDEN=0)	0x0AC0-0x0ACC	0x0540-0x054C
Rx SMB1 (FDEN=0)	0x0AD0-0x0ADC	0x0550-0x055C
Tx SMB (FDEN=1)	0x0F28-0x0F6C	0x0530-0x0574
Rx SMB0 (FDEN=1)	0x0F70-0x0FB4	0x0578-0x05BC
Rx SMB1 (FDEN=1)	0x0FB8-0x0FFC	0x05C0-0x0604

Table 40-23 ECC injection and RAM address mapping when Number_of_MB =128

CAN register	Memory map	ECC Injection address
MBs	0x0080-0x087C	0x0000-0x07F0
RXIMRs	0x0880-0x0A7C	0x0800-0x09FC
RXFIR	0x0A80-0x0A94	0x0A00-0x0A14
RXMGMSK	0x0AA0	0x0A20
RXFGMASK	0x0AA4	0x0A24
RX14MASK	0x0AA8	0x0A28
RX15MASK	0x0AAC	0x0A2C
Tx SMB (FDEN=0)	0x0AB0-0x0ABC	0x0A30-0x0A3C
Rx SMB0 (FDEN=0)	0x0AC0-0x0ACC	0x0A40-0x0A4C
Rx SMB1 (FDEN=0)	0x0AD0-0x0ADC	0x0A50-0x0A5C
Tx SMB (FDEN=1)	0x0F28-0x0F6C	0x0A30-0x0A74
Rx SMB0 (FDEN=1)	0x0F70-0x0FB4	0x0A78-0x0ABC
Rx SMB1 (FDEN=1)	0x0FB8-0x0FFC	0x0AC0-0x0B04

The injection is done by flipping the data and parity bits correspondent to the bits in 1 in CAN_ERRIDPR and CAN_ERRIPPR registers.

Injection can be selected specifically for memory accesses requested by Host or by CAN internal processes. In case of accesses larger than 32-bits, the EXTERRIE bit in CAN_MECR register extends the injection pattern, replicating it in 32-bit words to fill the width of the access.

Note: It is very unlikely, but error injection may correct a bit with error. This will not raise the error flags and reports as expected.

To ensure coherence among error injection registers and avoid spurious error injections, the HAERRI and FAERRIE bits of CAN_MECR shall be cleared while configuring the memory injection registers.

40.4.18 CLOCK domain and restrictions

The CAN module has two clock domains asynchronous to each other:

- The Bus Domain feeds the Control Host Interface (CHI) submodule and is derived from the peripheral clock. Notice that “peripheral clock” occurring on whole chapter means bus clock.
- The PE Domain feeds the CAN Protocol Engine (PE) submodule and is derived directly from function clock. Notice that “oscillator clock” occurring on whole chapter means function clock.

When doing matching and arbitration, CAN needs to scan the whole Message Buffer memory during the time slot of one CAN frame, comprised of some CAN bits. To have sufficient time to do that, the following requirement must be observed:

- The bus clock frequency must be bigger than the protocol engine clock frequency from function clock.
- There must be a minimum number of peripheral clocks per CAN bit, as specified in the table shown below.

Table 40-24 Minimum number of peripheral clocks per CAN bit for NON-FD format

Number of MB	CAN_MCR[RFEN]	Minimum peripheral clocks per CAN bit
16	0	16
32	0	16
64	0	25
96	0	37
128	0	49
16	1	16
32	1	17
64	1	30
96	1	42
128	1	54

For classical frame format, the minimum number of peripheral clocks per CAN bit specified in the preceding table determines the minimum peripheral clock frequency for a given number of Mailboxes and an expected CAN bit rate. The CAN bit rate depends on the number of time quanta in a CAN bit, which can be defined by adjusting one or more of the bit timing values in either the Control 1 Register (CAN_CTRL1) or CAN Bit Time register (CAN_CBT). The time quantum (Tq) is defined in Protocol timing. The minimum number of time quanta per CAN bit must be 8. Therefore, the oscillator clock frequency should be at least 8 times the CAN bit rate.

The number of CAN bits in the Data Phase of a FD Frames with the BRS bit set (fast CAN bits, in short) depends on the number of data bytes in the payload. The number of fast CAN bits (NumofFastBits) can be determined in the table below. The less the number of data bytes is, the less the number of the fast CAN bits is, and less time is available for CAN to scan the whole Message Buffer memory during the internal matching and arbitration processes.

Table 40-25 Number of fast CAN bits in FD frame

Minimum number of data bytes	DLC field	NumofFastbits
0	0x0	21
1	0x1	29
2	0x2	37
3	0x3	45
4	0x4	53

Minimum number of data bytes	DLC field	NumofFastbits
5	0x5	61
6	0x6	69
7	0x7	77
8	0x8	85
12	0x9	117
16	0xA	149
20	0xB	186
24	0xC	218
32	0xD	282
48	0xE	410
64	0xF	538

The critical part of a CAN FD frame is during the Data Phase, where the CAN bit rate is faster than in the Arbitration Phase. The minimum number of peripheral clocks per fast CAN bit (MinNumClkFastBit) can be calculated to ensure that enough time is available for CAN to scan the Message Buffer memory during reception and transmission. The equation below calculates this constraint.

$$\text{MinimumCLKfastbitA} = \{(8.5 * \text{MAXMB}) + 64 - (9 * \text{CPCBn})\} / \text{NumofFastbits}$$

The clock domain crossing circuit between the CHI and PE sub-blocks also imposes a minimum number of peripheral clocks per fast CAN bit for the handshake mechanism to work properly without losing status information through the interface, as shown in the equation below.

$$\text{MinimumCLKfastbitB} = 3 * (1 + \text{fchi}/\text{fpe})$$

Therefore, the minimum number of peripheral clocks per fast CAN bit (MinNumClkFastBit) is determined by the larger of the two values calculated above.

$$\text{MinimumCLKfastbit} = \text{maximum}(\text{MinimumCLKfastbitA}, \text{MinimumCLKfastbitB})$$

The maximum CAN bit rate in the Data Phase of CAN FD frames (DataBitRateMAX) can be calculated as below.

$$\text{DATABITRATEmax} = \text{fpe\_clk} / \{ \text{ROUNDUP}(\text{MinimumCLKfastbit} * \text{fpe\_clk}) / \text{fchi} \}$$

The peripheral and oscillator clock frequencies, the maximum number of mailboxes and the expected nominal bit rate affect the maximum data bit rate attainable by CAN in CAN FD mode. Besides, the data bit rate depends on the minimum payload size of FD frames used in a given application.

To illustrate how the CAN FD bit rate is affected by the configuration of CAN variables, the following application example is displayed with the peripheral and oscillator clock frequencies set to 50 MHz and 40 MHz respectively.

Step 1 - The number of peripheral clocks per CAN bit in nominal bit rate is calculated as below, when setting the nominal bit rate to 1 Mbps.

$$\text{CPCBn} = 50 \text{ M}/1\text{M} = 50$$

Step 2 - The number of fast CAN bits (NumOfFastBits) is determined in the table presented above. For example, if the minimum payload in FD frames is 8 bytes, then there are 85 CAN bits in the Data Phase.

Step 3 - Assuming the maximum number of mailboxes is 96, the minimum number of peripheral clocks per fast CAN bit (MinNumClkFastBit) can be calculated.

$$\text{MinimumCLKfastbitA} = (8.5 * 96 + 64 - (9 * 50)) / 85 = 5.06$$

$$\text{MinimumCLKfastbitB} = 3 * (1 + 50/40) = 6.75$$

$$\text{MinimumCLKfastbit} = \text{maximum}(5.06, 6.75) = 6.75$$

Step 4 - The maximum CAN bit rate in the Data Phase can be calculated via the following equation.

$$\text{DATABITRATEmax} = 40\text{M} / \text{ROUNDUP}(6.75 * 40\text{M}/50\text{M}) = 6.67\text{Mbps}$$

As demonstrated in this example, even though the oscillator clock frequency (40 MHz) is

adequate to generate a data rate of 8 Mbps in CAN FD mode, the specific CAN configuration limits this rate to 6.667 Mbps. This limitation is mainly due to the low peripheral clock frequency that imposes the MinNumClkFastBitB bound.

The table below shows the maximum data rate for CAN FD according to clock frequencies, payload size, and the number of available mailboxes. From the table, it is noticeable that when the available mailboxes are reduced, the CAN then achieves a data rate up to 8 Mbps.

Table 40-26 Maximum CAN bit rate on CAN FD frames

CHI bus frequency	number of data bytes	num of available MB maximum	data rate (Mbps)
40	8	94	6.667
40	8	114	5
40	12	117	6.667
40	12	128	5.714
50	12 to 64	128	6.667
60	8	126	8.0
60	12	128	8
67	6	128	8
80	3	128	8
100	0	128	8

40.4.19 Interrupt and DMA signals

The module has many interrupt sources: interrupts due to message buffers and interrupts due to the OR interrupts from MBs, Bus Off, Bus Off Done, Error, Error Fast (errors detected in the data phase of CAN FD format messages with the BRS bit set), Wake Up, Wake Up Match, Wake Up Timeout, Tx Warning, and Rx Warning.

Each one of the message buffers can be an interrupt source, if its corresponding IMASK bit is set. There is no distinction between Tx and Rx interrupts for a particular buffer, under the assumption that the buffer is initialized for either transmission or reception. Each of the buffers has an assigned flag bit in the CAN_IFLAG registers. The bit is set when the corresponding buffer completes a successful transfer and is cleared when the CPU writes it to 1 (unless another interrupt is generated concurrently).

If the Legacy Rx FIFO is enabled (CAN_MCR[RFEN] = 1) and DMA is disabled (CAN_MCR[DMAE] = 0), the interrupts corresponding to MBs 0 to 7 have different meanings. Bit 7 of the CAN_IFLAG1 register becomes the "Legacy FIFO Overflow" flag; bit 6 becomes the "Legacy FIFO Warning flag", bit 5 becomes the "Frames Available in Legacy FIFO" flag and bits 1-4 are unused. See the description of the Interrupt Flags 1 Register (CAN_IFLAG1) for more information. (Legacy RXFIFO could not be enabled when FD is enabled).

If both Legacy Rx FIFO and DMA are enabled (CAN_MCR[RFEN] and CAN_MCR[DMAE] = 1) the CAN does not generate any Legacy FIFO interrupt. Bit 5 of the CAN_IFLAG1 register still indicates "Frames Available in Legacy FIFO" and generates a DMA request. Bits 6, 7, 1-4 are unused.

For a combined interrupt where multiple MB interrupt sources are OR'd together, the interrupt is generated when any of the associated MBs (or FIFO, if applicable) generates an interrupt. In this case, the CPU must read the CAN_IFLAG registers to determine which MB or FIFO source caused the interrupt.

The interrupt sources for Bus Off, Bus Off Done, Error, Error Fast, Wake Up, Tx Warning and Rx Warning generate interrupts like the MB interrupt sources, and can be read from CAN_ESR1 register. The Bus Off, Error, Tx Warning, and Rx Warning interrupt mask bits are located in the CAN_CTRL1 Register; the Wake-Up interrupt mask bit is located in the CAN_MCR.

The interrupt sources for Pretended Networking (Wake up by Match Flag and Wake up by Timeout Flag) can be read in the CAN_WU_MTC Register and the respective interrupt masks bits are located in CAN_CTRL1_PN Register.

Chapter 41 Debug

41.1 Introduction

This device's debug is based on the ARM coresight architecture and it is configured in each device to provide the maximum flexibility as allowed by the restrictions of the pinout and other available resources.

Several debug interfaces are supported:

- IEEE 1149.1 JTAG
- Serial Wire Debug (SWD)

41.2 Block diagram

The basic Cortex-M4F debug architecture is very flexible. The following diagram shows the topology of the core debug architecture and its components.

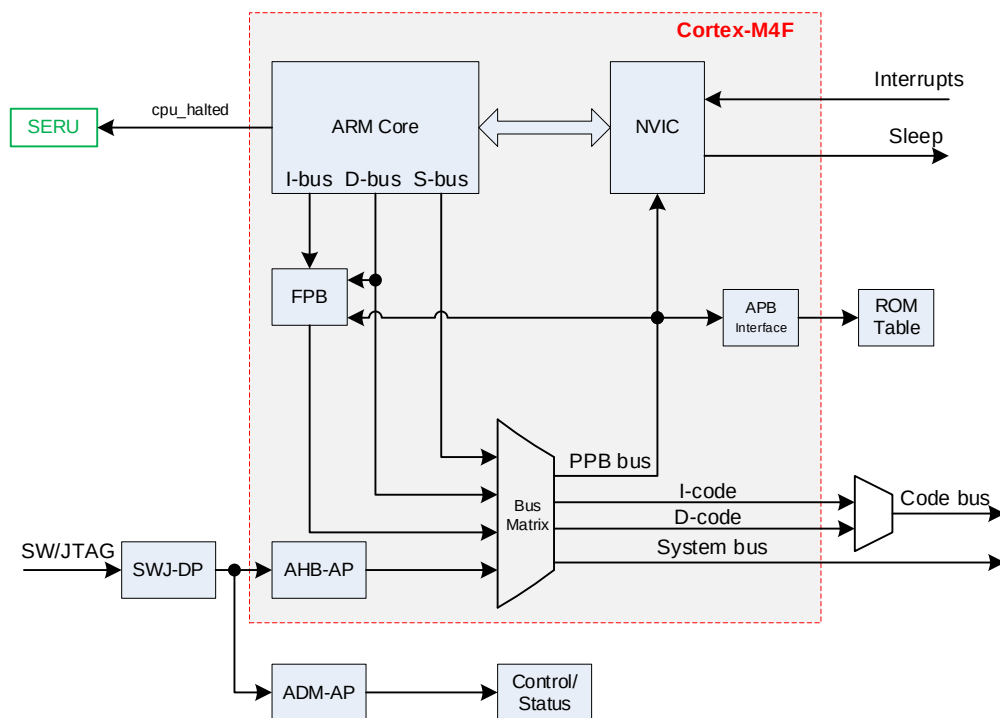


Figure 41-1 Cortex-M4F debug topology

The following table describes each of the debug components.

Table 41-1 Debug components description

Module	Description
SWJ-DP	Modified Debug Port with support for SWD, JTAG
AHB-AP	AHB Master Interface from JTAG to debug module and SOC system memory maps.
ADM-AP	Provides centralized control and status registers for an external debugger to control the device.
ROM Table	Identifies which debug IP is available.
Core Debug	Single step, Register Access, Run, Core Status

For more information on ARM debug components, see these documents:

- *ARMv7-M Architecture Reference Manual*
- *ARM Debug Interface v5.1*
- *ARM CoreSight Architecture Specification*

The configuration of the JTAG controller, and debug port is illustrated in the following figure:

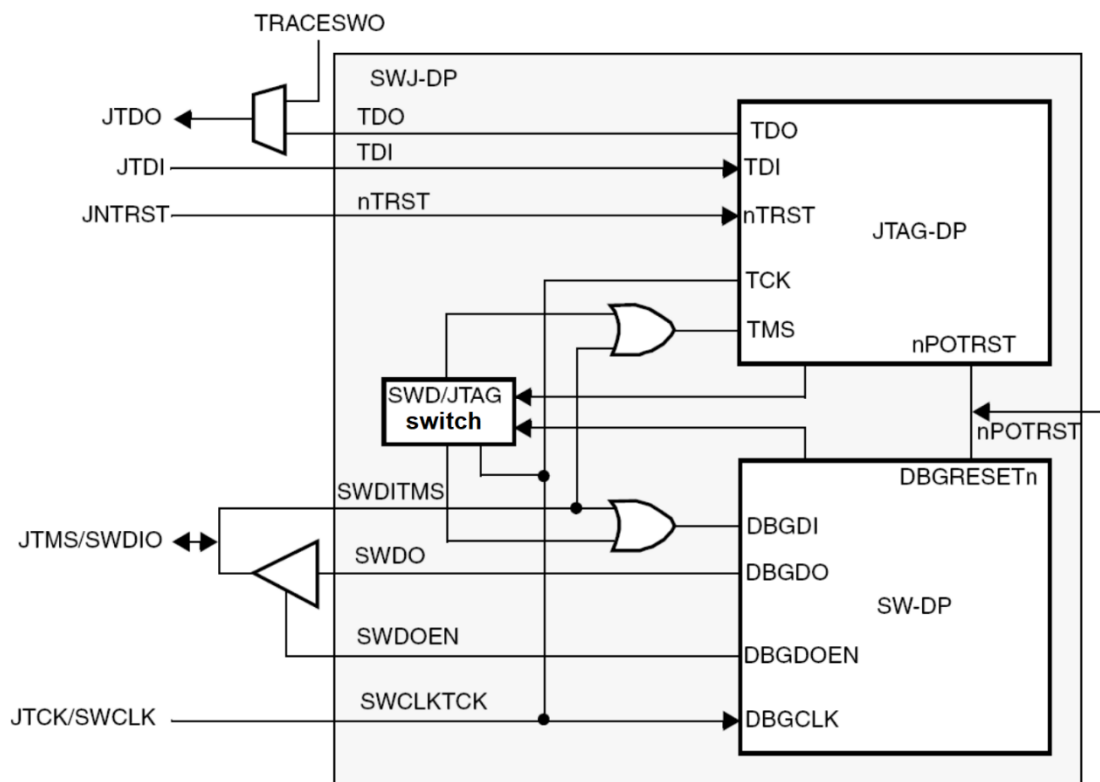


Figure 41-2 JTAG controller

The debug port comes out of reset in standard JTAG mode and is switched into SWD mode by the following sequences. Once the mode has been changed, unused debug pins can be reassigned to any of their alternative muxed functions.

1. Send more than 50 TCK cycles with TMS (SWDIO) =1
 2. Send the 16-bit sequence on TMS (SWDIO) = 0111_1001_1110_0111 (MSB transmitted first)
 3. Send more than 50 TCK cycles with TMS (SWDIO) =1
- JTAG IR codes are as below:

Table 41-2 JTAG IR code

Instruction	Code[3:0]	Instruction Summary
IDCODE	0000	Selects device identification register for shift
SAMPLE/PRELOAD	0010	Selects boundary scan register for shifting, sampling, and preloading without disturbing functional operation
SAMPLE	0011	Selects boundary scan register for shifting and sampling without disturbing functional operation
EXTEST	0100	Selects boundary scan register while applying preloaded values to output pins
ABORT	1000	ARM JTAG-DP Instruction

Instruction	Code[3:0]	Instruction Summary
HIGHZ	1001	Selects bypass register while three-stating all output pins and asserting functional reset
DPACC	1010	ARM JTAG-DP Instruction
APACC	1011	ARM JTAG-DP Instruction
CLAMP	1100	Selects bypass register while applying preloaded values to output pins and asserting functional reset
ARM_IDCODE	1110	ARM JTAG-DP Instruction
BYPASS	1111	Selects bypass register for data operations
Factory debug reserved	0101, 0110, 0111, 1101	Intended for factory debug only
ARM JTAG-DP Reserved	1000, 1010, 1011, 1110	These instructions will go the ARM JTAG-DP controller. Please refer to ARM JTAG-DP documentation for more information on these instructions.
Reserved	All other opcodes	Decoded to select bypass register

41.3 ROM table

The ROM table is used to hold the information about the debug components.

Table 41-3 Bit assignments in the ROM table

Bits	Name	Description
[31:12]	Address offset	The base address of the component, relative to the ROM address. Negative values are permitted using two's complement. ComponentAddress = ROMAddress + (AddressOffset SHL 12).
[11:2]	-	Reserved SBZ.
[1]	Format	1b: 32-bit format. In the DAP Debug ROM this is set to 1. 0b: 8-bit format.
[0]	Entry present	Set HIGH to indicate an entry is present.

Table 41-4 CM4F ROM table

Component	Base address
SCS	0xE000E000

41.4 Debug port pin descriptions

The debug port pins default after POR to their JTAG functionality with the exception of JTAG_TRST_b and can be later reassigned to their alternate functionalities. In SWD mode, JTAG_TDI and JTAG_TRST_b can be configured to alternate GPIO functions.

Table 41-5 Debug port pins

Pin name	JTAG debug port		SWD debug port		Internal pull-up \down
	Type	Description	Type	Description	

Pin name	JTAG debug port		SWD debug port		Internal pull-up \down
	Type	Description	Type	Description	
JTAG_TMS/ SWD_DIO	I/O	JTAG Test Mode Selection	I/O	Serial Wire Data	Pull-up
JTAG_TCLK/ SWD_CLK	I	JTAG Test Clock	I	Serial Wire Clock	Pull-down
JTAG_TDI	I	JTAG Test Data Input	-	-	Pull-up
JTAG_TDO	O	JTAG Test Data Output	-	-	-
JTAG_TRST_b	I	JTAG Reset	-	-	Pull-up

41.5 ADM-AP status and control registers

Through the Additional Debug Access Port (DAP), the debugger can access the status and control elements, implemented as registers on the DAP bus as shown in the figure below. These registers provide additional control and status for low power mode recovery and typical run-control scenarios. The status register bits also allow the debugger to get updated status of the core without having to initiate a bus transaction across the crossbar switch, thus remaining less intrusive during a debug session.

These DAP control and status registers are not memory mapped within the system memory map and can be only accessed via the DAP using SWD. The ADM-AP is accessible as Debug Access Port 1 with the available registers shown in the table below.

Table 41-6 ADM-AP register summary

Address	Register	Description
0x0100_0000	Status	See ADM-AP Status Register
0x0100_0004	Control	See ADM-AP Control Register
0x0100_00FC	IDR	Read-only Identification register that always reads as 0x0C86_0000

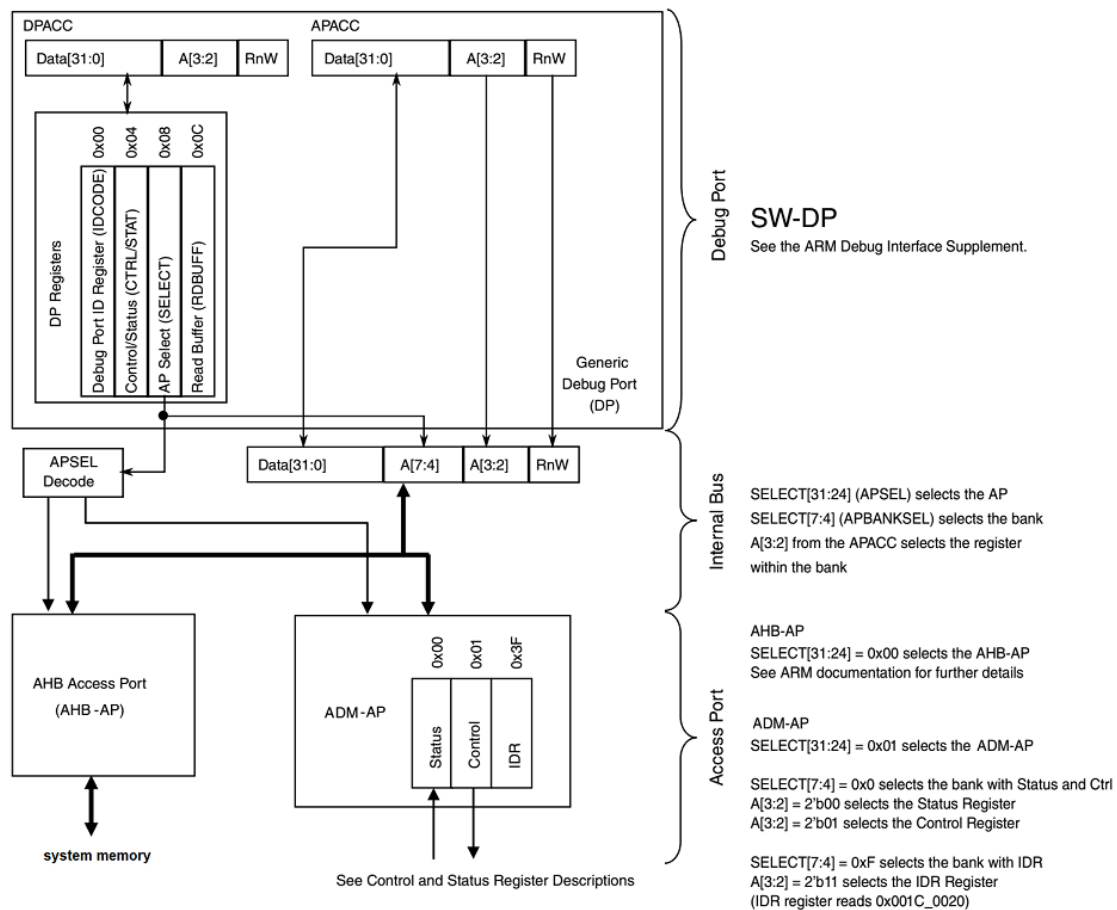


Figure 41-3 ADM-AP address

41.5.1 ADM-AP control register

Table 41-7 ADM-AP control register assignments

Bit	Name	Command available in secure mode	Description
0	Flash mass erase request	Yes	Set to cause mass erase. Cleared by hardware after mass erase operation completes.
1	Debug disable	No	Set to disable debug. Clear to allow debug operation.
2	Debug request	No	Set to force the core to halt.
3	System reset request	Yes	Set to force a system reset. The system remains held in reset until this bit is cleared.
4	Core hold reset	Yes	Configuration bit to control core operation at the end of system reset sequencing. 0b: Normal operation - Release the core from reset along with the rest of the system at the end of system reset sequencing. 1b: Suspend operation - Hold the core in reset at the end of reset sequencing. After the system enters this suspended

Bit	Name	Command available in secure mode	Description
			state, clearing this control bit immediately releases the core from reset and CPU operation begins.
5 – 31	Reserved for future use	No	

41.5.2 ADM-AP status register

Table 41-8 ADM-AP status register assignments

Bit	Name	Description
[0]	Flash mass erase acknowledge	This bit is cleared on the debug reset or at the launch of a mass erase command due to writing the Mass Erase in Progress bit in the ADM-AP Control Register. This bit is set after the flash control logic has started the mass erase operation.
[1]	Flash ready	Indicates flash memory has been initialized and ready to access
[2]	System security	Indicates the security state. When secure, the debugger does not have access to the system bus or any memory mapped peripherals. This bit indicates when the part is locked, and no system bus access is possible.
[3]	System reset	Indicates the system reset state. 0b: System is in reset 1b: System is not in reset
[4]	Flash busy	Indicates Flash is busy to execute some command. 0b: Flash is idle for read 1b: Flash is busy to execute some command. Reading to the Flash block is prohibited.
[5]	Debug erase all disable	Indicates debug erase all function is disabled. 0b: Debug erase all to Flash is allowed 1b: Debug erase all to Flash is disabled
[6-15]	Reserved	
[16]	Core halted	Indicates the core has entered the debug halt mode
[17]	WAIT	Indicates the core has entered the WAIT mode SLEEPING == 1 and SLEEPDEEP == 0
[18]	STOP	Indicates the core has entered the STOP mode SLEEPING==1 and SLEEPDEEP==1
[19-31]	Reserved	

41.6 Debug reset

The debug system receives the following sources of reset:

- System POR reset

Conversely, the debug system can generate system reset using the following mechanism:

- A system reset in the DAP control register that allows the debugger to hold the system in reset.
- Writing 1 to the SYSRESETREQ field in the NVIC Application Interrupt and Reset Control register.
- A system reset in the DAP control register that allows the debugger to hold the core in reset.

41.7 Debug in low-power modes

In low-power modes where the debug modules are kept static or powered off, the debugger cannot gather any debug data for the duration of the low-power mode.

- If the debugger is held static, the debug port returns to full functionality as soon as the low-power mode exits and the system returns to a state with active debug.
- If the debugger logic is powered off, the debugger is reset on recovery and must be reconfigured once the low-power mode is exited.

The active debug will prevent the chip from entering low-power mode. If the chip is already in low-power mode, a debug request from the ADM-AP control register will wake the chip from low-power mode.

41.8 Debug and security

When security is enabled (`FLASH_FSTAT[FSEC] = 1`), the debug port capabilities are limited to prevent exploitation of secure data. In the secure state, the debugger still can access the ADM-AP Status Register and determine the current security state of the device. In the case of a secure device, the debugger also is capable of performing a mass erase operation via writes to the ADM-AP Control Register.

Appendix A Release History

Z20K14xMC Reference Manual Release History

Table A-1 Z20K14xMC Reference Manual Release History

Rev. No	Release Date
0.1	9 October 2021
0.2	3 November 2021
0.3	22 February 2022
0.4	11 March 2022
0.6	25 April 2022
0.7	29 July 2022
0.8	20 April 2023
1.0	28 April 2023
1.1	5 September 2023
1.2	23 January 2024
1.3	18 November 2024

Appendix B Revision History

DATE	VERSION	DESCRIPTION
2023/08/14	1.0&1.05	correct grammar mistakes of the whole doc; modify Z20K14xM to Z20K14xMC; add instance and channel information of mcpwm, tim, stim, rtc, tdg, adc, cmp, uart, spi, i2c, i2s, can; Figure 2-1 delete mpu; delete instance number; modify flash; 2.3 update table 2-1 to add information of 144, 146; delete “83 GPIOs for 100 LQFP”; delete stop in I2C; modify CMP feature; 2.3&2.7 delete calibration function of ADC; 2.4 modify eei to eiru in figure 2-1; 2.5 add feature comparison; 2.7.2 update table 2-5; 2.7.6 update table 2-9; 2.7.8 update table 2-11; 3.2&3.3 update sram and flash mem map; 3.3 update table 3-2; 3.3.2 modify writeable to writable; 5.3.1 update table 5-2; 6.3.1 update reset values of SCM_FLSCFG, SCM_RAMCFG, SCM_DEVID; 6.3.2 update description of SCM_CHIPCTL[CLKOUTSEL]; 6.3.3 reserve SCM_MISCCTL1.CCACHEDIS_WB, CCACHE_LIM_WB, CCACHE_WBE_INT_EN; 6.3.5 reserve SCM_MISCSTAT1.CCACHE_WBE_FLAG; 6.3.11 update description of SCM_DEVID[SERIES_ID], [SUBFAMILY_ID], [FAMILY_ID]; 7.1.1 update features; 9 Update CPRI number in DMA_CPRI1, DMA_CPRI2, DMA_CPRI3; 9.3.2 update Figure 9-3 DMA major loop transaction sequence; 13.2.1&13.2.3&13.2.4 update offset of SMPU_Sn_EA and SMPU_Sn_ES; 13.2.2 update SMPU_CS; 13.3.2 add memory protect address; 13.3.3 update bus master; 13.3.4 add slave port; 13.3.6 update bus master number; 13.3.8 update the number of n; 14.2 update features; 14.3.3 update descriptions of WDOG_CS; 14.4.5.1 update Test each byte of the counter; 14.4.5.2 delete Enter into normal mode; 14.4.6 update Clock source;
		15.1.1 update features; 15.1.2 update wait mode; 15.2.1 update The EWDT_OUT signal; 15.3.3.3 update description of EWDT_CS[INPP];

DATE	VERSION	DESCRIPTION
		15.3.6.3 update description of EWDT_CNT[CNT]; 18 delete channel 7 for loss of PLL; 19.2.5 update description of TRNG_CTR_DRBG_REQUEST_SIZE[RS]; 21.2.1 update power-on reset; 21.2.2.2 update Low Voltage Detect (LVD); 21.2.2.4 update Loss-of-Clock (LOC) Reset; 21.2.2.8 update Wakeup Reset Request; 21.2.3.2 update Chip POR; 22.1 update Introduction; 22.2 update figure 22-1; 22.4 update Internal clocking requirements; 22.6 change table 22-4; 23.2.1 update scc block diagram; 23.3.9 update descriptions of SCC_SPLLC[SPLLRDY], [RSVD_13_12]; 23.3.11 update reset value of SCC_SPLLCFG2 and LDOODEN bit description; 23.4.3 update descriptions of SPLL clock mode; 25.2 update power modes description; 25.3.3 update description of PMU_LPO_32K_CFG; 25.3.3.3 update PMU_LPO_32K_CFG[LPO_CLK_SEL]; 25.3.5 update description of PMU_CSR[VDD_LVD_LP_EN]; 25.4.4 update Entry and exit sequencing of the power modes; 25.4.5 update ewdt in stop mode in table 25-4; 25.4.7.1 update LVD under low power mode; 26.5.4 update STANDBY mode; 27.1.2 update Features; 28.3.1.1 update clock source selection; 29.4.1.1 update clock source selection; 29.4.8.3 update descriptions of Complementary PWM output mode; 29.4.8.3 switch figure 29-21 and figure 29-22; 29.4.12.3 update CCV(n) register synchronization; 31.3.3 Update RTC_IER[2:0] and RTC_IER[3] descriptions; 31.3.2 update description of RTC_CSR[SW_RST];
		33.1 delete information about calibration in introduction and features; 33.1 Change ADC max rate from 1.1Msps to 1.0Msps in introduction; 33.3.1&33.3.11 delete ADC_CAL_VAL; 33.3.3.3, 33.3.4.3 delete information about calibration; 33.3.3 update description of ADC_CTRL[STABLE_T]; 33.3.3 update description of ADC_CTRL[STABLE_T]; 33.3.6 update description of ADC_CFG[STS]; 33.4.6 delete self-calibration mode; 34.1 & 34.2.3 Modify the number of external analog inputs; 34.2.1 update figure 34-1; 34.3.5 update CMP_CCR2; 34.3.4.3 modify the description of

DATE	VERSION	DESCRIPTION
		CMP_CCR1[FILT_CNT]; chapter 35 delete UART_DMA_SA; 35.3.1&35.3.9 modify full name of UART_LCR; 35.3.6.3 modify access of UART_IER[RSVD_6_4] from rw to ro; 35.3.11 update descriptions of of UART_LSR[HEADER_OP_DONE],[RSP_OP_DONE]; 35.3.9.3 Update description of UART_LCR[IDLE_DET_LENGTH]; 36.3.6.3 update SPI_FTLR; 36.3.10 update fields of SPI_ISR; 36.3.16.3 update SPI_DMATDLR; 36.3.17.3 update SPI_DMARDLR; 36.4.3 update clock config description; 36.4.4 update figure 36-2 and figure 36-3; update figure 38-3; 39.1.2 Update freeze mode; 39.3.63 Update access of CAN_FDCTRL[TDCOFF]; 39.3.69 modify description of Table 39-14 RX filter ID table structure; 39.4.18 update CLOCK domain and restrictions; 40.8 update Debug and security
2023/08/17	1.0.6	2.2 modify 128K D-Flash to 128KB D-Flash; 2.3 update WIC, EWDT, Unique identification (ID) number, Flash read buffer, PMU, ADC, I2S in table 2-1; 2.4 update product block diagram; 2.7 update system, security in Table 2-3; 2.7.2 update GPIO in table 2-5; 2.7.3 update flash and regfile in table 2-6; 2.7.4 update pmu in table 2-7; 2.7.1&5.1 delete cortex M1; 3 add cmu0 in Table 3-2 Peripheral space assignment and delete register file; 4.1 modify PORT_PCR to PORTx_PCRn; 4.3 update truth table; 5.1.2 update system tick timer; 7.1.1 update features; 7.3.1 update offset; 7.4.1 update pin control; 7.4.3 update global interrupt status; 8.1 update introduction; 8.3.1.2 update GPIO Memory map; 8.4.1 update general-purpose input; 9.2.6 update DMA_ES; 9.2.10.3 update description of DMA_CPRI0[CSE3]; 10.2 update DMAMUX in DMA_MUX_CHn_CFG[SRC_MUX]; 11.4 update table 11-3 and 11-4; 12.2.4.3 update AES_CONFIG[BS]; 12.2.8.3 update description of AES_DATA_I3[DATA_I]; 12.3 update function description; 12.3.6 update GCM; 13.2.2 update SMPU_CS; 14.3.3 update WDOG_CS[INTF] access; 14.4 update functional description;

DATE	VERSION	DESCRIPTION
		17.2.4 update EIRU_SRAMn_EESR[EFSIZE], [EFMST]; 18.2.9 update descriptions of SERU_CFG_CHn[CH_RESET_DLY]; 19.2.17 update TRNG_ADDITIONAL_MEAS_CONFIG[SEL]; 19.3.1&19.3.2 update the enable operation of normal and stream modes; 20.3.1 update CRC initialization/reinitialization;
		21.1 update introduction; 21.2.2.1 update external pin reset; 21.2.2.7 update ADM-AP System Reset Request; 21.2.4 update reset pin; 21.2.5 delete JTAG reset; 22.2 update clock diagram; 22.4 update Internal clocking requirements; 22.6 update table 22-4; 23.3 update reset values of SCC_PARAM, SCC_OSCCFG, SCC_SPLLCS, SCC_SPLLCFG1, and SCC_SPLLCFG2; 23.2 delete LOL; 23.4.5 update figure; 25.3.2 update PMU_LOCK_USER[LOCK]; 25.3.3 update PMU_LPO_32K_CFG[LPO_CLK_READY]; 25.4.4 update entry sequence of the power modes; 25.4.5 update LVD/LVR, REGFILE module operation in power modes; 26.2 update reset values of SRMC_PARAM, SRMC_DSMWUPC2, SRMC_DSMWUPS; 26.3.6 update SCC loss-of-clock (LOC); 26.3.7 delete SCC loss-of-lock (LOL); 28 add regfile; 29.2.6 update descriptions of MCPWM_GLBCCR[NTEN0], [CNTEN1], [CNTEN2],[CNTEN3]; 29.2.8 update description of MCPWM_CFGn[CPP]; 29.2.13 update description of MCPWM_PCR[ASYMEN0], [ASYMEN1],[ASYMEN2], [ASYMEN3]; 29.2.15 update access of MCPWM_FLTSR[FAULTF0], [FAULTF1], [FAULTF2], [FAULTF3], [FAULTFA], [FAULTFB]; 31.1 update introduction; 31.1.1 update features; 31.1.2 update stim block diagram; 32.3.2.2 update function of RTC_CSR; 32.3.3.2 update function of RTC_IER; 33.1.2 update figure 33-1 tdg block diagram; 33.3.4 update interrupt in tdg; 34.2 update Signal description of ADC; 34.4.1 update ADC voltage reference select; 35.1 update introduction; 35.3.6 modify the description of CMP_DCR[DAC_BUFF_EN];

DATE	VERSION	DESCRIPTION
		37 modify TX to transmit; update signal description; update SPI_CTRLR0 [DFS_32] description; update SPI_CTRLR1[NDF] description; update SPI_SSENr[SER] description; update SPI_BAUDR, SPI_ICR and SPI_FTLR register description; update description of Transfer formats; update description of DMA and interrupt; update description of Initialization application information; modify 4 to 6 in Clock config; 37.3.4 update SPI_SSENr[SER]; 37.3.6 update SPI_FTLR[TFT], [RFT]; 38.4.6 update I2C SCL configuration; 40.1.2 update Module Disable mode; 41.2 update Cortex-M4F debug topology figure and JTAG IR code table; 41.4 update ADM-AP register summary table;
2023/09/05	1.0.7	5.1.3 update debug facilities; 5.2.3 update Interrupt vector assignment; 21.1 update introduction; 21.2.2.5 add Low power mode acknowledge error (SACKERR); 21.2.2.7 modify title to ADM-AP System Reset; 21.2.2.9 add SERU normal reset; 21.2.2.10 add SERU cold reset; 21.2.2.8 modify title and sequence number to 21.2.2.11 Wake up Reset from standby mode; 26.3.7 update Low power mode acknowledge error (SACKERR); 26.3.10, 26.3.11, 26.3.12 delete request of the title; 26.3.11 update SERU normal reset request; 26.3.12 update SERU cold reset request; 26.2.3 update description of SRMC_SRS[SACKERR]; 26.2.5 update description of SRMC_SSRS[SSACKERR]; 26.2.8 update description of SRMC_PMCTRL[DSMACKTMO]; 26.2.9 delete "PMSTAT" in SRMC_PMSTAT[PMSTAT] "PMSTAT Power Mode Status" and update description of SRMC_PMSTAT[DSMABORT]; 26.2.10.2 update function; 26.2.10.3 update descriptions of SRMC_DSMWUPC1[WUPSRC_EN2_0], [WUPSRC_EN15_4]; 26.2.11.2 update function; 26.2.12.2 update function; 26.2.12.3 update descriptions of SRMC_DSMWUPS[WUPS2_0], [WUPS16_4]; 26.2.1 update reset values of SRMC_SRS, SRMC_SSRS, SRMC_PMCTRL;
2023/09/05	1.1	Release
2024/01/16	1.1.1	whole doc uppercase run, stop, wait, standby; 2.5 update package of 148MC; 6.3 update Unique Identification Register; 6.3.3 update description of SCM_MISCCTL1 [SWTRIGN], update description of

DATE	VERSION	DESCRIPTION
		SCM_MISCCTL1[CCACHE_PARITY_MISS_EN], update description of SCM_MISCCTL1 [CCACHE_CLR]; 6.3.5 update SCM_MISCSTAT1; 6.3.8 update SCM_CHIPOPT; 6.3.9 update reset value of SCM_FLSCFG [RSVD_23_0]; 6.3.10 update reset value of SCM_RAMCFG [RSVD_27_4]; 6.3.11 update description of SCM_DEVID; 6.3.12 update function of SCM_UNIQUEn, update field name of SCM_UNIQUEn[UID_31_0] to [UID]; 11.3.1&11.3.2 update reset value of TMU_CFGn; 12.1.1 update features; 12.2.4 update description of AES_CONFIG[BS]; 14.2&14.4.3 modify the time interval: reset delay time; 14.3.3 update description of WDOG_CS[CFGUF]; 16.2.2 update access of CMU_PARAM[RST_IMP]; 16.2.3.3 update description of CMU_CTRL[LOCK]; 16.3.1 update Clock monitor mode; 16.3.2 update Frequency measure mode; 16.3.3 add note to Monitored clock and reference clock; 17.2.2 modify SRAM0 to SRAML and SRAM1 to SRAMU in EIRU_CR; 18.1 update introduction; 18.1.1 modify EOUT parity to EOUT polarity; 18.1.1.1 delete channel 15, update description of error channels 9, 11, 13, 14 mapping information, add note; 18.1.2.2 update STOP mode; 18.2 update offset of SERU_CFG_CHn; 18.2.6 update SERU_CH_FFI; 18.2.8 update SERU_CH_FLAG; 18.3.2 update Reset requests; 19.2.1&19.2.2 update reset value of TRNG_STATUS; 19.2.1&19.2.9 update reset value of TRNG_CTR_DRBG_ALARM; 19.2.1&19.2.18 update reset value of TRNG_ADDITIONAL_MEAS_STATUS; 19.2.3 update access of TRNG_CONFIG[RSVD]; 19.2.4 update access of TRNG_CTR_DRBG_CONTROL [RSVD]; 19.2.10 update access of TRNG_RAW_RANDOM_CONTROL[RSVD]; 19.2.13 update access of TRNG_ADDITIONAL_CONFIG_0[RSVD]; 19.2.14 update access of TRNG_ADDITIONAL_CONFIG_1[RSVD]; 19.2.16 update access of TRNG_ADDITIONAL_MEAS_CONTROL[RSVD]; 19.2.22 update access of TRNG_IRQ_CLEAR[RSVD]; 19.2.23 update access of

DATE	VERSION	DESCRIPTION
		TRNG_IRQ_CONFIG[RSVD]; 19.2.25 update access of TRNG_ERROR_CONTROL[RSVD]; 19.2.27 update access of TRNG_STATS_CONFIG[RSVD]; 19.2.28 update access of TRNG_STATS_CONTROL[RSVD]; 21.2.2.4 update Loss-of-Clock (LOC) Reset; 23.2.1 update SCC block diagram; 23.3.6 update access of SCC_OSCCS[OSCLOC], update description of SCC_OSCCS[OSCCME]; 23.3.7 update description of SCC_OSCCFG[OLMEN]; 23.3.8 update description of SCC_FIRCCS[FIRCLOC]; update access of SCC_FIRCCS[FIRCLOC]; 23.3.10 update description of SCC_SPLLCFG1[PREDIV]; 23.3.11 update description of SCC_SPLLCFG2[LDOODEN]; 23.4.4 update PLL frequency; 23.4.5 update figure in System clock source; 24.2.1 modify register names of PARCC_TIMERn to PARCC_TIMn; 24.2.2 update description of PARCC_WDOG[PRSTB]; 25.2 update Table 25-1; 25.3.1&25.3.2 update register name to PMU_LOCK; 25.3.1&25.3.3 update reset value of PMU_LPO_32K_CFG; 25.3.3 update description of PMU_LPO_32K_CFG[LPO_CLK_DIS]; 25.3.4 update function of PMU_ISO_CLR and update description of PMU_ISO_CLR[ISO_CLR]; 25.3.5 update description of PMU_CSR[VDD_LVD_LP_EN], [VDD_LVD_ACT_EN], [VDD_LVD_RE], [REF_BUF_1V_EN]; 25.3.6 update description of PMU_LDO_EN_CFG[LDO_CORE_EN], [LDO_CLOCK_EN], [LDO_FLASH_EN]; 25.3.8 update description of PMU_LDO_LVD_CFG [LVD_CORE_RE], [LVD_CLOCK_RE], [LVD_FLASH_RE], [LVD_AOA_RE], [ACT_CORE_LVDE], [ACT_CLOCK_LVDE], [ACT_AOA_LVDE], [LP_CORE_LVDE], [LP_CLOCK_LVDE], [LP_AOA_LVDE]; 25.4.4 update Entry and exit sequencing of the power modes; 26.2.3 update function of SRMC_SRS and update description of SRMC_SRS[LOC]; 26.2.4 update description of SRMC_CTRL[RSTFLTSS]; 26.2.5 update function of SRMC_SSRS; 26.2.7 modify bit name of SRMC_PMPORT[ASTB] to APD; 26.2.6 update access of SRMC_SRIE[SERU_COLD],[SERU_SYS]; 26.2.8 update description of SRMC_PMCTRL[DSMC],

DATE	VERSION	DESCRIPTION
		update description of SRMC_PMCTRL [DSMACKTMO]; 26.3.1 add note to Power-on reset (POR); 26.5.3 update STOP mode; 26.5.4 update STANDBY mode; 26.6 add Reset and interrupt; 27.3.2 update descriptions of FLASH_FSTAT[DFDIF] and [SFDIF]; 27.3.4 update description of FLASH_FCTRL[RWSC]; 29.2.7 update access and description of MCPWM_GLSR[RLDF0], [RLDF1], [RLDF2], [RLDF3], [TOF0], [TOF1], [TOF2], [TOF3]; 29.2.8 update access of MCPWM_CFGn[CHF]; 29.2.10 update descriptions of MCPWM_OUTCR[POL0], [POL1], [POL2], [POL3], [POL4], [POL5], [POL6], [POL7]; 29.2.11 update access of MCPWM_OUTTRIG[TRIGF]; 29.2.21 update MCPWM_MODn_DITHER[MOD_DITHER]; 29.2.22 update MCPWM_CVn_DITHER[CV_DITHER]; 29.3.7 update figures 29-12 and 29-13; 29.3.8.2 add note to MOD(n) and MOD(n)_DITHER register synchronization; 29.3.8.3 add note to CV(n) and CV(n)_DITHER register synchronization; 29.3.8.4 delete SYNC[SWRSTCNT] = 1; 29.3.9.1 update PWM period dithering; 29.3.9.2 update PWM edge dithering; 30.1 update Introduction; 30.2 update Signal description; 30.3.4 update function of TIM_CNTINIT; 30.3.7 update access of TIM_GLSR[RLDF], [TOF]; 30.3.8 update access of TIM_CMCn[CHF]; 30.3.10 update TIM_OUTCR; 30.3.11 update TIM_OUTSWCR; 30.3.14 update description of TIM_FLTCR [FIFVAL]; 30.3.15 update access of TIM_FLTSR [FAULTF0], [FAULTF1], [FAULTF]; 30.4 update figures 30-2, 30-3, 30-4, 30-6, 30-7, 30-8, 30-9, 30-10, 30-11, 30-12, 30-13, 30-14, 30-15, 30-16, 30-17, 30-18, 30-19, 30-20, 30-21, 30-22, 30-23, 30-26, 30-27, 30-28; 30.4.1.1 update Clock source selection; 30.4.3.2 update Channels output initialization 30.4.4 update Input capture mode; 30.4.8.1 update Center-aligned PWM output mode; 30.4.8.4 update Deadtime insertion for complementary PWM mode; 30.4.9 update Generation of match trigger and initial trigger; 30.4.10 update Channel output software control; 30.4.11 update Fault input control;

DATE	VERSION	DESCRIPTION
		31.1.2 update STIM block diagram; 32.1.2 update RTC block diagram; 32.3.2 update description of RTC_CSR[SW_RST]; 32.3.7 update description of RTC_COMPENSATIONR [DLY]; 32.4.4 Periodic interrupt; 34.3.3 update description of ADC_CTRL[IREF_SEL]; 34.3.4 update description of ADC_STAT[DMA_REQ], update access and description of ADC_STAT [TC_ERR]; 34.3.6 update description of ADC_CFG[REF_SEL]; 36.3.1&36.3.20 update register name of LIN PID Register(UART_LIN_PID_VALUE) to LIN ID Register (UART_LIN_ID_VALUE); 36.3.12 update description of UART_MSR[CTS]; 36.4.1 add Transmit Programming; 36.4.2 add Receive Programming; 37.3.1&37.3.9 update reset value of SPI_IER; update full name of SPI_IER and update the description of SPI_IER register; 37.3.3 update description of SPI_CTRLR1[NDF]; 38.3.1&38.3.17 update reset value of I2C_HS_SCL_LCNT; 38.3.3 delete function of I2C_CONFIG0 and update description of I2C_CONFIG0[H_MCODE]; 38.3.24 update description of I2C_STATUS0[TXFIFO_EMPTY]; 38.3.26 update I2C_ERROR_STATUS; 39.2 update register name of I2S_CHANNELn_LEFT_DATA, I2S_CHANNELn_RIGHT_DATA, I2S_CHANNELn_CFG, I2S_CHANNELn_INT_CFG, I2S_CHANNELn_TRIGGER_LEVEL_CFG to I2S_CHn_LEFT_DATA, I2S_CHn_RIGHT_DATA, I2S_CHn_CFG, I2S_CHn_INT_CFG, I2S_CHn_TRIG_LEVEL_CFG in the titles; 39.2.7 update access of I2S_CHn_INT_CFG [RXFIFO_ORF] and [TXFIFO_ORF]; 40.1.2 update Modes of operation; 40.3.1 update Table 40-3 CAN memory map; 40.3.2 update description of CAN_MCR[DMAE], [IRMQ], [SUPV], [RFEN]; 40.3.3 update function of CAN_CTRL1; 40.3.4 update function of CAN_TIMER and update description of CAN_TIMER[TIMER]; 40.3.5 update function of CAN_RXMGMSK and update description of CAN_RXMGMSK[MG]; 40.3.6 update function of CAN_RX14MASK and update description of RX14M; 40.3.7 update function of CAN_RX15MASK and update description of RX15M; 40.3.8 update function of CAN_ECR; 40.3.9 add function of CAN_ESR1 and update description of CAN_ESR1[WAKINT], [RXWRN], [TXWRN]; 40.3.10 add function of CAN_IMASK2;

DATE	VERSION	DESCRIPTION
		40.3.11 add function of CAN_IMASK1; 40.3.12 add function of CAN_IFLAG2 and update description of CAN_IFLAG2[BUF63TO32I]; 40.3.13 add function of CAN_IFLAG1 and update description of CAN_IFLAG1[BUF0I], [BUF4TO1I], [BUF5I], [BUF7I]; 40.3.1&40.3.14 update reset value of CAN_CTRL2; 40.3.14 add function of CAN_CTRL2 and update description of CAN_CTRL2[ECRWRE], [BOFFDONEMSK], [ERRMASK_FAST]; 40.3.15 add function of CAN_ESR2 and update description of CAN_ESR2[VPS]; 40.3.16 add function of CAN_CRCCR; 40.3.17 update description of CAN_RXFGMASK [FGM]; 40.3.18 add function of CAN_RXFIR; 40.3.19 add function of CAN_CBT; 40.3.20 add function of CAN_IMASK4; 40.3.21 add function of CAN_IMASK3; 40.3.22 add function of CAN_IFLAG4; 40.3.23 add function of CAN_IFLAG3 and update description of CAN_IFLAG3[BUF95TO64I]; 40.3.24 update name and offset of CAN_MBn_CS; 40.3.25 update name and offset of CAN_MBn_ID and update description of PRIO; 40.3.26 update name and offset of CAN_MBn_DATA03; 40.3.27 update name and offset of CAN_MBn_DATA47; 40.3.28 update offset of CAN_RXIMRn, add function, and update description of CAN_RXIMRn[MI]; 40.3.29 add function of CAN_MECCR; 40.3.30 add function of CAN_ERRIAR and update access of CAN_ERRIAR[INJADDR_L] and [RSVD_31_14]; 40.3.31 add function of CAN_ERRIDPR; 40.3.32 add function of CAN_ERRIPPR; 40.3.33 update function of CAN_RERRAR; 40.3.34 add function of CAN_RERRDR; 40.3.35 add function of CAN_RERRSYNR; 40.3.36 add function of CAN_ERRSR; 40.3.37 update function of CAN_CTRL1_PN; 40.3.38 add function of CAN_CTRL2_PN; 40.3.39 update function of CAN_WU_MTC; 40.3.40 update function of CAN_FLT_ID1; 40.3.41 update function of CAN_FLT_DLC and add access of CAN_FLT_DLC[FLT_DLC_HI]; 40.3.42 update function of CAN_PL1_LO; 40.3.43 update function of CAN_PL1_HI; 40.3.44 update function of CAN_FLT_ID2_IDMASK; 40.3.45 update function of CAN_PL2_PLMASK_LO; 40.3.46 add function of CAN_PL2_PLMASK_HI;

DATE	VERSION	DESCRIPTION
		40.3.47 update offset and add function of CAN_WMBn_CS; 40.3.48 update offset and add function of CAN_WMBn_ID; 40.3.49 update offset and add function of CAN_WMBn_D03; 40.3.50 update offset and add function of CAN_WMBn_D47; 40.3.51-40.3.62 delete CAN_WMB1_CS, CAN_WMB1_ID, CAN_WMB1_D03, CAN_WMB1_D47, CAN_WMB2_CS, CAN_WMB2_ID, CAN_WMB2_D03, CAN_WMB2_D47, CAN_WMB3_CS, CAN_WMB3_ID, CAN_WMB3_D03, CAN_WMB3_D47; 40.3.51 add function of CAN_FDCTRL; 40.3.52 add function of CAN_FDCBT; 40.3.53 add function of CAN_FDCRC; 40.3.54 add superscript in table 40-5; 40.4.1 update Transmit process; 40.4.4 modify CAN_MCR[DMA] to CAN_MCR[DMAE]; 40.4.5 update Match process; 40.4.8.1 modify CAN_MCR[DMA] to CAN_MCR[DMAE]; 40.4.8.2 modify CAN_MCR[DMA] to CAN_MCR[DMAE]; 40.4.9.2 modify CAN_FDCTRL[FDRATE] to CAN_FDCTRL[FD_RATE], RXERRCNT_FAST and TXERRCNT_FAST to RX_ERRCNT_FAST and TX_ERRCNT_FAST, BITERR1_FAST, BITERR0_FAST, CRCERR_FAST, FRMERR_FAST to BIT1ERR_FAST, BIT0ERR_FAST, CRCERR_FAST, FRAMERR_FAST; 40.4.11 delete contents related to ERFEN; 40.4.13 update Message buffer time stamp; 40.4.14 delete 40.4.14 High resolution time stamp; 40.4.14 modify FDPROPSEG, FDPSEG1, FDPSEG2 to FPROPSEG, FPSEG1, FPSEG2; 40.4.17 add Error injection; 40.4.19 modify CAN_MCR[DMA] to CAN_MCR[DMAE];
2024/01/23	1.1.2	18.1.1 modify the number of error channels; 18.1.1.1 update detailed description of channels 9 and 10; 26.3.1 update Power-on reset (POR); 29.1.1, 29.2.2 update system clock to core clock; 29.2.6 update access of MCPWM_GLBCCR[CNTE_N]; 29.2.13 update access of MCPWM_PCR[CVSEL_n], [PEC_n], [ASYMEN_n]; 30.1.1, 30.3.2, 30.4, 30.4.4, 30.4.7, 30.4.8.4, 30.4.9 update system clock to core clock; 38.3.4 update description of I2C_CONFIG1 [TXFIFO_EMPTY_EN]; 38.3.12 update description of I2C_STD_SCL_HCNT[STD_SCL_HCNT]; 38.3.13 update description of

DATE	VERSION	DESCRIPTION
		I2C_STD_SCL_LCNT[STD_SCL_LCNT]; 38.3.14 update description of I2C_FST_SCL_HCNT[FST_SCL_HCNT]; 38.3.15 update description of I2C_FST_SCL_LCNT[FST_SCL_LCNT]; 38.3.16 update description of I2C_HS_SCL_HCNT[HS_SCL_HCNT]; 38.3.17 update description of I2C_HS_SCL_LCNT[HS_SCL_LCNT];
2024/01/23	1.2	Release
2024/11/18	1.2.1	1.5.3 update access types in the table; 2.3 update RTC; 5.1.4 add cache address; 5.2.2 add description of NMI; 5.2.3 add column of Type in table 5-1; 6.3.3 update description of SCM_MISCCTL1[SWTRIGN], [CCACHE_CLR], [CCACHE_PARITY_MISS_EN], [CCACHE_PARITY_FAULT_EN], [CACHE_DIS], [FPU_INVALIDOP_INT_EN], [FPU_DIVZERO_INT_EN], [FPU_DENORMALIN_INT_EN]; 6.3.5 add CACHE_BE_FLAG, CACHE_BE_SUPV, CACHE_BE_HSIZE, CACHE_BE_HMAST; 6.3.10 update description of SCM_RAMCFG[CACHESIZE]; 7.1.1 add support NMI interrupt; 7.3 update full register name of PORTx_IRQFLG; 7.3.4 update function, add note in description of PORTx_PCRn[PS] and [PE], delete UNLOCK_KEY in description of PORTx_PCRn[LK], update access of PORTx_PCRn[ISF]; 7.3.5 update access of PORTx_IRQFLG[ISF]; 7.4.4 add NMI pin; 7.4.5 modify filter width register configuration to PORTx_PCRn[FLT]; 9.2.3 add REQDIS function impact in description of fields in DMA_DMAE; 9.2.6 modify source to destination in 0b description of DMA_ES[DBEF]; 9.2.7 update access and description of fields in DMA_CES; 9.2.8 update access and description of fields in DMA_CIS; 9.3.5 add note; 10.1.1 update the number of input sources; 10.2.2 add 0b description to DMA_MUX_CHn_CFG[ENABLE], update input sources (update the number of input sources, delete I2S dma request, delete MCPWM1 channel 7 dma request, delete ADC dma request, delete CMP dma request, modify DMAMUX always enabled on channel 0/1 to DMAMUX always-on channel 0/1) in description of DMA_MUX_CHn_CFG[SRC_MUX]; 11.4.1 rename MCPWM EXT trigger;

DATE	VERSION	DESCRIPTION
		12.2.3-12.2.8, 12.2.17-12.2.20 update function; 12.2.4 modify CMAC_FINISH_K to CMAC_FINISH_K1, CMAC_FINISH_K2 in description of AES_CONFIG[BS]; 13.1.1 add SMPU slave port assignments; 13.1.3 update SMPU block diagram; 13.3.7 delete figure and the related specific AHB bus signals; 14.2.2 update bus clock delay in WDOG block diagram; 15.3.3 update description of EWDT_CS[INPE], [INPP]; 15.4.4 modify EWM_OUT to EWDT_OUT; 15.4.6 add EWDT_CS[INTF] and delete table; 16.1.2 update block diagram; 16.3.1 add threshold calculation; 17.2.1 modify EIRU_CHn_DATAMASK to EIRU_SRAMn_DATAMASK; 17.2.7 & 17.2.8 update function; 18.1.1.1 update error description of channel 4, and update note; 18.3.5 update description of normal reset process; 19.2.4, 19.2.10, 19.2.16, 19.2.22, 19.2.25, 19.2.28 update function; 19.3.1.1 add step 11; 20.2.4 modify field name of bit 0 from mode to MODE; 20.3.3 add CRC_DATA[31:16] in steps 7 and 8, add note; 20.4 modify [mode] to [MODE]; 21.1 reclassify reset sources; 21.2 add figure; 21.2.1 update description of Power-on reset (POR); 21.2.3 delete MCU Resets; 21.2.5 modify until flash memory initialization has completed to until flash memory initialization starts; 22.6 delete flash memory, add note in comments of flash controller, update comments of I2S in table 22-4; 23.3 add the SCC registers can be written only in supervisor mode; 23.3.4 add note in description of SCC_CFG[SCS]; 23.3.6 update description of SCC_OSCCS[OSCCMRE]; 23.3.7 update description of SCC_OSCCFG[ITRIM]; 23.3.8 update description of SCC_FIRCCS[FIRCCMRE], update access of bit 12; 23.3.9 update access of bit 2 and bit 10 in SCC_SPLLC; 23.4.4 update SPLLC frequency calculation formula; 23.4.5 update figures; 24.1.2 update PARCC block diagram to delete PARCC_CFG; 24.2 add the PARCC registers can be written only in supervisor mode; 24.2 update reset value of PARCC_FLASH; 24.2.13-24.2.20, 24.2.41-24.2.42, 24.2.50-24.2.51 update description of PRSTB; 24.2.41 update access of bit 16 and bit 20 in PARCC_FLASH;

DATE	VERSION	DESCRIPTION
		24.3.3 modify RSTB to PRSTB; 24.3.4 modify PCLKMD to PCKMD; 24.3.5 add “This register field is writable only when PARCC[PRSTB] or PARCC[PCKMD] is 0”; 25.2 update description and normal recovery method in STANDBY mode; 25.3 add the PMU registers can be written only in supervisor mode; 25.3.2 update 0b and 1b description of PMU_LOCK[LOCK]; 25.3.4 update function; 25.4.1 delete Entering and exiting the power modes; 25.4.1 modify PLD to PCKMD; 25.4.4 delete entry and exit sequencing of the power modes; 25.4.3 update UART/LIN functionality in STOP mode in table 25-4; 25.4.5.1 modify Vld_vdd5v to LVW_VDD5V; 25.4.5.2 modify Vlw_vdd5v to LVW_VDD5V; 26.2.4-26.2.8, 26.2.10-26.2.12 add the register can be written only in supervisor mode in function; 26.2.4 update description of SRMC_CTRL[RSTFLTSS]; 26.2.11 update reset value of SRMC_DSMWUPC2[RSVD_31_1]; 26.2.12 modify field names from WUPS2_0 and WUPS16_4 to WUPS_2_0 and WUPS_16_4, add wake-up sources in description of fields; 26.3.1-26.3.2 delete repetitive contents with chapter 21 reset and boot; 26.3.1 add reset generate; 26.3.2.3 update wakeup sources to delete I2C and add STIM; 26.3.2.4 add description of PTA5; 26.3.2.5 modify section title from power mode to power mode transitions; 26.3.2.5 add enter low power mode step description; 26.4 delete MCU reset signals; 27.1.2 add note; 27.3.2 update description of FLASH_FSTAT[FAIL]; 27.3.4 reserve bit 8, update description of FLASH_FCTRL[RWSC]; 27.4.3 add note; 27.4.4 modify Key[96:64] to Key[95:64] in table; 27.4.5 modify XNOR to XOR in step 3; 27.4.7 modify FCMD to FLASH_FCMD[CMDCODE]; 29.2.6 add write protection in description of MCPWM_GLBCCR[GLBCNTEN]; 29.3 add system clock represents core clock; 29.3.1.1 update note; 30.1 modify active state or tri-state to active state, modify inactive state to inactive state or tri-state; 30.3.7 update access of TIM_GLBSTR[CHnF]; 30.3.10 add note in description of TIM_OUTCR[INITRIGE]; 30.3.17 update description of TIM_RELOAD[HCSEL];

DATE	VERSION	DESCRIPTION
		30.4.2.2 modify chapter of register update to registers synchronization; 30.4.3.1 update table 30-4 (delete the first row, add ELS[1:0] =00 to CMS[1:0] =00 and CMS[1:0]=01); 30.4.3.2 add note; 30.4.5.2 delete warning and add note; 30.4.8.3 update figures 30-21 and 30-22; 30.4.10 modify inactive state to forced to inactive state for Channel(n+1) output when CH(n)OCV =1 and CH(n+1)OCV =x in table 30-9, update note; 30.5 delete application information; 32.3 add write access information; 32.3.2 modify RTC output clock to pin output enable to RTC output clock selection in description of RTC_CSR[CLKOUT_SEL]; 33.2.3 add 11b description of TDG_CTRL1[UPMD]; 33.2.3 update description of TDG_CTRL1[CLRMD], [CFGUP]; 33.2.17 modify CDO to complete delay output, DOC to delay output complete in description of TDG_CHnCDOINTDLY[LDCO]; 33.3.3 add note; 34.3.3 update description of ADC_CTRL[RST]; 34.3.4 update 0b and 1b description of ADC_STAT[FOF]; 34.3.6 update description of ADC_CFG[REF_SEL]; 34.4.1 update figure; 34.4.7 add conversion time; 36.3.6 update description of UART_IER[PTIME]; 36.3.7 add 01b/10b in description of UART_IIR[FIFOSE]; 36.3.8 add note in description of UART_FCR[TET]; 36.3.9 add note in description of UART_LCR[LBKM]; 36.3.9 update description of UART_LCR[LBKEN], [BC]; 36.3.9 add note in description of UART_LCR[IDLE_DET_LENGTH]; 36.3.10 update access of bit 6; 36.3.11 update function; 36.4.7 add Busy detect interrupt; 36.5.2&36.5.3 modify UART_LIN_PID_VALUE to UART_LIN_ID_VALUE; 37.3.3 add note in description of SPI_CTRLR1[NDF]; 37.3.9 modify mask to disable in function; 37.3.10 update description of all fields; 37.3.16 modify wdata to the data written to in description of SPI_DMATDLR[DMATDL]; 37.3.17 modify wdata to the data written to in description of SPI_DMARDLR[DMARDL]; 37.4 modify CTRL/CTRL0 to SPI_CTRLR0; 37.4.1.2 add note; 37.5 modify TXFTLR and RXFTLR to SPI_FTLR; 38.1.2 update high-speed mode; 38.3 update full register names of I2C_DEST_ADDR, I2C_FSTD_SPKCNT, I2C_HS_SPKCNT, I2C_ERROR_STATUS;

DATE	VERSION	DESCRIPTION
		38.3.3 update description of I2C_CONFIG0[SLAVE_ACK_GENCALL], [SLAVE_NACK], [H_MCODE]; 38.3.4 modify high speed mode in description of I2C_CONFIG1[SPEED_SEL]; 38.3.6 update function and description of all fields; 38.3.8 modify overrun to overflow in description of I2C_INT_ENABLE[RX_OVF_IE]; 38.3.9 update description of I2C_COMMAND_DATA[CMD_STOP]; 38.3.17 update reset value of I2C_HS_SCL_LCNT[HS_SCL_LCNT]; 38.3.22 update function; 38.3.25 add receiver in description of I2C_STATUS1[SLAVE_IS_DISABLED_UNDER_ACT]; 38.3.27 add function; 38.3.28 add function; 39.2 modify I2S_CHn_TRIG_LEVEL_CFG to I2S_CHn_TRIGGER_LEVEL_CFG; 39.2.3 update description of I2S_MASTER_CONFIG[I2S_NUM_BCLK], [I2S_NUM_BCLK_GATE]; 39.2.6 update 000b description of I2S_CHn_CFG[TX_AUDIO_RESOLUTION] and [RX_AUDIO_RESOLUTION]; 39.3.3.1 add I2S_RE; 39.3.4.1 add I2S_TE; 40.1.1 delete debug registers; 40.3.2 update function, update 0b description of CAN_MCR[SUPV], update table name in description of [RFEN]; 40.3.5 modify MG[31:0] to CAN_MG[31:0] in description of CAN_RXMGMSK[MG]; 40.3.7 modify CAN_RXMGMSK to CAN_RXMGMSK in description of CAN_RX15MASK[RX15M]; 40.3.9 update access of CAN_ESR1[RWRNINT], [TWRNINT], modify FD_EN to CAN_MCR[FDEN] in description of [BIT0ERR_FAST], [BIT1ERR_FAST]; 40.3.12 update access of CAN_IFLAG2[BUF63TO32I]; 40.3.24 modify DL to EDL in description of CAN_MBn_CS[EDL]; 40.3.25 add the least significant bit number in description of CAN_MBn_ID[ID_STD_EXT]; 40.3.28 modify CAN_RXFGMASK register to Legacy Rx FIFO structure in description of CAN_RXIMRn[MI]; 40.3.50 update function; 40.3.55 add Non-FD to all table captions; 40.3.56 modify MBDSR0 to MBDSR1/2/3 in RAM block 1/2/3 in table 40-12; 40.3.57 update offset in table header of table 40-13; 40.4.9.2 update figure caption of figure 40-2, update example about maximum configurable bit rate in the CAN FD data phase; 40.4.17 add detection and correction of memory errors; 40.4.18 add notice of PE domain, delete the last five rows from the bottom of table 40-24, update table

DATE	VERSION	DESCRIPTION
		caption of table 40-26; 40.4.19 modify the bit number of CAN_IFLAG1;
2024/11/18	1.3	Release

How to Reach Us:

Home Page:

zhixin-semi.com/

Information in this document is provided solely to enable system and software implementers to use Zhixin products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. Zhixin reserves the right to make changes without further notice to any products herein.

Zhixin makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Zhixin assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Zhixin data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer's technical experts. Zhixin does not convey any license under its patent rights nor the rights of others. Zhixin sells products pursuant to standard terms and conditions of sale, which can be found at the following address: <http://www.zhixin-semi.com/>.

Zhixin and the Zhixin logo are trademarks of Zhixin Semiconductor Co., Ltd. All rights reserved.

©2024 Zhixin Semiconductor Co., Ltd.

Document Number: Z20K14xMC-RM
Rev. 1.3, November, 2024

