

MA4823 Microprocessor

Systems Notes

Hankertrix

Contents

1 Abbreviations	17
1.1 MCU	17
1.2 ARM	17
1.3 RISC	17
1.4 ISA	17
1.5 SVC	17
1.6 FIQ	17
1.7 DMA	17
1.8 SPSR	17
1.9 CPSR	17
1.10 SP	17
1.11 LR	17
1.12 MMU	17
1.13 MPU	17
1.14 AMBA	17
1.15 SoC	17
1.16 APB	17
1.17 AHB	17
1.18 JTAG	18
1.19 RAM	18
1.20 CPU	18
1.21 FET	18
1.22 SRAM	18
1.23 DRAM	18
1.24 SLC	18
1.25 ALU	18
1.26 APSR	18
1.27 EPSR	18
1.28 IPSR	18
1.29 I/O	18
1.30 NVIC	18
1.31 SysTick	18
1.32 Opcode	18
1.33 MSB	18
1.34 LSB	18
1.35 VFP	19
1.36 FPU	19
1.37 MPU	19

1.38	GiB	19
1.39	IDE	19
1.40	I/O or IO	19
1.41	GPIO	19
1.42	ADC	19
1.43	PWM	19
1.44	UART	19
1.45	DTE	19
1.46	DCE	19
1.47	TX or Tx	19
1.48	RX or Rx	19
1.49	GND	19
1.50	BRD	19
1.51	TTL	19
1.52	RTOS	20
1.53	CCP	20
1.54	IRQ	20
1.55	ISR	20
2	Definitions	21
2.1	Microprocessor system	21
2.2	Word	21
2.3	Using I/O modules	21
2.4	Bits and bytes	21
2.5	Fast interrupt request (FIQ)	21
2.6	Direct memory access (DMA)	21
2.7	Truth table	21
2.8	Rising edge	22
2.9	Falling edge	22
2.10	Bus	23
2.10.1	Address bus	23
2.10.2	Data bus	23
2.11	Endianness of memory	23
2.12	Base address	23
2.13	Address offset	23
2.14	Address indexing	23
2.15	Configuration register	23
2.16	Maskable data register	24
2.16.1	Example	24
2.16.2	Why have maskable data registers?	24
3	Basics of brains	25
3.1	Characteristics of the human brain	25
3.2	Blueprint of a machine brain	25
3.3	Characteristics of a machine brain	25
3.4	Van Neumann architecture	26
3.5	Harvard architecture	26
4	ARM: Advanced RISC Machines	27
4.1	ARM Family	27
4.2	ARM Cortex uses the RISC architecture	27
4.2.1	Architecture of the ARM7TDMI Processor	28

4.3	Modes in ARM Cortex	28
4.4	Registers in ARM Cortex	29
4.4.1	Register table	30
4.4.2	Current program status register (CPSR, R16)	31
4.4.3	Cortex M4's status register	32
4.5	ARM Cortex as a microcontroller	32
4.6	ARM features	32
4.7	Memory inside TM4C123G	33
4.8	ARM Cortex M's memory space	35
4.8.1	On-chip memory subspace	36
4.8.2	Off-chip memory subspace	36
4.8.3	Private memory subspace	37
4.9	Cortex M4 pin diagram	37
4.10	Cortex M4 table of pin usage	38
4.11	Cortex M4's floating point unit's registers	39
5	Typical types of application software	40
6	Binary logic devices	40
6.1	Binary number systems	40
6.2	Binary logic operations	41
6.3	NOT logic gate	41
6.3.1	Truth table	41
6.3.2	Circuit diagrams	42
6.4	AND logic gate	43
6.4.1	Truth table	43
6.4.2	Circuit diagrams	44
6.5	OR logic gate	45
6.5.1	Truth table	45
6.5.2	Circuit diagrams	46
6.6	XOR logic gate	47
6.6.1	Truth table	47
6.6.2	Circuit diagrams	47
6.7	XNOR logic gate	48
6.7.1	Truth table	48
6.7.2	Circuit diagrams	48
6.8	D flip-flop	49
6.8.1	Truth table	49
6.9	8-bit shift register	50
6.10	Single bit Static RAM (SRAM)	50
6.11	Single bit Dynamic RAM (DRAM)	51
6.12	1-bit flash cell	51
6.12.1	Line of flash cells	52
6.12.2	Writing	52
6.12.3	NOR erasing	52
6.12.4	NOR writing	53
6.12.5	Reading	53
6.12.6	NOR reading	54
7	Number representation	55
7.1	History of number systems	55
7.2	Binary number system	55

7.2.1	Format of binary numbers	55
7.2.2	Integer example	55
7.2.3	Fractional example	55
7.2.4	Binary addition	56
7.2.5	Binary subtraction	57
7.3	Decimal number system	58
7.3.1	Integer example	58
7.3.2	Fractional example	58
7.3.3	Decimal addition	59
7.3.4	Decimal subtraction	60
7.3.5	Hexadecimal number system	61
7.3.6	Integer example	61
7.3.7	Fractional example	61
7.3.8	Hexadecimal addition	62
7.3.9	Hexadecimal subtraction	63
7.3.10	Hexadecimal ASCII table	64
7.4	Conversion look-up table	64
7.4.1	Conversion from binary to hexadecimal	65
7.4.2	Conversion from hexadecimal to binary	65
7.4.3	Conversion from decimal to binary, integer part	66
7.4.4	Conversion from decimal to binary, fractional part	66
7.4.5	Example of converting a decimal number with decimals to binary	67
7.4.6	Conversion from decimal to hexadecimal, integer part	67
7.4.7	Conversion from decimal to hexadecimal, fractional part	68
7.5	Representation of integers	68
7.5.1	Binary-coded decimal of integers	68
7.5.2	Signed magnitude format of integers	68
7.5.3	Two's complement format of integers	69
7.5.4	Reasons to use two's complement format	69
7.6	Representation of floating point numbers	70
7.6.1	Binary-coded decimal of floating-point numbers	70
7.6.2	Exponent-mantissa format of floating point numbers	70
7.6.3	Conversion of decimals to exponent-mantissa format	71
7.6.4	Zero and infinity in exponent-mantissa format	72
8	Nature of programming	72
8.1	Programming versus writing	72
8.2	How to organise your solution?	73
8.2.1	Example of ($x = (a + b) - c$)	73
8.3	Procedure of planning computations	74
8.3.1	Example of transforming a solution into an algorithm	74
8.3.2	Example of flowchart corresponding to algorithm	74
9	Allocation of memory	75
9.1	Why allocate memory?	75
9.2	Memory allocation scenarios	75
9.3	Memory allocation example	76
10	ARM programming tools	76
10.1	Overview	76
10.2	Assembler directives for memory destination (GCC and ArmClang)	77
10.2.1	Example 1	78

10.2.2	Example 2	78
10.2.3	Example 3	79
10.2.4	Example 4	79
10.3	Assembler directives for controlling conditional computations (GCC and ArmClang)	79
10.3.1	Example	80
10.4	Directive for creating reusable block of codes (GCC and ArmClang)	81
10.4.1	Example	81
11	Memory-mapped I/O devices	82
11.1	Data flow	82
11.2	Basic unit of input/output data	82
11.3	Memory allocation	82
11.4	Example of internal data input from port F	83
11.5	TM4C123G GPIO module	85
11.6	TM4C123G ADC module	86
11.7	TM4C123G PWM module	86
11.8	Data organisation nomenclature	87
11.9	Address indexing methods	88
11.9.1	Pre-indexing without write-back	89
11.9.2	Pre-indexing with write-back	89
11.9.3	Post-indexing	89
11.9.4	Example 1: Pre-indexed address without write-back	89
11.9.5	Example 2: Pre-indexed address with write-back	89
11.10	ARM Cortex M4 GPIO ports	90
11.10.1	Read and write example	90
11.10.2	Example with port F	91
11.11	Encoding of instructions in ARM	92
11.11.1	LDR and STR instructions for data I/O	92
11.11.1.1	For sizes other than a word	93
12	Digital input and output (GPIO)	94
12.1	Digital signals	94
12.1.1	Logic levels	94
12.1.2	Transistor-transistor logic (TTL) levels	94
12.1.3	Acceptable input digital signals (TTL levels)	94
12.1.4	Representation and events of digital signals	95
12.1.5	Signal lines of digital signals	95
12.1.6	Tri-state logic	95
12.1.7	Schmitt filters or triggers	95
12.1.8	Principle of Schmitt filters	95
12.1.9	Communication of digital signals	96
12.2	Synchronous digital input and output	96
12.2.1	Working principle of synchronous serial input and output	96
12.2.2	SPI and I2C example	97
12.2.3	Example of an I2C bus	97
12.2.4	Working principle of synchronous parallel input and output	98
12.2.5	Receiver initiated parallel input	99
12.2.6	Transmitter initiated parallel input	100
12.3	Asynchronous digital input and output	101
12.3.1	Working principle of asynchronous digital input and output (data line)	101
12.3.2	Working principle of asynchronous digital input and output (data line)	102

12.3.2.1	Example: Keypad	103
12.3.2.2	Example: LED dot matrix	103
13	Analogue input and outputs	104
13.1	Types of analogue inputs	104
13.2	Types of analogue outputs	104
13.3	Working principle of a digital to analogue converter (DAC)	104
13.3.1	Exercise	105
13.4	Specification of DACs	105
13.4.1	Exercise	105
13.5	Working principle of an analogue to digital converter (ADC)	106
13.5.1	Example: Speech signals	107
13.6	Specification of ADCs	108
13.6.1	Exercise	108
14	Universal asynchronous receiver/transmitter (UART)	109
14.1	Basic elements of a parallel/serial interface	109
14.2	UART connection to external device	110
14.3	Data communication (serial)	110
14.4	Asynchronous serial format (data framing)	110
14.4.1	Standard format	111
14.5	ASCII format	112
14.6	Baud rate (data transmission speed)	113
14.6.1	Example	113
14.7	UART data transmission	114
14.8	Syncing receiver section with serial data	115
14.8.1	Syncing procedure	116
15	Synchronous serial interface (SSI)	117
15.1	UART vs SSI	117
16	Stepper motors	118
16.1	Working principle	118
16.2	Advantages	119
16.3	Disadvantages	119
16.4	Applications	119
16.5	Output control	119
16.5.1	Example	120
17	Brushed DC motors	121
17.1	Working principle	121
17.2	L289N H-bridge motor driver	121
17.2.1	Output control	122
17.2.1.1	PWM logic control	122
17.2.1.2	PWM power control	122
18	Brushless DC motors	123
18.1	Hall effect sensor	123
18.2	Working principle	124
18.3	Sequence	124
18.4	Advantages	126
18.5	Disadvantages	126
19	TM4C123GH6PM General Purpose Input/Output (GPIO)	127
19.1	Key features	127
19.2	Process of operating GPIO	127

19.3	Pin diagram	128
19.4	Analogue and digital I/O pads	129
19.5	GPIO pins and alternate functions	130
19.6	GPIO register map	132
19.7	Registers	133
19.7.1	General-Purpose Input/Output Run Mode Clock Gating Control (RCGCGPIO) ..	133
19.7.2	GPIO Digital Enable (GPIODEN)	135
19.7.3	GPIO Direction (GPIODIR)	136
19.7.4	GPIO Interrupt Sense (GPIOIS)	137
19.7.5	GPIO Interrupt Event (GPIOEV)	138
19.7.6	GPIO Interrupt Both Edges (GPIOIBE)	139
19.7.7	GPIO Interrupt Mask (GPIOIM)	140
19.7.8	GPIO 2-mA Drive Select (GPIODR2R)	141
19.7.9	GPIO 4-mA Drive Select (GPIODR4R)	142
19.7.10	GPIO 8-mA Drive Select (GPIODR8R)	143
19.7.11	GPIO Open Drain Select (GPIOODR)	144
19.7.12	GPIO Pull-Up Select (GPIOPUR)	145
19.7.13	GPIO Pull-Down Select (GPIOPDR)	146
19.7.14	GPIO Pins with special considerations	147
19.7.15	GPIO Alternate Function Select (GPIOAFSEL)	148
19.7.16	GPIO Lock (GPIOLOCK)	149
19.7.17	GPIO Commit (GPIOCR)	150
19.7.18	GPIO Analog Mode Select (GPIOAMSEL)	152
19.7.19	GPIO Data (GPIODATA)	153
19.7.19.1	GPIO write example	154
19.7.19.2	GPIO read example	154
19.7.20	GPIO Raw Interrupt Status (GPIOISR)	155
19.7.21	GPIO Masked Interrupt Status (GPIOMIS)	156
19.7.22	GPIO Interrupt Clear (GPIOICR)	157
19.7.23	GPIO Port Control (GPIOPCTL)	158
19.7.24	GPIO ADC Control (GPIOADCCTL)	159
19.8	Initialisation	160
19.8.1	Example	160
19.9	Quick reference	161
20	TM4C123GH6PM Cortex M4 analogue to digital converter (ADC)	162
20.1	Functions	162
20.2	Main features	162
20.3	Implementation of 2 ADC blocks	162
20.4	ADC module block diagram	163
20.5	ADC signal pin names	164
20.6	ADC register map	165
20.7	GPIOCTL for ADC	166
20.8	Functional description	167
20.9	Registers	168
20.9.1	ADC Peripheral Properties (ADCPP)	168
20.9.2	ADC Peripheral Configuration (ADCPC)	170
20.9.3	Analogue-to-Digital Converter Run Mode Clock Gating Control (RCGCADC) ..	171
20.9.4	ADC Sample Sequencer Priority (ADCSSPRI)	172
20.9.5	ADC Control (ADCCTL)	174

20.9.6	ADC Sample Sequence Input Multiplexer Select 0 (ADCSSMUX0)	175
20.9.7	ADC Sample Sequence Input Multiplexer Select n (ADCSSMUXn)	177
20.9.8	ADC Event Multiplexer Select (ADCEMUX)	178
20.9.8.1	Valid configurations for the trigger select fields	179
20.9.9	ADC Trigger Source Select (ADCTSSEL)	180
20.9.10	ADC Clock Configuration (ADCCC)	182
20.9.11	ADC Sample Sequence Control 0 (ADCSSCTL0)	183
20.9.12	ADC Sample Sequence Control n (ADCSSCTLn)	192
20.9.13	ADC Sample Sequence Control 3 (ADCSSCTL3)	198
20.9.14	ADC Active Sample Sequencer (ADCACTSS)	200
20.9.15	ADC Sample Phase Control (ADCSPC)	202
20.9.16	ADC Digital Comparator Range n (ADCDCCMPn)	203
20.9.17	ADC Digital Comparator Reset Initial Conditions (ADCDCRIC)	204
20.9.18	ADC Sample Sequence 0 Digital Comparator Select (ADCSSDC0)	210
20.9.19	ADC Sample Sequence 0 Operation (ADCSSOP0)	212
20.9.20	ADC Processor Sample Sequence Initiate (ADCPSSI)	214
20.9.21	ADC Sample Averaging Control (ADCSAC)	216
20.9.22	ADC Interrupt Mask (ADCIM)	217
20.9.23	ADC Sample Sequence Result FIFO n (ADCSSFIFOn)	219
20.9.24	ADC Sample Sequence FIFO n Status (ADCSSFSTATn)	220
20.9.24.1	Illustration of the head and tail address	221
20.9.25	ADC Raw Interrupt Status (ADCRIS)	222
20.9.26	ADC Digital Comparator Interrupt Status and Clear (ADCDCISC)	224
20.9.27	ADC Interrupt Status and Clear (ADCISC)	226
20.9.28	ADC Overflow Status (ADCOSTAT)	229
20.10	Reference voltages of Cortex M4 ADC	231
20.11	Output from Cortex M4 ADC	231
20.11.1	Exercise	232
20.12	Initialisation	232
20.12.1	Example	233
20.13	Quick reference	235
20.14	Differential sampling	236
20.14.1	Differential sampling pairs	236
20.15	Output of interrupts	236
20.15.1	Low band operation	236
20.15.2	Mid band operation	237
20.15.3	High band operation	237
20.15.4	Hardware output averaging	238
21	TM4C123GH6PM UART	239
21.1	UARTs in Cortex M4	240
21.2	UART module block diagram	241
21.3	J connectors	242
21.3.1	J1 connector	242
21.3.2	J2 connector	243
21.3.3	J3 connector	243
21.3.4	J4 connector	244
21.4	UART description and configuration	245
21.5	UART register map	245
21.6	Essential registers	246

21.7	Registers	247
21.7.1	Universal Asynchronous Receiver/Transmitter Run Mode Clock Gating Control (RCGUART)	247
21.7.2	UART Integer Baud-Rate Divisor (UARTIBRD)	249
21.7.3	UART Fractional Baud-Rate Divisor (UARTFBRD)	250
21.7.4	UART Line Control (UARTLCRH)	251
21.7.5	UART Clock Configuration (UARTCC)	253
21.7.6	UART DMA Control (UARTDMACTL)	254
21.7.7	UART Control (UARTCTL)	255
21.7.8	UART Data (UARTDR)	260
21.7.9	UART Flag (UARTFR)	262
21.7.10	UART Raw Interrupt Status (UARTRIS)	264
21.7.11	UART Interrupt Mask (UARTIM)	268
21.7.12	UART Masked Interrupt Status (UARTMIS)	271
21.7.13	UART Interrupt Clear (UARTICR)	274
21.7.14	UART Interrupt FIFO Level Select (UARTIFLS)	276
21.8	Initialisation and configuration	278
21.9	UART baud rate settings	278
21.10	Baud rate setting	278
21.10.1	Example	278
21.10.1.1	How to convert fractions in decimal to binary?	279
21.10.1.2	Continuing the example	279
21.10.2	Procedure	279
21.11	UART transmission procedure	280
21.12	UART transmission sequence	280
21.13	UART receiving sequence	281
22	TM4C123GH6PM SSI	281
22.1	SSI pin names	282
22.2	SSI module	283
22.3	J connectors	284
22.3.1	J1 connector	284
22.3.2	J2 connector	285
22.3.3	J3 connector	285
22.3.4	J4 connector	286
22.4	SSI register map	286
22.5	SSI register names	287
22.6	Registers	288
22.6.1	SSI Control 0 (SSICR0)	288
22.6.2	SSI Control 1 (SSICR1)	290
22.6.3	SSI Clock Configuration (SSICC)	292
22.6.4	SSI Clock Prescale (SSICPSR)	293
22.6.5	SSI Status (SSISR)	294
22.6.6	Synchronous Serial Interface Run Mode Clock Gating Control (RCGCSSI)	296
22.6.7	SSI Data (SSIDR)	297
22.7	Connection to I/O device	298
22.8	SSI clock frequency configuration	300
22.9	SSI mode control bits	301
22.10	Initialisation	301
22.11	Configuration	302

22.11.1	Example	302
23	TM4C123GH6PM PWM	303
23.1	Block diagrams	303
23.1.1	PWM module	303
23.1.2	PWM generator block	304
23.2	PWM pins	305
23.3	PWM register map	307
23.4	Registers	310
23.4.1	Run-Mode Clock Configuration (RCC)	310
23.4.1.1	Possible system clock frequencies using SYSDIV field	316
23.4.2	Run Mode Clock Gating Control Register 0 (RCGC0)	317
23.4.3	Run Mode Clock Gating Control Register 2 (RCGC2)	320
23.4.4	PWMn Control (PWMMnCTL)	322
23.4.5	PWMn Generator A Control (PWMMnGENA)	328
23.4.6	PWMn Generator B Control (PWMMnGENB)	330
23.4.7	PWMn Load (PWMMnLOAD)	332
23.4.8	PWMn Compare A (PWMMnCMPA)	333
23.4.9	PWMn Compare B (PWMMnCMPB)	334
23.4.10	PWM Output Enable (PWMMENABLE)	335
23.4.11	PWMn Dead-Band Rising-Edge Delay (PWMMnDBRISE)	337
23.4.12	PWMn Dead-Band Falling-Edge-Delay (PWMMnDBFALL)	338
23.4.13	PWMn Fault Source 0 (PWMMnFLTSRC0)	339
23.4.14	PWMn Fault Source 1 (PWMMnFLTSRC1)	341
23.4.15	PWMn Dead-Band Control (PWMMnDBCTL)	345
23.4.16	PWM Time Base Sync (PWMSYNC)	346
23.4.17	PWM Master Control (PWMCTL)	347
23.4.18	PWMn Minimum Fault Period (PWMMnMINFLTPER)	349
23.4.19	PWMn Fault Pin Logic Sense (PWMMnFLTSEN)	350
23.4.20	PWM Enable Update (PWMMENUPD)	351
23.4.21	PWM Output Inversion (PWMMINVERT)	356
23.4.22	PWM Output Fault (PWMMFAULT)	358
23.4.23	PWM Fault Condition Value (PWMMFAULTVAL)	360
23.4.24	PWM Interrupt Enable (PWMMINTEN)	363
23.4.25	PWMn Interrupt and Trigger Enable (PWMMnINTEN)	365
23.4.26	PWM Peripheral Properties (PWMMPP)	369
23.4.27	PWMn Fault Status 0 (PWMMnFLTSTAT0)	371
23.4.28	PWMn Fault Status 1 (PWMMnFLTSTAT1)	373
23.4.29	PWMn Counter (PWMMnCOUNT)	376
23.4.30	PWM Raw Interrupt Status (PWMMRIS)	377
23.4.31	PWM Interrupt Status and Clear (PWMMISC)	380
23.4.32	PWMn Raw Interrupt Status (PWMMnRIS)	383
23.4.33	PWMn Interrupt Status and Clear (PWMMnISC)	385
23.4.34	PWM Status (PWMMSTATUS)	388
23.5	PWM module clock source	389
23.6	PWM generator timer	389
23.7	PWM generator comparators	390
23.8	PWM signal generator	390
23.9	PWM dead-band generator	391
23.10	Initialisation and configuration	392

24	TM4C123GH6PM timers and counters	393
24.1	System timer (SysTick)	393
24.1.1	Register map	394
24.1.2	Registers	394
24.1.2.1	SysTick Control and Status Register (STCTRL)	395
24.1.2.2	SysTick Reload Value Register (STRELOAD)	397
24.1.2.3	SysTick Current Value Register (STCURRENT)	398
24.1.3	Initialisation sequence	399
24.1.3.1	Example code	399
24.2	General purpose timers (GPTM)	400
24.2.1	Block diagram	401
24.2.2	Input capture pins	402
24.2.3	Functional description	403
24.2.4	Timer modes	403
24.2.5	Register map	404
24.2.6	Timer 0 registers	405
24.2.7	Essential function in registers	405
24.2.8	Overview of GPTM use	405
24.2.9	Prescaler configurations	406
24.2.10	Registers	407
24.2.10.1	GPTM Configuration (GPTMCFG)	407
24.2.10.2	GPTM Control (GPTMCTL)	409
24.2.10.3	GPTM Timer A Mode (GPTMTAMR)	414
24.2.10.4	GPTM Timer A Value (GPTMTAV)	419
24.2.10.5	GPTM Timer A Prescale (GPTMTAPR)	420
24.2.10.6	GPTM Timer A Match (GPTMTAMATCHR)	421
24.2.10.7	GPTM Timer A Prescale Match (GPTMTAPMR)	422
24.2.10.8	GPTM Timer A Interval Load (GPTMTAILR)	423
24.2.10.9	GPTM Timer A (GPTMTAR)	424
24.2.10.10	GPTM Timer A Prescale Snapshot (GPTMTAPS)	425
24.2.10.11	GPTM Timer A Prescale Value (GPTMTAPV)	426
24.2.10.12	GPTM Timer B Mode (GPTMTBMR)	427
24.2.10.13	GPTM Timer B Value (GPTMTBV)	432
24.2.10.14	GPTM Timer B Prescale (GPTMTBPR)	433
24.2.10.15	GPTM Timer B Match (GPTMTBMATCHR)	434
24.2.10.16	GPTM Timer B Prescale Match (GPTMTBPMR)	435
24.2.10.17	GPTM Timer B Interval Load (GPTMTBILR)	436
24.2.10.18	GPTM Timer B (GPTMTBR)	437
24.2.10.19	GPTM Timer B Prescale Snapshot (GPTMTBPS)	438
24.2.10.20	GPTM Timer B Prescale Value (GPTMTBPV)	439
24.2.10.21	16/32-Bit General-Purpose Timer Run Mode Clock Gating Control (RCGCTIMER)	440
24.2.10.22	32/64-Bit Wide General-Purpose Timer Run Mode Clock Gating Control (RCGCWTIMER)	442
24.2.10.23	GPTM Interrupt Clear (GPTMICR)	444
24.2.10.24	GPTM Interrupt Mask (GPTMIMR)	446
24.2.10.25	GPTM Raw Interrupt Status (GPTMRIS)	449
24.2.10.26	GPTM Masked Interrupt Status (GPTMMIS)	453
24.2.11	Initialisation and configuration	457

24.2.11.1	One-shot or periodic timer configuration	457
24.2.11.2	Real time clock mode	457
24.2.11.3	Input Edge Count mode	458
24.2.11.4	Input Edge Time mode	459
24.2.11.5	PWM mode	459
25	TM4C123GH6PM interrupts	460
25.1	Interrupts	461
25.2	Registers	464
25.2.1	Interrupt 0-31 Set Enable (EN0)	464
25.2.2	Interrupt 128-138 Set Enable (EN4)	465
25.2.3	Interrupt 0-31 Clear Enable (DIS0)	466
25.2.4	Interrupt 128-138 Clear Enable (DIS4)	467
25.2.5	Interrupt 0-3 Priority (PRI0)	468
25.2.6	System Handler Priority 3 (SYSPRI3)	471
25.3	Working with interrupts	472
25.4	Interrupt priority	473
25.5	Initialising edge triggered interrupt on a pin	473
25.6	SysTick periodic interrupt	474
25.6.1	How to determine the frequency of the periodic interrupt	474
25.6.2	SysTick periodic interrupt initialisation	474
25.6.2.1	Example	474
25.7	Timer period interrupt	475
25.7.1	Initialisation	475
26	ARM reference	476
26.1	ARM's instruction format	476
26.2	ARM's data formats	476
26.3	Syntax	477
26.3.1	Semicolon (;)	477
26.3.2	Hash (#)	477
26.3.2.1	Example	477
26.3.3	Square brackets ([])	477
26.3.3.1	Example 1	477
26.3.3.2	Example 2	477
26.3.3.3	Example 3	477
26.3.3.4	Example 4	478
26.3.3.5	Example 5	478
26.3.3.6	Example 6	478
26.4	Addressing modes	478
26.4.1	Examples with LDR	479
26.4.2	Syntax examples	479
26.5	Condition code suffixes	480
26.6	ARM CPU instructions	480
26.7	MOV (Move)	480
26.7.1	Example	480
26.8	MVN (Move Not)	481
26.8.1	Example	481
26.9	MOVT (Move Top)	481
26.9.1	Example	481
26.10	MOVW (Move Wide)	481

26.10.1 Example	481
26.11 LDR (Load)	482
26.11.1 Example	482
26.11.2 Pseudo-instruction	482
26.12 LDRD (Load Double-Word)	482
26.12.1 Example	482
26.13 LDRB (Load Byte)	482
26.13.1 Example	482
26.14 LDRSB (Load Signed Byte)	483
26.14.1 Example	483
26.15 LDRH (Load Half-Word)	483
26.15.1 Example	483
26.16 LDRSH (Load Signed Half-Word)	483
26.16.1 Example	483
26.17 STR (Store)	484
26.17.1 Example 1	484
26.18 STRD (Store Double-Word)	484
26.18.1 Example	484
26.19 STRB (Store Byte)	484
26.19.1 Example	484
26.20 STRSB (Store Signed Byte)	485
26.20.1 Example	485
26.21 STRH (Store Half-Word)	485
26.21.1 Example	485
26.22 STRSH (Store Signed Half-Word)	485
26.22.1 Example	485
26.23 ADD (Add)	486
26.23.1 Example 1	486
26.23.2 Example 2	486
26.23.3 Example 3	486
26.23.4 Example 4	486
26.24 ADC (Add with Carry)	487
26.24.1 Example	487
26.25 SADD16 and SADD8 (Signed Add)	488
26.25.1 Example	488
26.26 SUB (Subtract)	489
26.26.1 Example	489
26.27 SBC (Subtract with Carry)	490
26.27.1 Example	490
26.28 RSB (Reverse Subtract)	490
26.28.1 Example	490
26.29 RSC (Reverse Subtract with Carry)	491
26.29.1 Example	491
26.30 SSUB16 and SSUB8 (Signed Subtract)	491
26.30.1 Example	491
26.31 MUL (Multiply)	492
26.31.1 Example	492
26.32 MLA (Multiply Accumulate)	492
26.32.1 Example	492

26.33	MLS (Multiply Subtract)	493
26.33.1	Example	493
26.34	UMULL (Unsigned Multiply Long)	493
26.34.1	Example	493
26.35	UMLAL (Unsigned Multiply Long with Accumulate)	494
26.35.1	Example	494
26.36	SMULL (Signed Multiply Long)	494
26.36.1	Example	494
26.37	SMLAL (Signed Multiply Long with Accumulate)	494
26.37.1	Example	494
26.38	UDIV (Unsigned Divide)	495
26.38.1	Example	495
26.39	AND (Bitwise AND)	495
26.39.1	Example	495
26.39.2	Example of forcing bits to 0	495
26.40	ORR (Bitwise OR)	496
26.40.1	Example	496
26.40.2	Example of forcing bits to 1	496
26.41	EOR (Exclusive OR, XOR)	497
26.41.1	Example	497
26.41.2	Example of flipping bits	497
26.42	BIC (Bit Clear)	498
26.42.1	Example	498
26.43	Shift operators	499
26.44	LSL (Logical Shift Left)	500
26.44.1	Example 1	500
26.44.2	Example 2	500
26.44.3	Example 3	500
26.45	LSR (Logical Shift Right)	501
26.45.1	Example	501
26.46	ASR (Arithmetic Shift Right)	502
26.46.1	Example	502
26.47	ROR (Rotate Right)	503
26.47.1	Example	503
26.48	RRX (Rotate Right Extended)	504
26.48.1	Example	504
26.49	B (Branch)	505
26.49.1	Example 1	505
26.49.2	Example 2	505
26.49.3	Example 3	505
26.50	BL (Branch with Link)	505
26.50.1	Example	505
26.51	BX and BXNS (Branch and Exchange, and Branch and Exchange Non-Secure)	506
26.51.1	Example 1	506
26.51.2	Example 2	506
26.52	ADR	506
26.52.1	Example	506
26.53	CMP (Compare)	507
26.53.1	Example	507

26.54 CMM (Compare Negative)	507
26.54.1 Example	507
26.55 TEQ (Test Equivalence)	508
26.55.1 Example	508
26.56 TST (Test bits)	508
26.56.1 Example	508
26.57 Vector floating point (VFP) operations	509
26.58 VLDR (Vector Load)	509
26.58.1 Example	509
26.59 VSTR (Vector Load)	510
26.59.1 Example	510
26.60 VMOV (Vector Move)	510
26.60.1 Example	510
26.61 VADD (Vector Add)	510
26.61.1 Example	510
26.62 Directives	511
26.63 Allocating memory to house instructions or data	511
26.63.1 Example	511
26.64 SETS	511
26.64.1 Example	511
26.65 EQU	511
26.65.1 Example	511
26.66 RN	512
26.66.1 Example	512
26.67 MACRO and MEND	512
26.67.1 Example 1	512
26.67.2 Example 2	513
26.68 AREA	513
26.68.1 Valid section attributes	513
26.68.2 Example	513
26.69 ENTRY and END	514
26.69.1 Example	514
26.70 DCB (Declare 8-bit bytes)	514
26.70.1 Example 1	514
26.70.2 Example 2	514
26.70.3 Example 3	514
26.71 DCW (Declare 16-bit half-words)	515
26.71.1 Example 1	515
26.71.2 Example 2	515
26.71.3 Example 3	515
26.72 DCD (Declare 32-bit words)	516
26.72.1 Example 1	516
26.72.2 Example 2	516
26.72.3 Example 3	516
26.73 ALIGN	517
26.73.1 Example 1	517
26.73.2 Example 2	517
26.74 SPACE	518
26.74.1 Example	518

26.75 FILL	518
26.75.1 Example	518
26.76 LTORG (Lookup table organised)	519
26.76.1 Example	519
26.77 EXPORT or GLOBAL	520
26.77.1 Example	520
26.78 IMPORT or EXTERN	520
26.78.1 Example	520
26.79 GET or INCLUDE	521
26.79.1 Example	521
26.80 Other directives offered by ARM Keil	521

1 Abbreviations

1.1 MCU

MCU stands for microcontroller.

1.2 ARM

ARM stands for Advanced RISC Machines.

1.3 RISC

RISC stands for Reduced Instruction Set Computer.

1.4 ISA

ISA stands for Instruction Set Architecture which is innate to all processors.

1.5 SVC

SVC stands for supervisor call instruction.

1.6 FIQ

FIQ stands for fast interrupt request.

1.7 DMA

DMA stands for direct memory access.

1.8 SPSR

SPSR stands for the saved program status register.

1.9 CPSR

CPSR stands for the current program status register.

1.10 SP

SP stands for the stack pointer.

1.11 LR

LR stands for the link register.

1.12 MMU

MMU stands for memory management unit.

1.13 MPU

MPU stands for memory protection unit.

1.14 AMBA

AMBA stands for advanced microcontroller bus architecture.

1.15 SoC

SoC stands for system-on-a-chip.

1.16 APB

APB stands for advanced peripheral bus.

1.17 AHB

AHB stands for advanced high-performance bus.

1.18 JTAG

JTAG stands for Joint Test Action Group.

1.19 RAM

RAM stands for random access memory.

1.20 CPU

CPU stands for central processing unit.

1.21 FET

FET stands for field effect transistor.

1.22 SRAM

SRAM stands for static RAM, or Static Random Access Memory.

1.23 DRAM

DRAM stands for dynamic RAM, or Dynamic Random Access Memory.

1.24 SLC

SLC stands for single level cell.

1.25 ALU

ALU stands for arithmetic logic unit.

1.26 APSR

APSR stands for application program status register.

1.27 EPSR

EPSR stands for execution program status register.

1.28 IPSR

IPSR stands for interrupt program status register.

1.29 I/O

I/O stands for input and output.

1.30 NVIC

NVIC stands for nested vectored interrupt controller.

1.31 SysTick

SysTick stands for system tick timer.

1.32 Opcode

Opcode stands for operation code.

1.33 MSB

MSB stands for most significant bit.

1.34 LSB

LSB stands for least significant bit.

1.35 VFP

VFP stands for vector floating point.

1.36 FPU

FPU stands for floating point unit.

1.37 MPU

MPU stands for multicore processing unit.

1.38 GiB

GiB stands for gibibyte.

1.39 IDE

IDE stands for integrated development environment.

1.40 I/O or IO

I/O or IO stands for input/output.

1.41 GPIO

GPIO stands for general purpose input/output.

1.42 ADC

ADC stands for analogue to digital converter.

1.43 PWM

PWM stands for pulse width modulation.

1.44 UART

UART stands for universal asynchronous receiver/transmitter.

1.45 DTE

DTE stands for data terminal equipment.

1.46 DCE

DCE stands for data communication equipment.

1.47 TX or Tx

TX or Tx stands for transmitting.

1.48 RX or Rx

RX or Rx stands for receiving.

1.49 GND

GND stands for ground.

1.50 BRD

BRD stands for baud-rate divider.

1.51 TTL

TTL stands for transistor-transistor logic.

1.52 RTOS

RTOS stands for real time operating system.

1.53 CCP

CCP stands for capture, compare and PWM.

1.54 IRQ

IRQ stands for interrupt request.

1.55 ISR

ISR stands for interrupt service routine.

2 Definitions

2.1 Microprocessor system

A microprocessor system is a system which is built on top of logic devices that can:

- Undertake arithmetic operations
- Undertake logic operations
- Interface with external devices

2.2 Word

A word is a 2 byte (16-bit) value.

2.3 Using I/O modules

- Configure control registers
- Clear or monitor status registers
- Read or write data registers
- Instructions:
 - ▶ `MOV <address of destination>, <source of value>`
Move instruction, which copies the value from the source to the destination.
 - ▶ `LDR <address of destination>, <source of value>`
Load register instruction, which loads a value from a memory address into a register.
 - ▶ `STR <address of destination>, <source of value>`
Store instruction, which stores a value from a register to a memory address.

2.4 Bits and bytes

- 8 bits are called a byte.
- 16 bits of 2 bytes are called a half-word.
- 32 bits or 4 bytes are called a word.
- 64 bits or 8 bytes are called a double-word.

2.5 Fast interrupt request (FIQ)

An fast interrupt request (FIQ) is a higher priority interrupt request, that is prioritised by disabling interrupt request (IRQ) and other fast interrupt request handlers during request servicing. Therefore, no other interrupts can occur during the processing of the active FIQ interrupt.

2.6 Direct memory access (DMA)

Direct memory access (DMA) is a feature of microprocessor systems that allows certain hardware subsystems to access main system memory such as RAM (Random Access Memory) independently of the central processing unit (CPU).

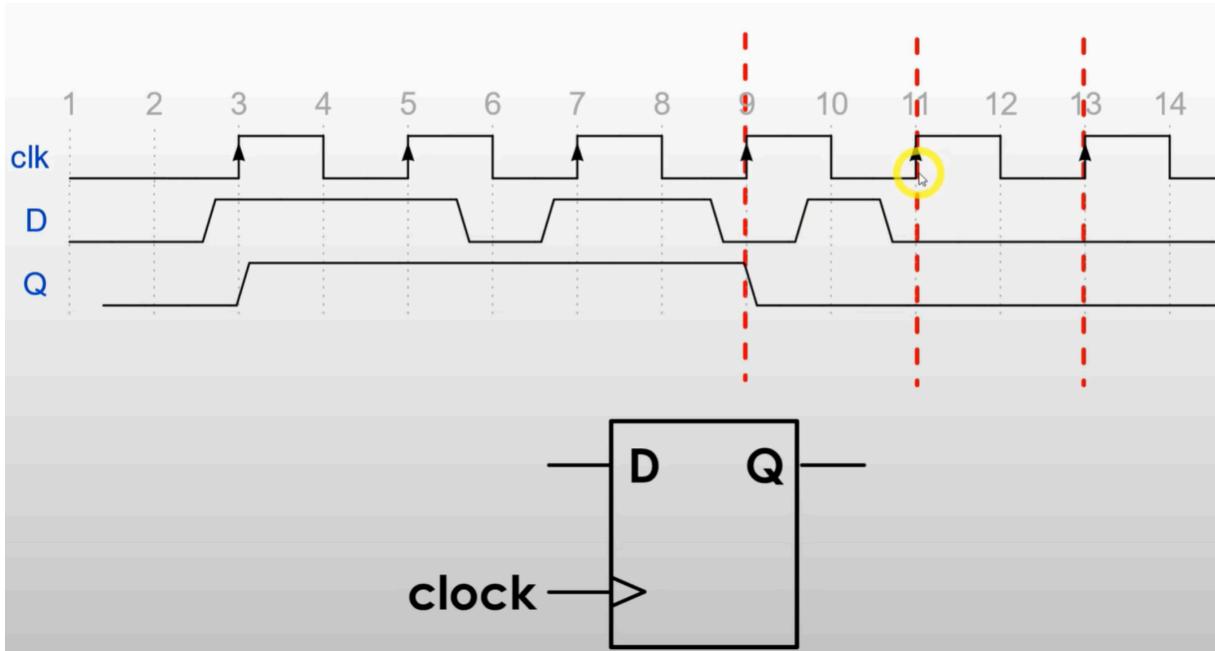
2.7 Truth table

A truth table maps the all the possible inputs to the outputs. Below is truth table of an inverter.

Input	Output
1	0
0	1

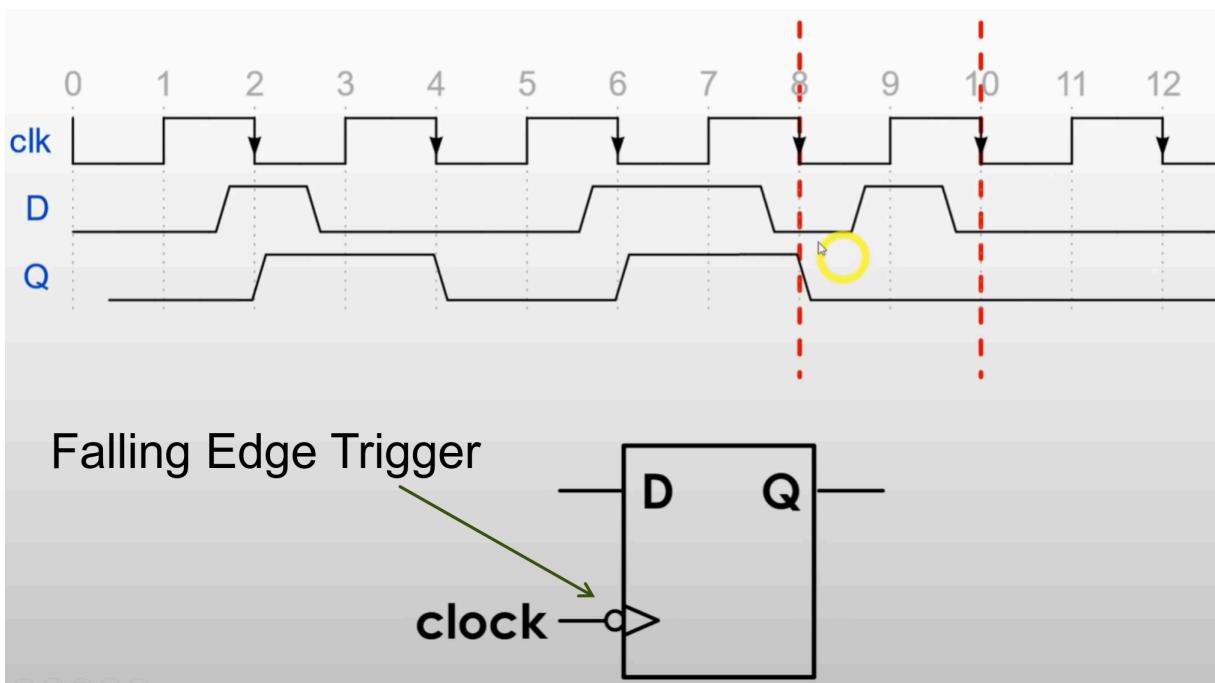
2.8 Rising edge

- A rising edge is when the input transitions from 0 to 1.
- A transistor, if not notated, is usually sensitive to the rising edge as shown below.



2.9 Falling edge

- A falling edge is when the input transitions from 1 to 0.
- A transistor sensitive to the falling edge is notated using a small circle at the input of the transistor, as shown below.



2.10 Bus

- A bus is essentially a **data highway** that transfers data between components inside a computer or between computers.
- It encompasses both hardware, like wires, optical fibre, and software, including communication protocols.

2.10.1 Address bus

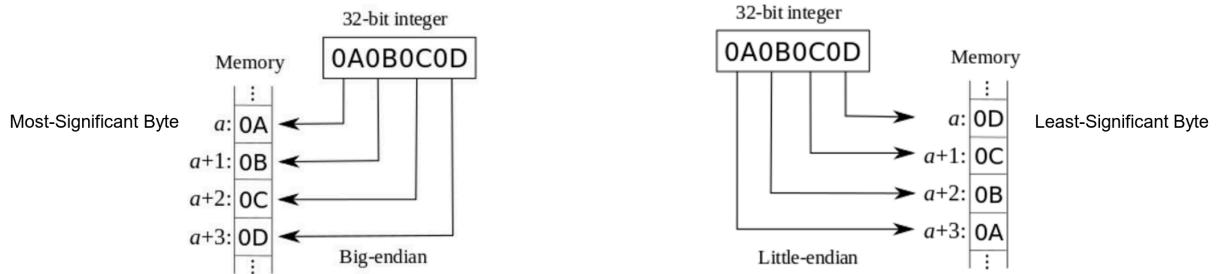
- An address bus is a bus that is used to specify a physical address.
- When a processor or DMA-enabled device needs to read or write to a memory location, it specifies that memory location on the address bus.
- The width of the address bus determines the amount of memory a system can address.
- For example, a system with a 32-bit address bus can address 2^{32} memory locations.
- If each memory location holds one byte, the addressable memory space is about 4 GB.

2.10.2 Data bus

- A data bus is a bus that transfers data between different components inside a computer or between computers.
- The amount of bytes a memory unit can read or write is dependent on the number of lines of the data bus, rather than the address bus.

2.11 Endianness of memory

- **Endianness** is the ordering or sequencing of bytes of a word of digital data in computer memory storage or during transmission.
- Words may be represented in **big-endian** or **little-endian** manner.
- **Big-endian** systems store the **most-significant** byte of a word at the **smallest** memory address and the **least-significant** byte at the **largest**.
- **Little-endian** systems store the **least-significant** byte of a word at the **smallest** memory address and the **most-significant** byte at the **largest**.



2.12 Base address

The base address is a common reference for a series of related addresses.

2.13 Address offset

The address of set is an individual offset from a common reference address.

2.14 Address indexing

Address indexing refers to the operation of adding an offset to a base address.

$$\text{Address indexing} = \text{Base address} + \text{Offset} = [\text{Base address}, \text{offset}]$$

2.15 Configuration register

A configuration register is a register with programmable switches.

2.16 Maskable data register

A maskable data register is a data register that has a pin selection mask.

2.16.1 Example

- GPIO_PORTF_DIR_R EQU 0x40025 400 is the GPIO direction (input or output).
- GPIO_PORTF_AFSEL_R EQU 0x40025 420 is the GPIO alternate function select.
- GPIO_PORTF_PUR_R EQU 0x40025 510 is the GPIO pull up select (whether to enable the pull-up resistors or not).
- GPIO_PORTF_DEN_R EQU 0x40025 51C is the GPIO digital enable.
- GPIO_PORTF_AMSEL_R EQU 0x40025 528 is the GPIO analogue mode select.
- GPIO_PORTF_PCTL_R EQU 0x40025 52C is the GPIO port control.
- PF1 EQU 0x40025 008 is the GPIO “port” address.
- PF2 EQU 0x40025 010 is the GPIO “port” address.
- PF3 EQU 0x40025 020 is the GPIO “port” address.
- PF4 EQU 0x40025 040 is the GPIO “port” address.
- PFA EQU 0x40025 038 is the address of 3 ports, PF1 - PF3.
- SYSCTL_RCGCGPIO_R EQU 0x400FE608 is the run mode clock gating control.

2.16.2 Why have maskable data registers?

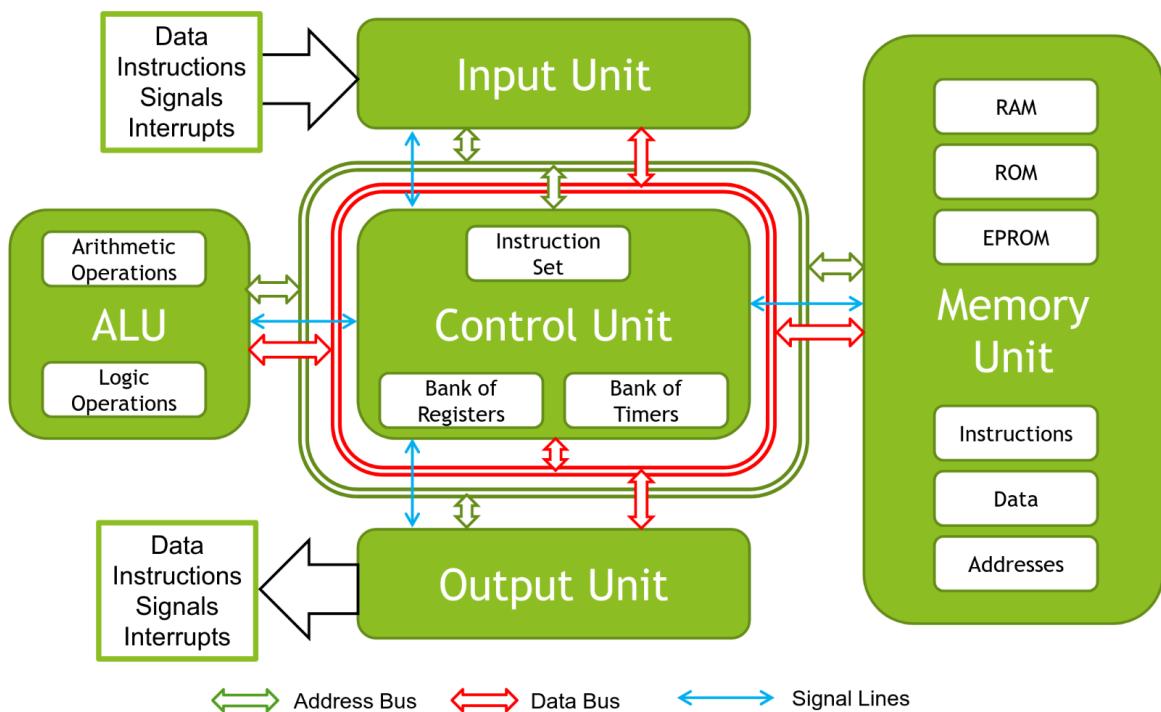
1. Security in control, to avoid situations where the intention is to turn on one motor, but several other motors have been turned on instead.
2. Security in communication, to stop others from interpreting the intercepted data in a communication. This is possible because intercepted data is not filtered by any specific mask, and received data is filtered by a specific mask.

3 Basics of brains

3.1 Characteristics of the human brain

- Able to memorise past, present and future.
- Able to capture input data.
- Able to undertake operations with numbers or symbols.
- Able to produce output data.
- Able to control mental operations.
- Able to control physical actions.
- Able to communicate.
- Able to learn human languages.
- Able to learn machine languages.
- Able to learn new knowledge and new skills.

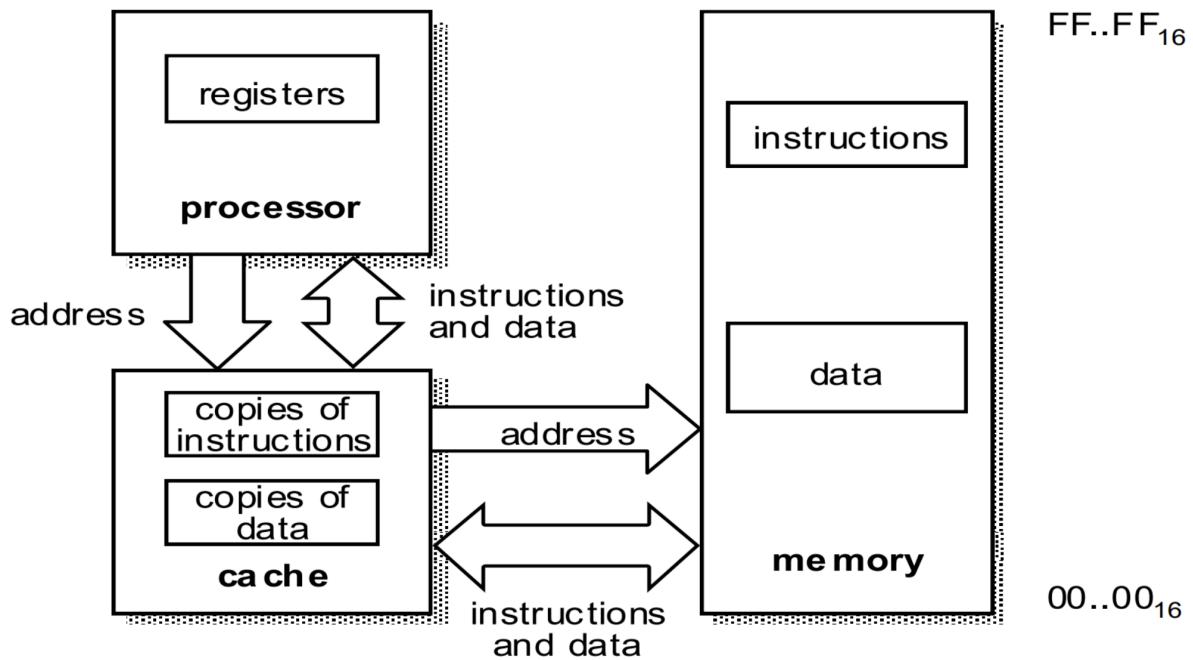
3.2 Blueprint of a machine brain



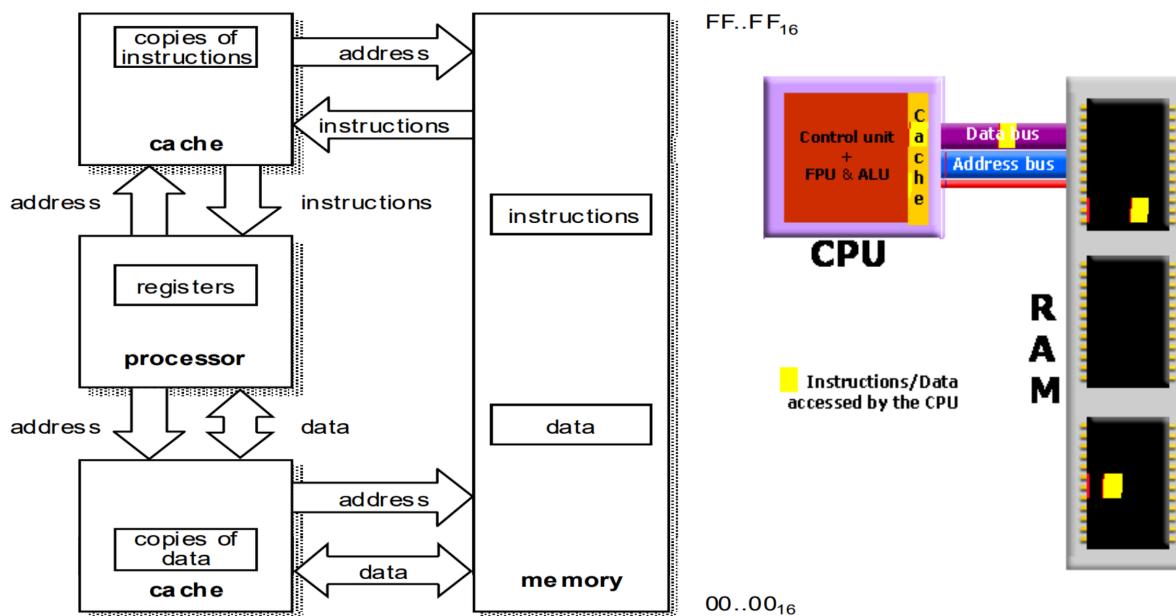
3.3 Characteristics of a machine brain

- Able to memorise data (to use) and instructions (to do)
- Able to capture input data.
- Able to undertake arithmetics operations with numbers.
- Able to undertake logic operations with numbers and symbols.
- Able to produce output data.
- Able to control mental operations.
- Able to control physical actions.
- Able to communicate.
- Able to learn human languages in the future.
- Able to learn machine languages in the future.
- Able to learn new knowledge and new skills in the future.

3.4 Van Neumann architecture



3.5 Harvard architecture

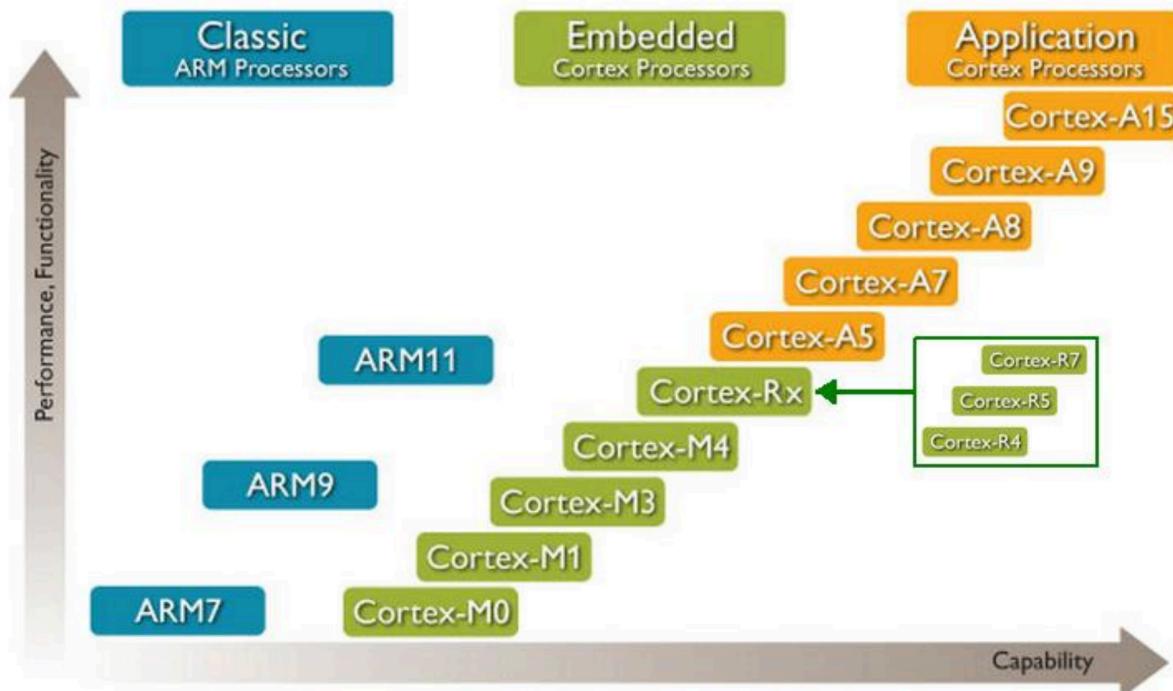


4 ARM: Advanced RISC Machines

- ARM Ltd was founded in 1990
- ARM Ltd is headquartered in Cambridge, UK

4.1 ARM Family

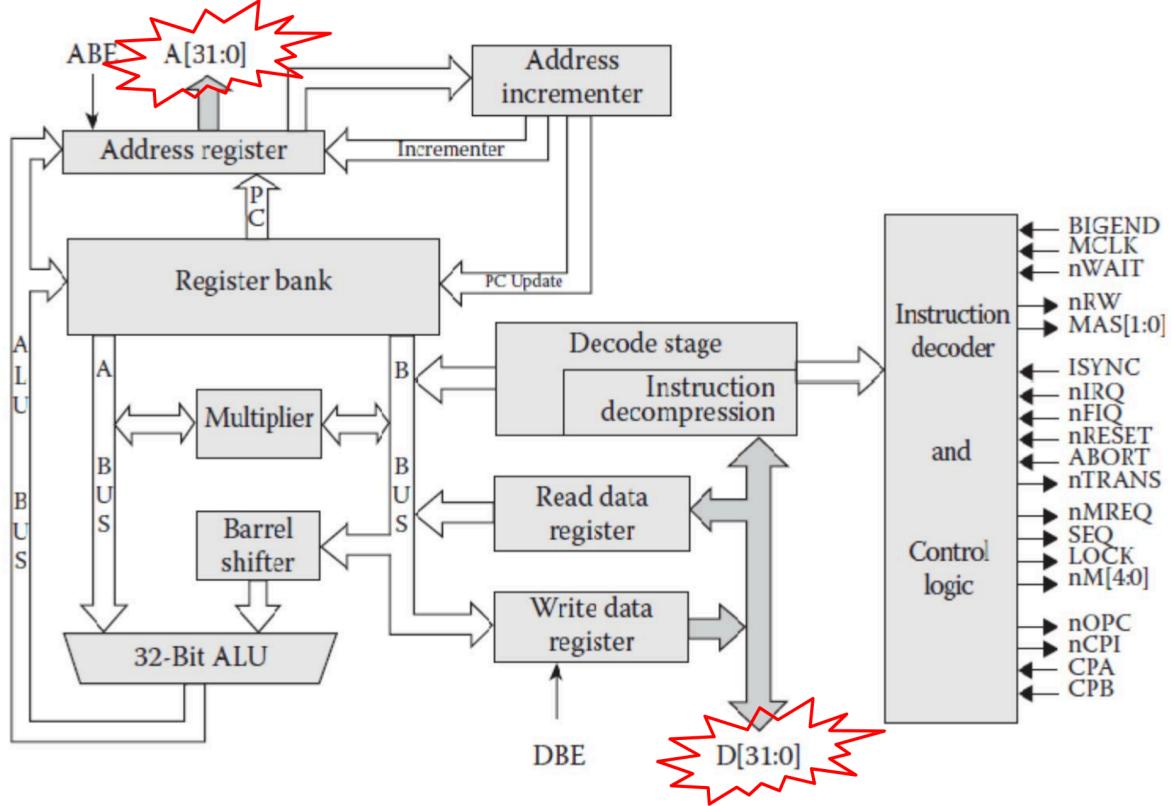
- Cortex-M Series: For microcontroller applications
- Cortex-R Series: For real-time applications
- Cortex-A Series: For advanced applications



4.2 ARM Cortex uses the RISC architecture

- ARM uses the RISC architecture.
 - ISA stands for Instruction Set Architecture which is innate.
 - RISC stands for Reduced Instruction Set Computer.
 - Most instructions execute in a single cycle.
 - ARM instruction set is 32-bit.
 - Thumb instruction set is 16 or 32-bit.
- ARM is a 32-bit load-store architecture.
 - 8 bits are called a byte.
 - 16 bits of 2 bytes are called a half-word.
 - 32 bits or 4 bytes are called a word.
 - 64 bits or 8 bytes are called a double-word.
 - The only memory accesses are loads and stores.

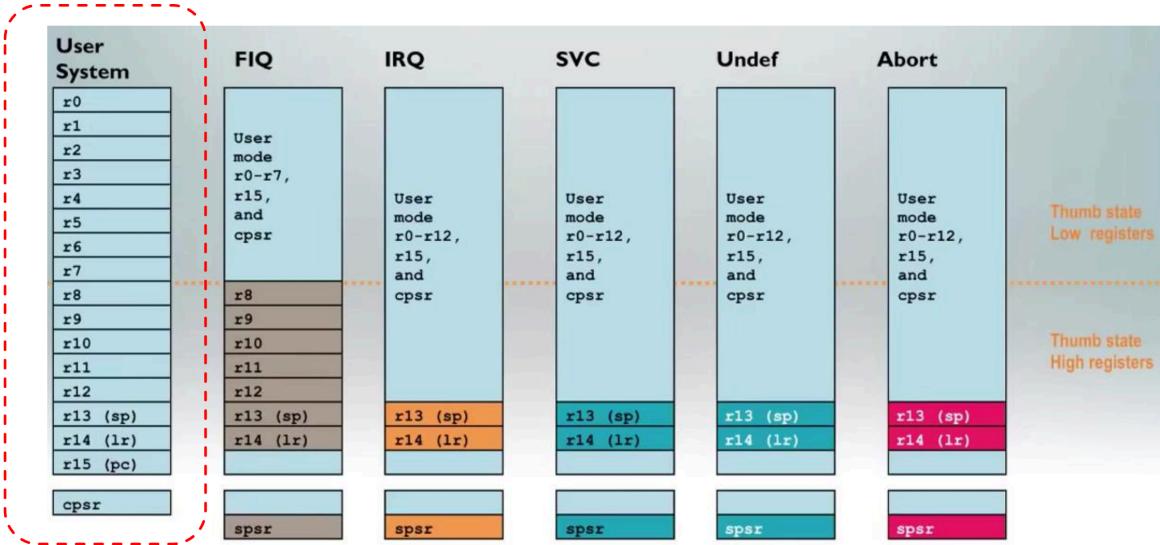
4.2.1 Architecture of the ARM7TDMI Processor



4.3 Modes in ARM Cortex

- **Supervisor (SVC):** Entered on reset and when a Supervisor call instruction (SVC) is executed. Privileged and exception mode.
- **FIQ:** Entered when a high priority (fast) interrupt is raised. Privileged and exception mode.
- **IRQ:** Entered when a normal priority interrupt is raised. Privileged and exception mode.
- **Abort:** Used to handle memory access violations. Privileged and exception mode.
- **Undef:** Used to handle undefined instructions. Privileged and exception mode.
- **System:** Privileged mode using the same registers as User mode.
- **User:** Mode under which most applications and operating system tasks run. Unprivileged mode.

4.4 Registers in ARM Cortex



Where:

- “SPSR” is the saved program status register
- “CPSR” is the current program status register
- “SP” is the stack pointer
- “LR” is the link register

Note that the system mode uses the user mode register set.

4.4.1 Register table

User/System	Supervisor	Abort	Undefined	Interrupt	Fast interrupt
R0	R0	R0	R0	R0	R0
R1	R1	R1	R1	R1	R1
R2	R2	R2	R2	R2	R2
R3	R3	R3	R3	R3	R3
R4	R4	R4	R4	R4	R4
R5	R5	R5	R5	R5	R5
R6	R6	R6	R6	R6	R6
R7	R7	R7	R7	R7	R7
R8	R8	R8	R8	R8	R8_FIQ
R9	R9	R9	R9	R9	R9_FIQ
R10	R10	R10	R10	R10	R10_FIQ
R11	R11	R11	R11	R11	R11_FIQ
R12	R12	R12	R12	R12	R12_FIQ
R13	R13_SVC	R13_ABORT	R13_UNDEF	R13_IRQ	R13_FIQ
R14	R14_SVC	R14_ABORT	R14_UNDEF	R14_IRQ	R14_FIQ
PC	PC	PC	PC	PC	PC

CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
SPSR_SVC	SPSR_ABORT	SPSR_UNDEF	SPSR_IRQ	SPSR_FIQ	

= banked register

4.4.2 Current program status register (CPSR, R16)

31	30	29	28	27	26	25	24	23	20	19	16	15	10	9	8	7	6	5	4	0
N	Z	C	V	Q	Res	J	RESERVED	GE[3:0]	RESERVED	E	A	I	F	T	M[4:0]					

Where GE stands for greater than or equal status flags.

Overview:

Field	Description	Architecture
N Z C V	Condition code flags	All
J	Jazelle state flag	5TEJ and above
GE[3:0]	SIMD condition flags	6
E	Endian load/store	6
A	Imprecise abort mask	6
I	IRQ interrupt mask	All
F	FIQ interrupt mask	All
T	Thumb state flag	4T and above
Mode[4:0]	Processor mode	All

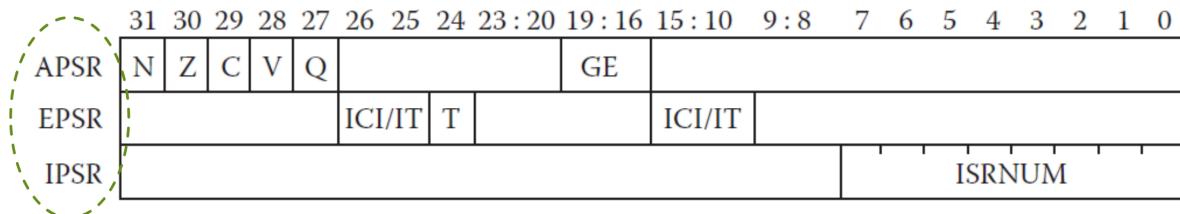
Flag field	Description
N	Negative result from the ALU
Z	Zero result from the ALU
C	ALU operation caused carry
V	ALU operation overflowed
Q	ALU operation saturated
J	Java byte code execution

,

Control bits	Description
I	1: Disables IRQ
F	1: Disables FIQ
T	1: Thumb, 0: ARM

Mode bits	Description
0b10000	User
0b11111	System
0b10001	FIQ
0b10010	IRQ
0b10011	SVC (Supervisor)
0b10111	Abort
0b11011	Undefined

4.4.3 Cortex M4's status register



Where:

- APSR stands for application program status register
- EPSR stands for execution program status register
- IPSR stands for interrupt program status register

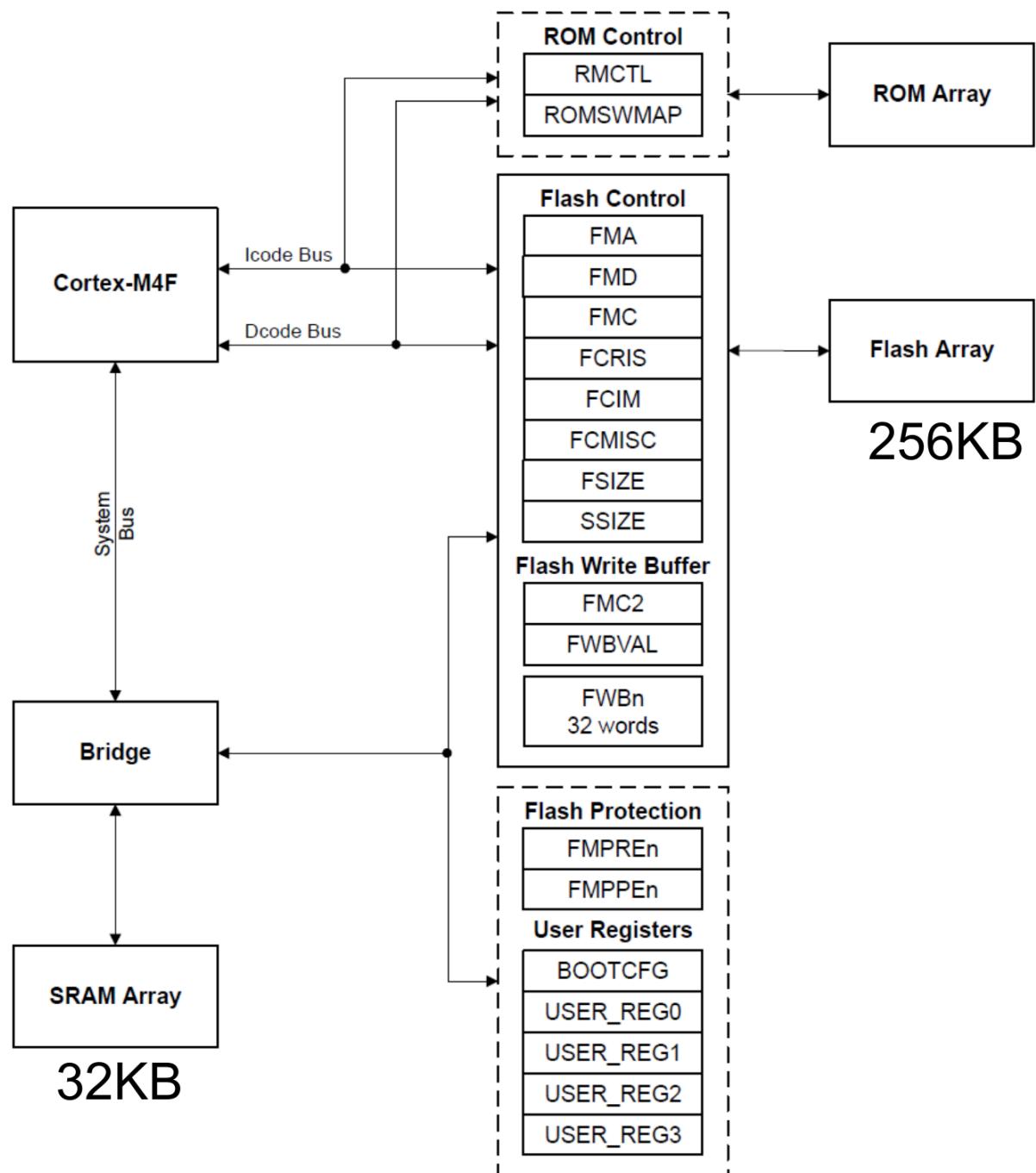
4.5 ARM Cortex as a microcontroller

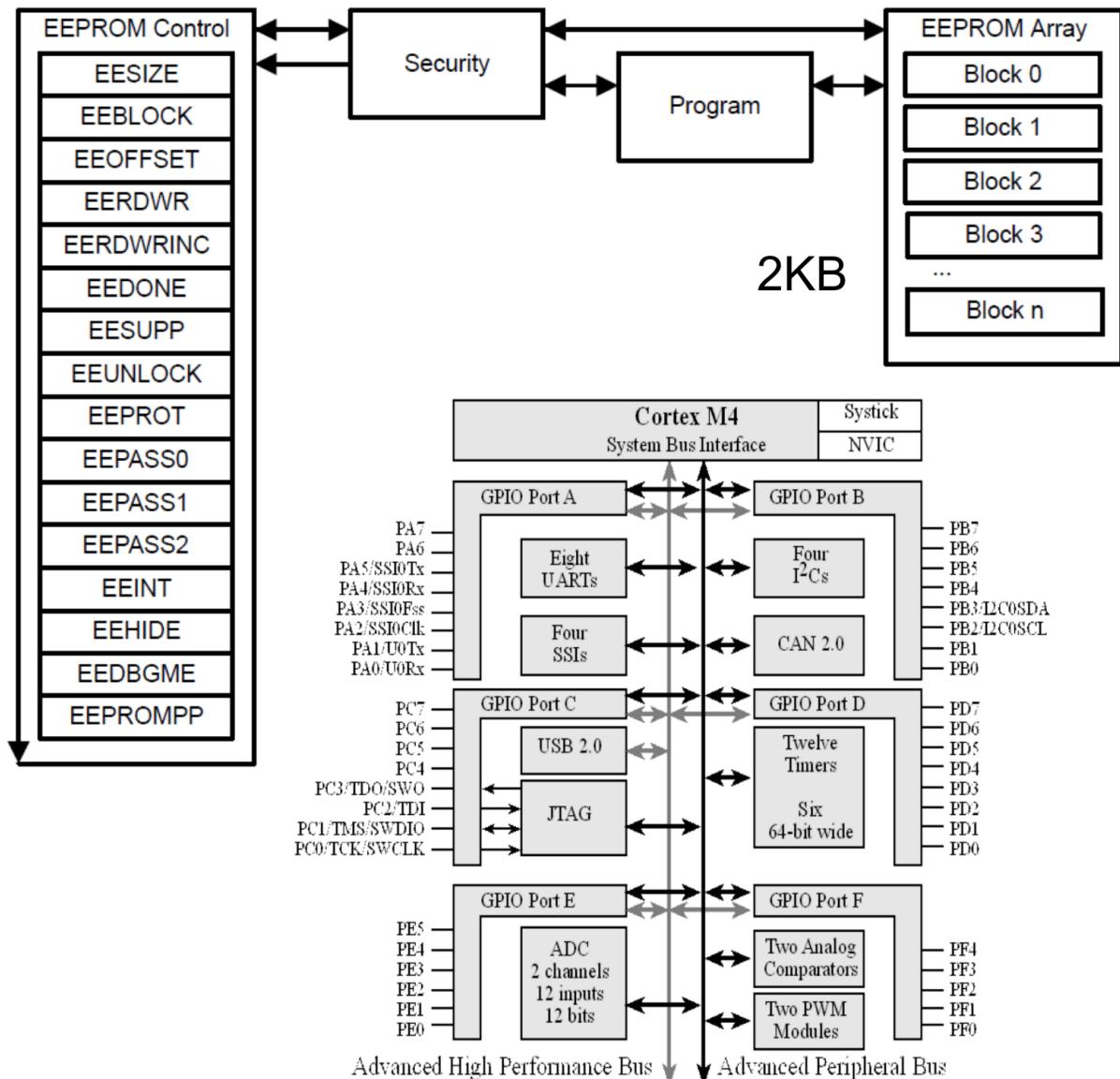
- ARM supports application profile.
 - Memory Management Unit (MMU).
 - Highest performance at lowest power consumption.
 - Trust zone for a safe and extensible system.
- ARM supports real-time profile
 - Memory Protection Unit (MPU).
 - Low latency and predictability of real-time needs.
 - Tightly coupled memory for fast and deterministic access.
- ARM supports microcontroller profile
 - Lowest gate count entry point.
 - Deterministic and predictable behaviour.
 - Deeply embedded use.

4.6 ARM features

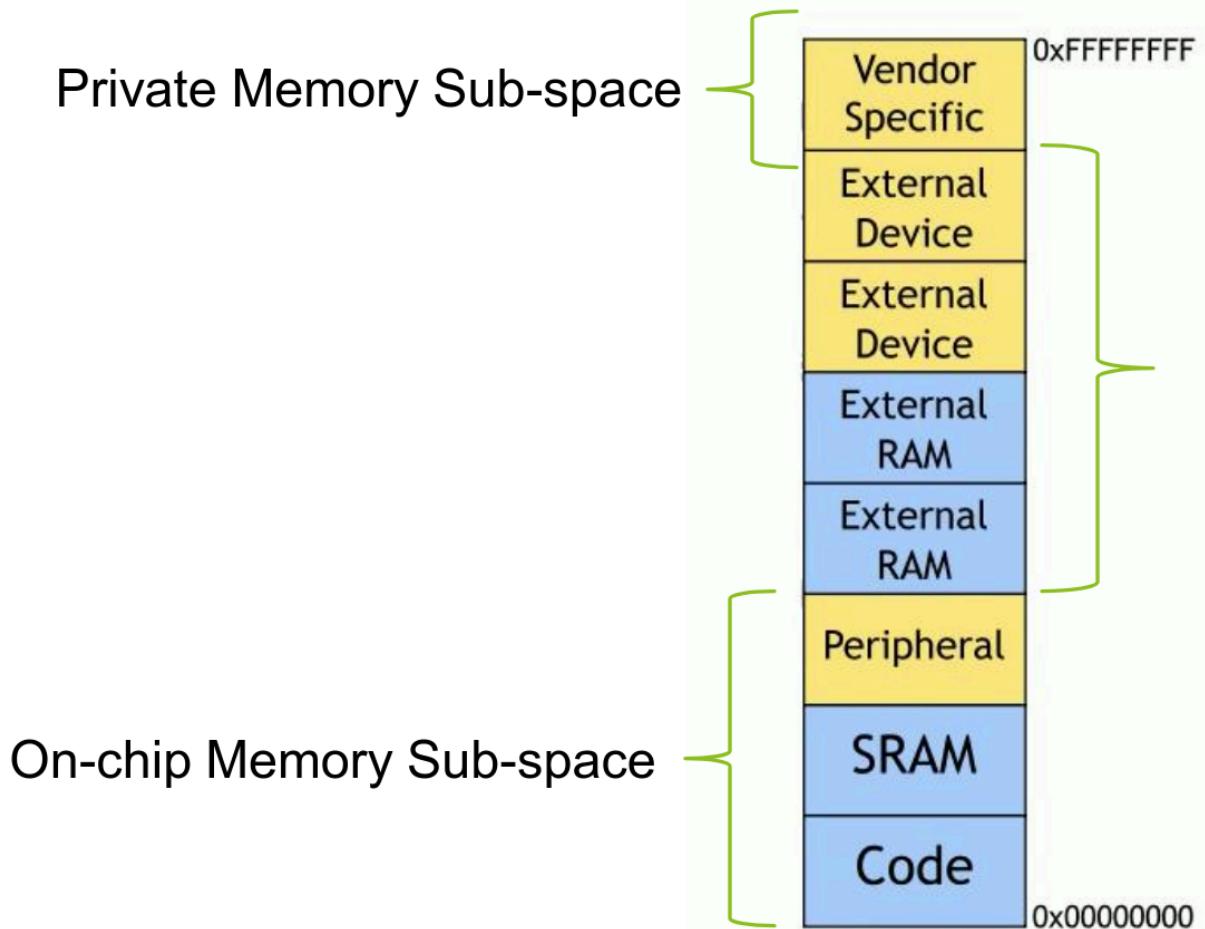
- ARM has two internal buses.
 - The ARM Advanced Microcontroller Bus Architecture (AMBA) is an open-standard, on-chip interconnect specification for the connection and management of functional blocks in system-on-a-chip (SoC) designs. It facilitates development of multiprocessor designs with large numbers of controllers and peripherals.
- ARM Cortex supports direct memory access.
- ARM Cortex includes debug port.
 - JTAG (Joint Test Action Group) specifies the use of a dedicated debug port implemented a serial communication interface for low-overhead access without requiring direct external access to the system address and data buses.

4.7 Memory inside TM4C123G



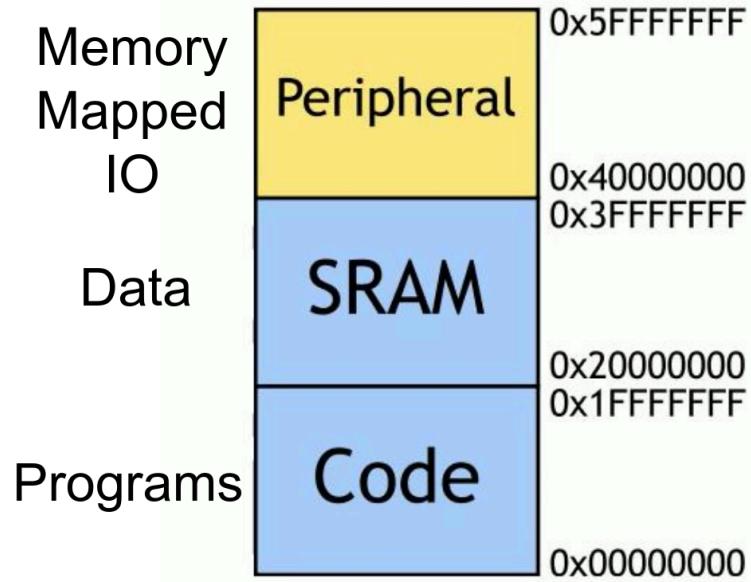


4.8 ARM Cortex M's memory space



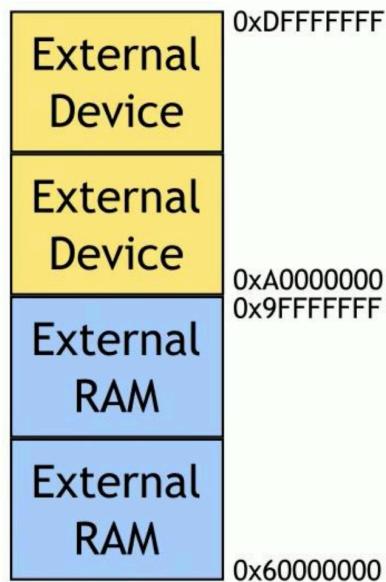
- 32-bit addresses support 4 GiB memory space.
- Code, data and I/O share the same memory space.
- Data types are **bytes**, **half-words**, and **words**.
- Predefined regions have distinct characteristics.
 - Executable
 - Device or strongly-ordered
 - Shareable

4.8.1 On-chip memory subspace



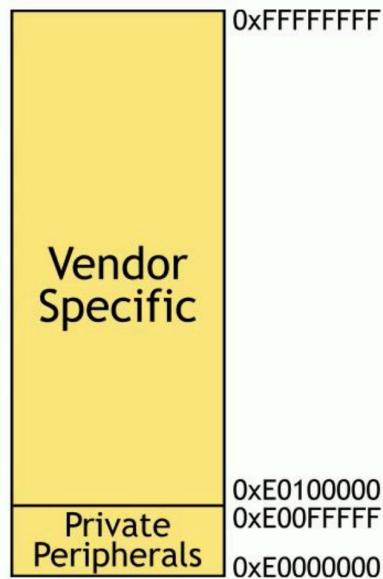
- On-chip code, data and I/O are located in the first 1.5 GiB of memory space.
- Each is allocated 0.5 GiB.
- May use physically separate buses for each space.

4.8.2 Off-chip memory subspace



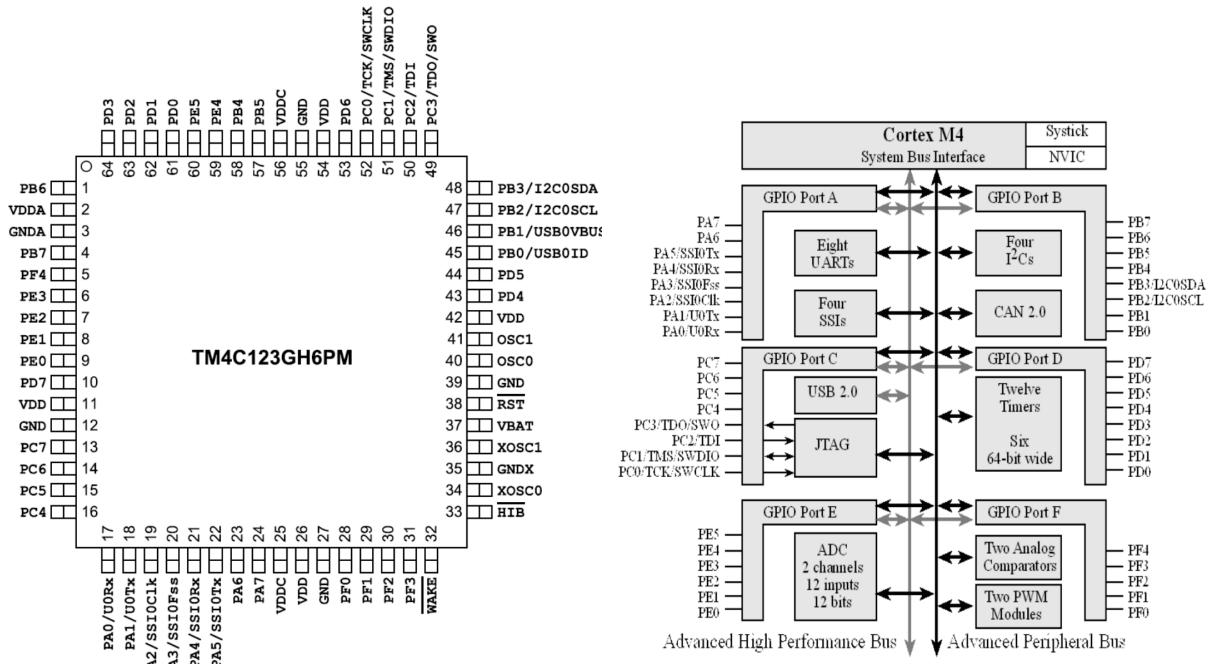
- 1 GiB reserved for each of off-chip data and off-chip devices.
- Off-chip memory bus requires many pins.
- Off-chip memory access time is usually slower and uses more power.

4.8.3 Private memory subspace



- Private peripheral bus occupies 1 MiB of space.
- Registers that control peripherals that are a mandatory part of the Cortex M architecture are mapped here.
 - Nested vectored interrupt controller (NVIC)
 - System tick timer (SysTick)
 - Fault status and control
 - Processor debugging

4.9 Cortex M4 pin diagram



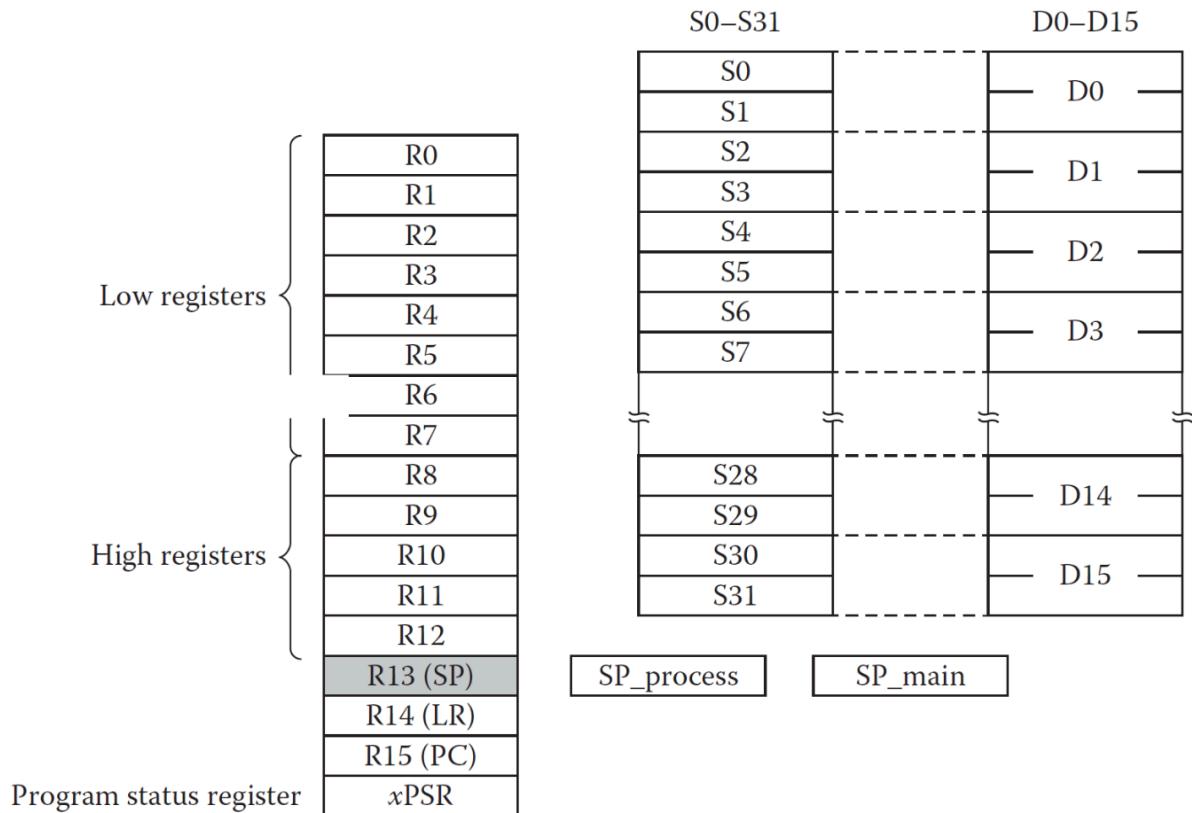
4.10 Cortex M4 table of pin usage

IO	Pin	Analog Function	Digital Function (GPIO PCTL PMCx Bit Field Encoding) ^a											
			1	2	3	4	5	6	7	8	9	14	15	
PA0	17	-	U0Rx	-	-	-	-	-	-	CAN1Rx	-	-	-	
PA1	18	-	U0Tx	-	-	-	-	-	-	CAN1Tx	-	-	-	
PA2	19	-	-	SSI0Clk	-	-	-	-	-	-	-	-	-	
PA3	20	-	-	SSI0Fss	-	-	-	-	-	-	-	-	-	
PA4	21	-	-	SSI0Rx	-	-	-	-	-	-	-	-	-	
PA5	22	-	-	SSI0Tx	-	-	-	-	-	-	-	-	-	
PA6	23	-	-	-	I2C1SCL	-	M1PWM2	-	-	-	-	-	-	
PA7	24	-	-	-	I2C1SDA	-	M1PWM3	-	-	-	-	-	-	
PB0	45	USB0ID	U1Rx	-	-	-	-	-	-	T2CCP0	-	-	-	
PB1	46	USB0VBUS	U1Tx	-	-	-	-	-	-	T2CCP1	-	-	-	
PB2	47	-	-	-	I2C0SCL	-	-	-	-	T3CCP0	-	-	-	
PB3	48	-	-	-	I2C0SDA	-	-	-	-	T3CCP1	-	-	-	
PB4	58	AIN10	-	SSI2Clk	-	M0PWM2	-	-	-	T1CCP0	CAN0Rx	-	-	
PB5	57	AIN11	-	SSI2Fss	-	M0PWM3	-	-	-	T1CCP1	CAN0Tx	-	-	
PB6	1	-	-	SSI2Rx	-	M0PWM0	-	-	-	T0CCP0	-	-	-	
PB7	4	-	-	SSI2Tx	-	M0PWM1	-	-	-	T0CCP1	-	-	-	

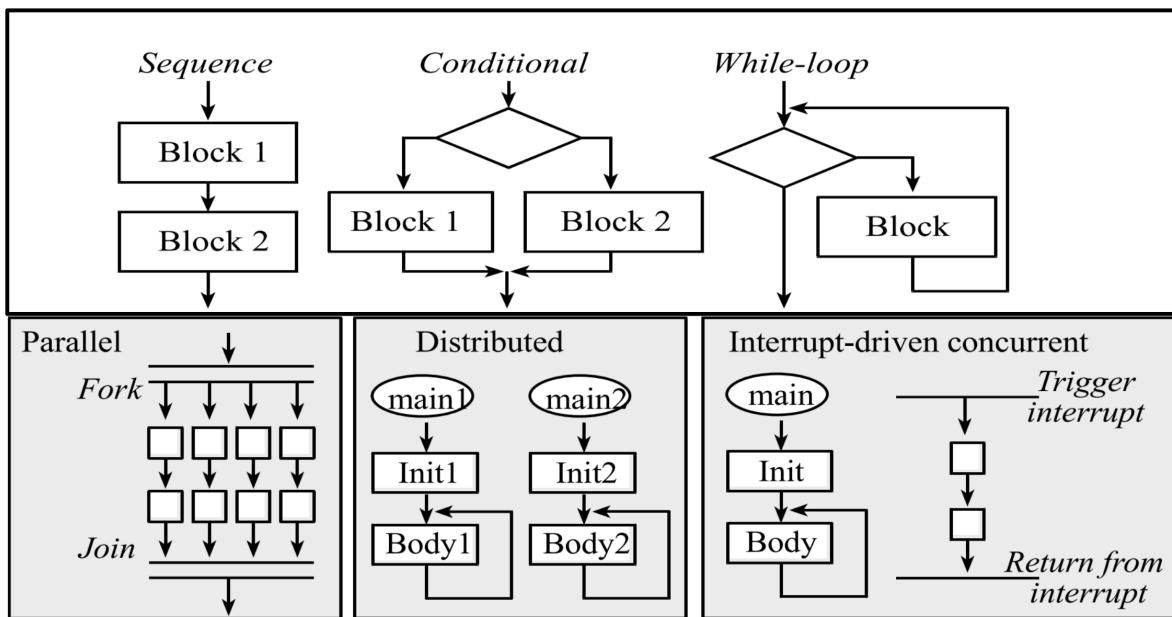
IO	Pin	Analog Function	Digital Function (GPIO PCTL PMCx Bit Field Encoding) ^a											
			1	2	3	4	5	6	7	8	9	14	15	
PC0	52	-	TCK SWCLK	-	-	-	-	-	-	T4CCP0	-	-	-	
PC1	51	-	TMS SWDIO	-	-	-	-	-	-	T4CCP1	-	-	-	
PC2	50	-	TDI	-	-	-	-	-	-	T5CCP0	-	-	-	
PC3	49	-	TDO SWO	-	-	-	-	-	-	T5CCP1	-	-	-	
PC4	16	C1-	U4Rx	U1Rx	-	M0PWM6	-	IDX1	WT0CCP0	U1RTS	-	-	-	
PC5	15	C1+	U4Tx	U1Tx	-	M0PWM7	-	PhA1	WT0CCP1	U1CTS	-	-	-	
PC6	14	C0+	U3Rx	-	-	-	-	PhB1	WT1CCP0	USB0EPEN	-	-	-	
PC7	13	C0-	U3Tx	-	-	-	-	-	WT1CCP1	USB0PFLT	-	-	-	
PD0	61	AIN7	SSI3Clk	SSI1Clk	I2C3SCL	M0PWM6	M1PWM0	-	WT2CCP0	-	-	-	-	
PD1	62	AIN6	SSI3Fss	SSI1Fss	I2C3SDA	M0PWM7	M1PWM1	-	WT2CCP1	-	-	-	-	
PD2	63	AIN5	SSI3Rx	SSI1Rx	-	M0FAULT0	-	-	WT3CCP0	USB0EPEN	-	-	-	
PD3	64	AIN4	SSI3Tx	SSI1Tx	-	-	-	IDX0	WT3CCP1	USB0PFLT	-	-	-	
PD4	43	USB0DM	U6Rx	-	-	-	-	-	WT4CCP0	-	-	-	-	
PD5	44	USB0DP	U6Tx	-	-	-	-	-	WT4CCP1	-	-	-	-	
PD6	53	-	U2Rx	-	-	M0FAULT0	-	PhA0	WT5CCP0	-	-	-	-	
PD7	10	-	U2Tx	-	-	-	-	PhB0	WT5CCP1	NMI	-	-	-	

IO	Pin	Analog Function	Digital Function (GPIO_PCTL PMCx Bit Field Encoding) ^a										
			1	2	3	4	5	6	7	8	9	14	15
PE0	9	AIN3	U7RX	-	-	-	-	-	-	-	-	-	-
PE1	8	AIN2	U7TX	-	-	-	-	-	-	-	-	-	-
PE2	7	AIN1	-	-	-	-	-	-	-	-	-	-	-
PE3	6	AIN0	-	-	-	-	-	-	-	-	-	-	-
PE4	59	AIN9	U5RX	-	I2C2SCL	M0PWM4	M1PWM2	-	-	CAN0Rx	-	-	-
PE5	60	AIN8	U5Tx	-	I2C2SDA	M0PWM5	M1PWM3	-	-	CAN0Tx	-	-	-
PF0	28	-	U1RTS	SSI1Rx	CAN0RX	-	M1PWM4	PhA0	T0CCP0	NMI	C0o	-	-
PF1	29	-	U1CTS	SSI1Tx	-	-	M1PWM5	PhB0	T0CCP1	-	C1o	TRD1	-
PF2	30	-	-	SSI1Clk	-	M0FAULT0	M1PWM6	-	T1CCP0	-	-	TRD0	-
PF3	31	-	-	SSI1Fss	CAN0Tx	-	M1PWM7	-	T1CCP1	-	-	TRCLK	-
PF4	5	-	-	-	-	-	M1FAULT0	IDX0	T2CCP0	USB0EPEN	-	-	-

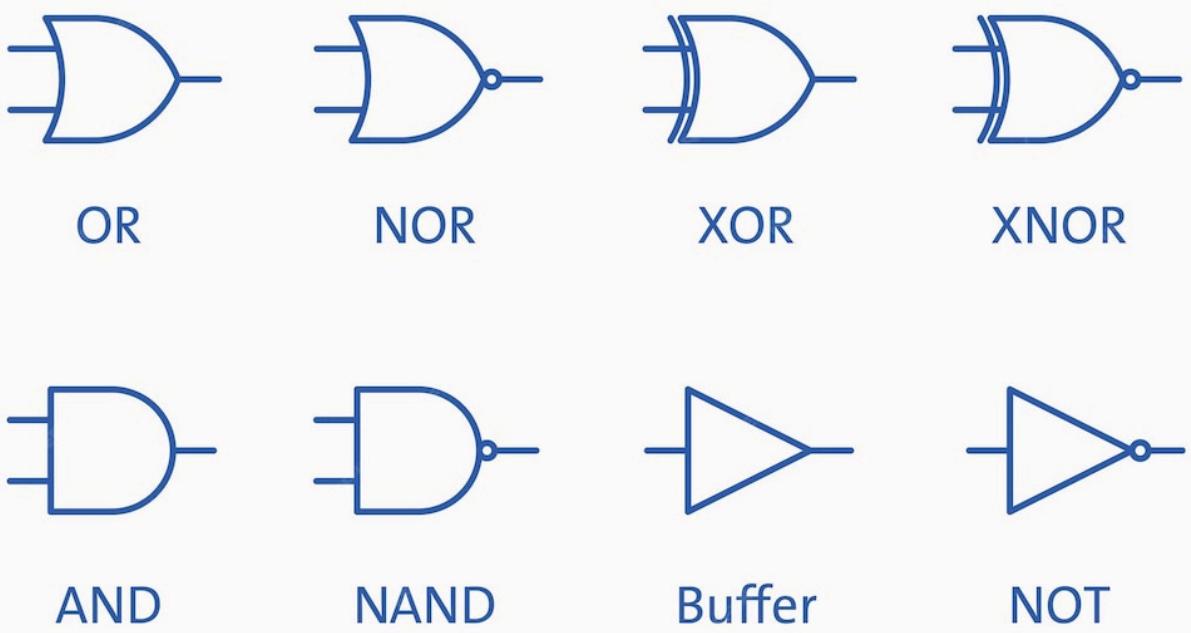
4.11 Cortex M4's floating point unit's registers



5 Typical types of application software



6 Binary logic devices



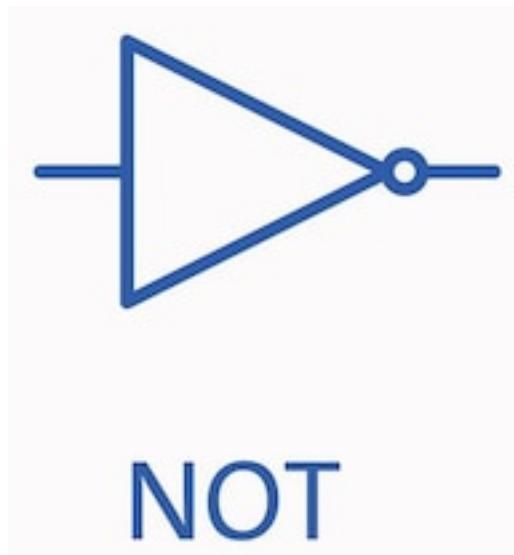
6.1 Binary number systems

Binary values	Implementation
1 (logic high)	+5 V (logic high)
0 (logic low)	0 V (logic low)

6.2 Binary logic operations

Operation	Boolean expressions				
NOT	A'	$\neg A$	\bar{A}	$\neg A$	
AND	AB	A^*B	$A \cdot B$	$A \wedge B$	$A \cap B$
OR	$A + B$	$A \vee B$	$A \cup B$		
NAND	$(AB)'$	\overline{AB}			
NOR	$(A + B)'$	$\overline{A + B}$			
XOR	$A \oplus B$	$A @ B$			
XNOR	$(A \oplus B)'$	$\overline{A \oplus B}$	$(A @ B)'$	$\overline{A @ B}$	

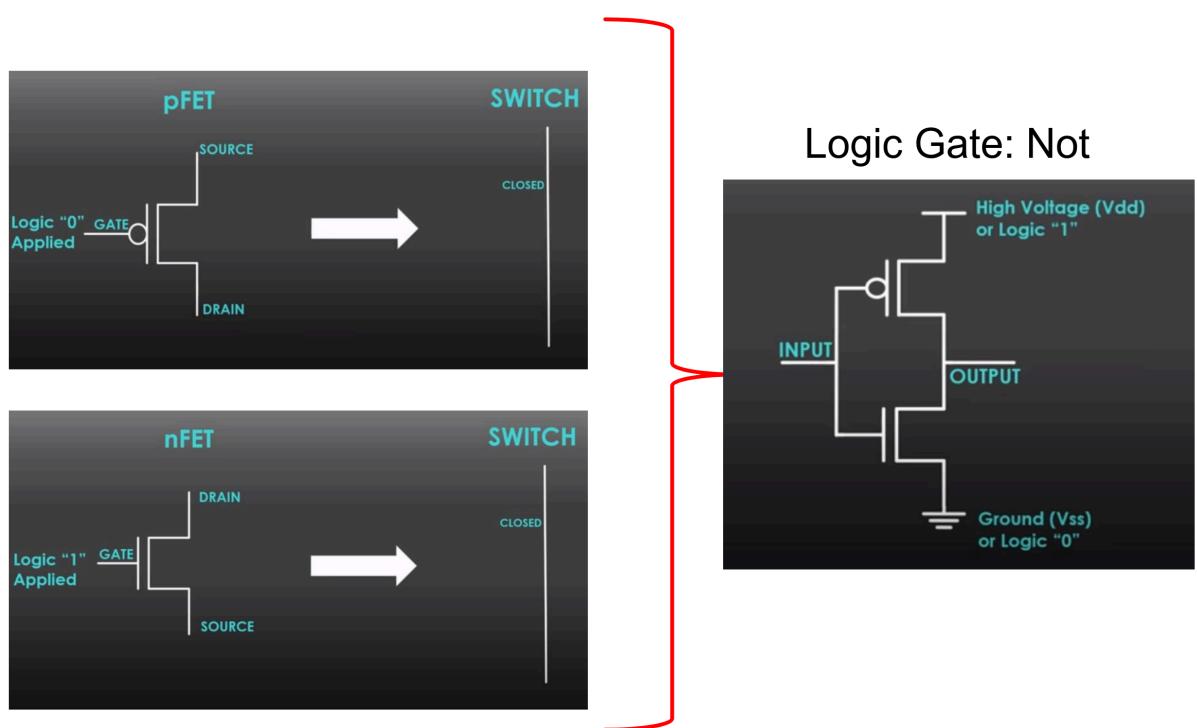
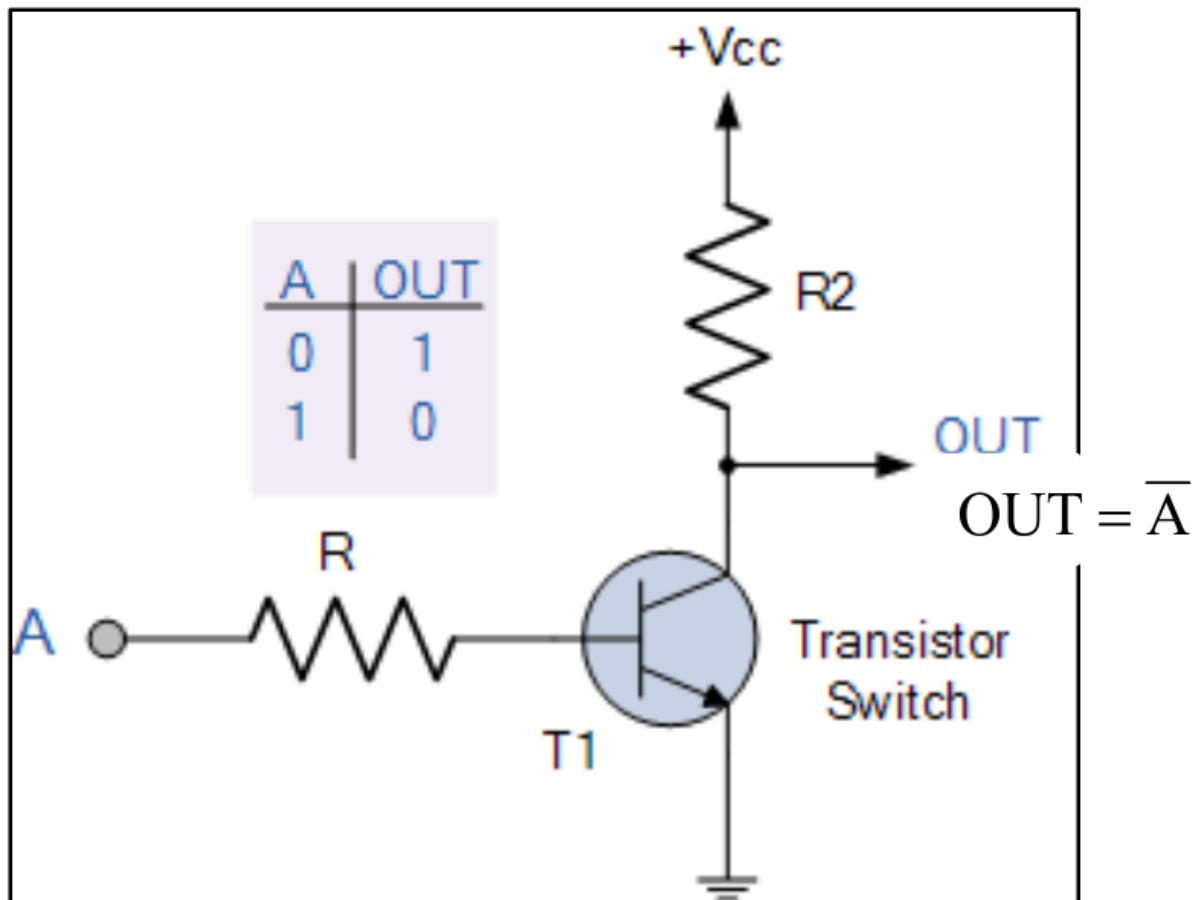
6.3 NOT logic gate



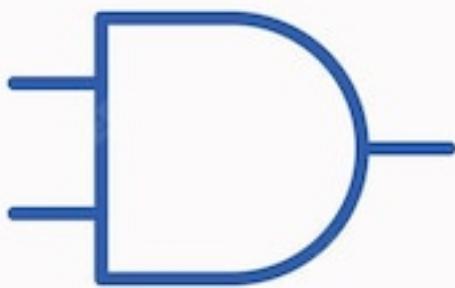
6.3.1 Truth table

Input (A)	Output (\bar{A})
1	0
0	1

6.3.2 Circuit diagrams



6.4 AND logic gate

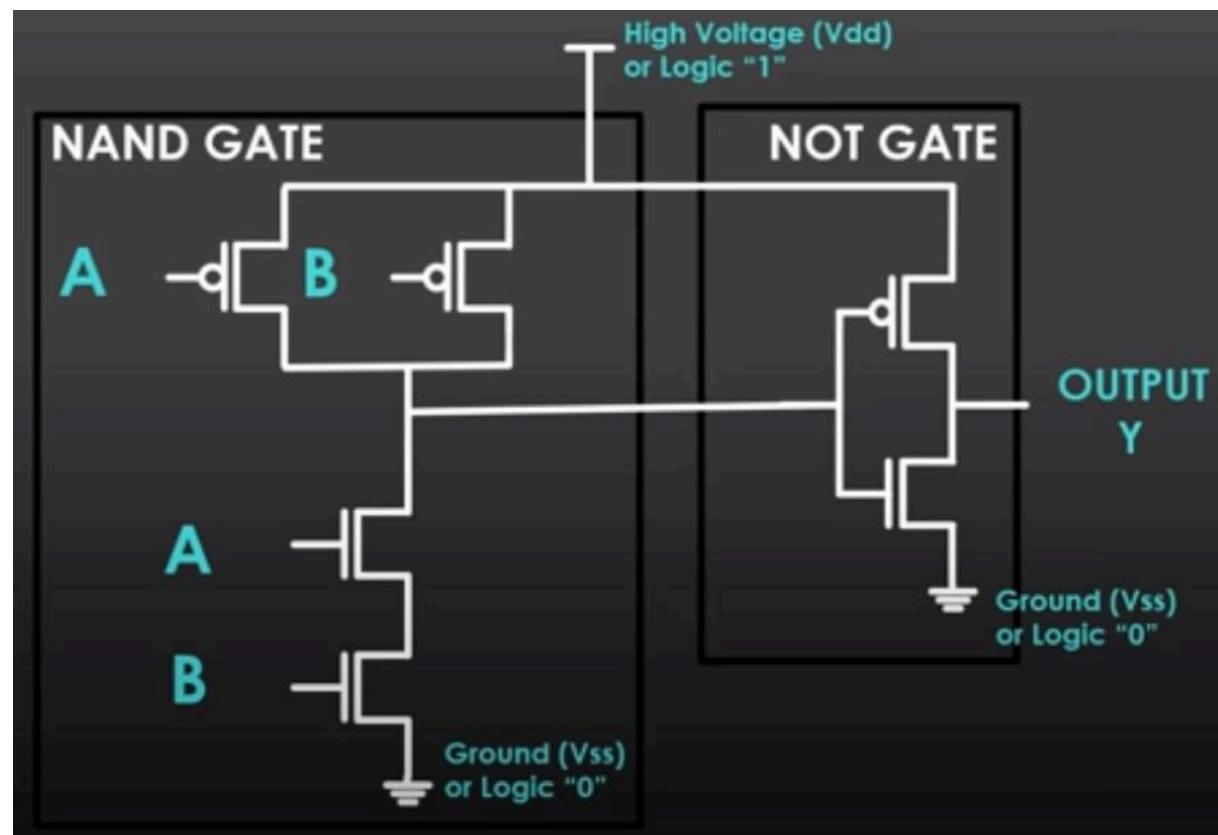
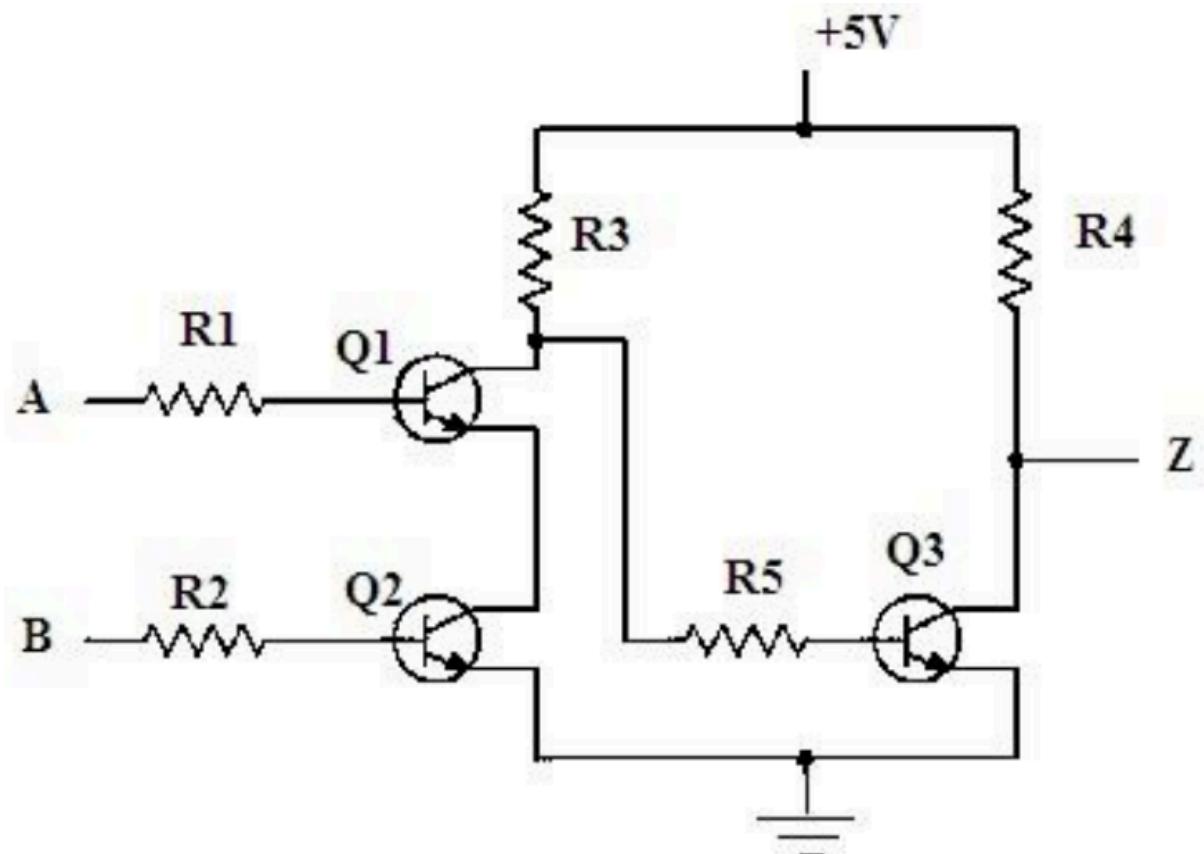


AND

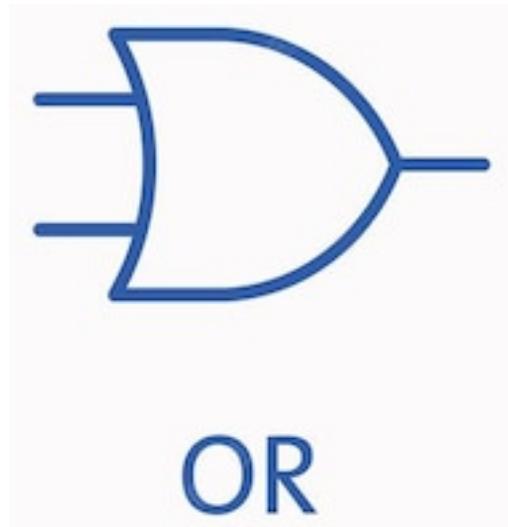
6.4.1 Truth table

Input 1 (X)	Input 2 (Y)	Output ($X \cdot Y$)
0	0	0
0	1	0
1	0	0
1	1	1

6.4.2 Circuit diagrams



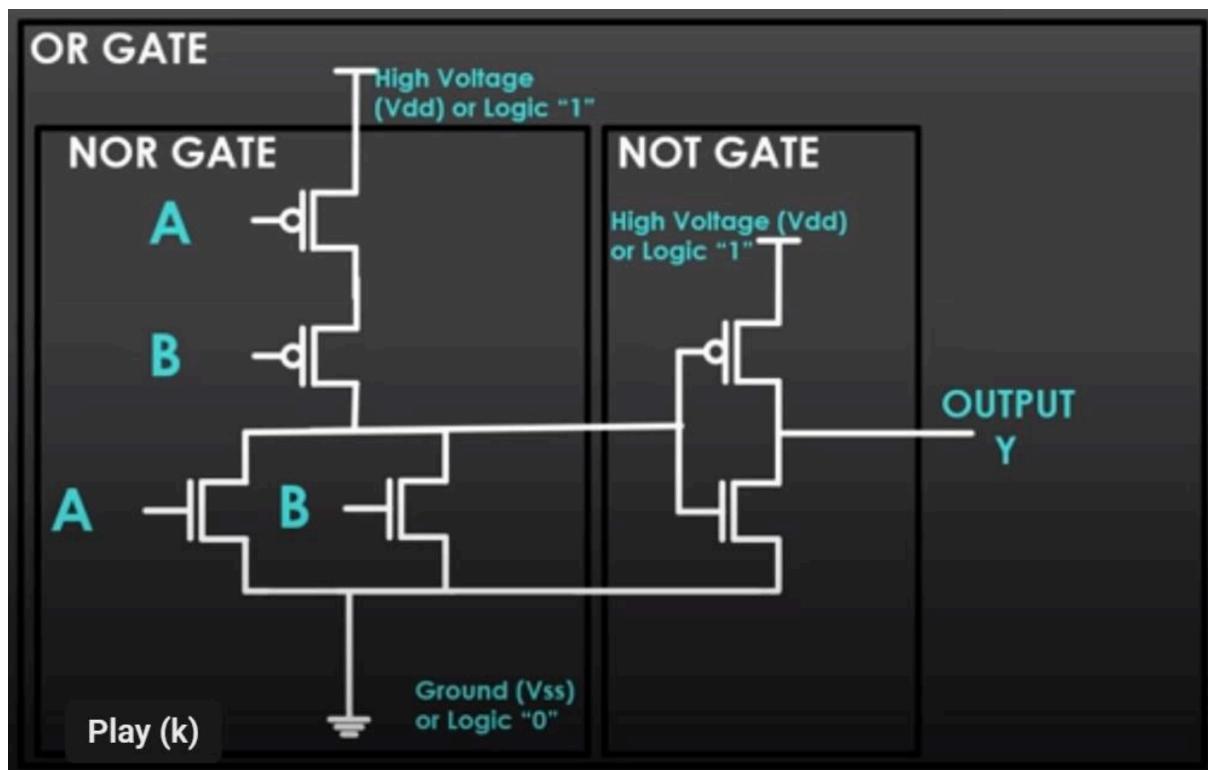
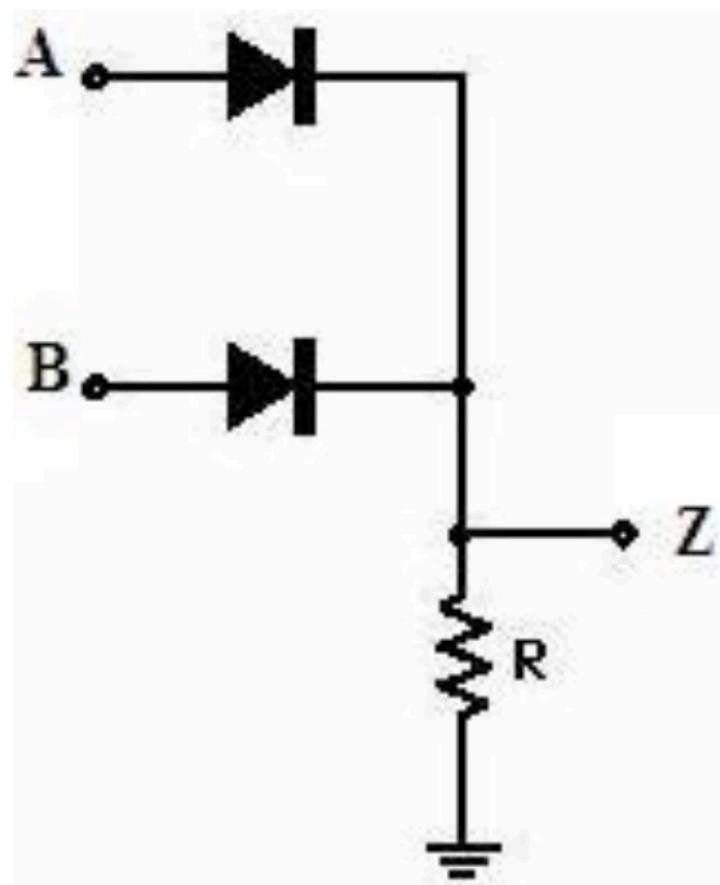
6.5 OR logic gate



6.5.1 Truth table

Input 1 (X)	Input 2 (Y)	Output ($X + Y$)
0	0	0
0	1	1
1	0	1
1	1	1

6.5.2 Circuit diagrams



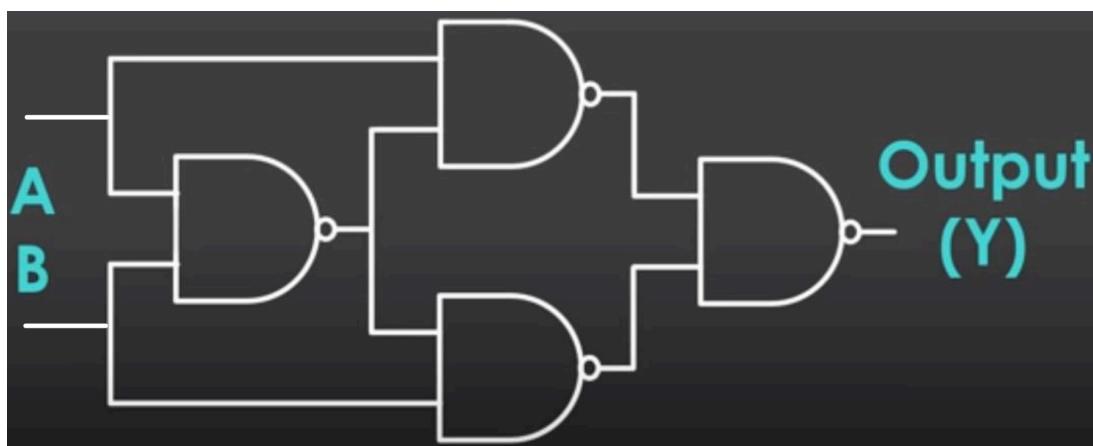
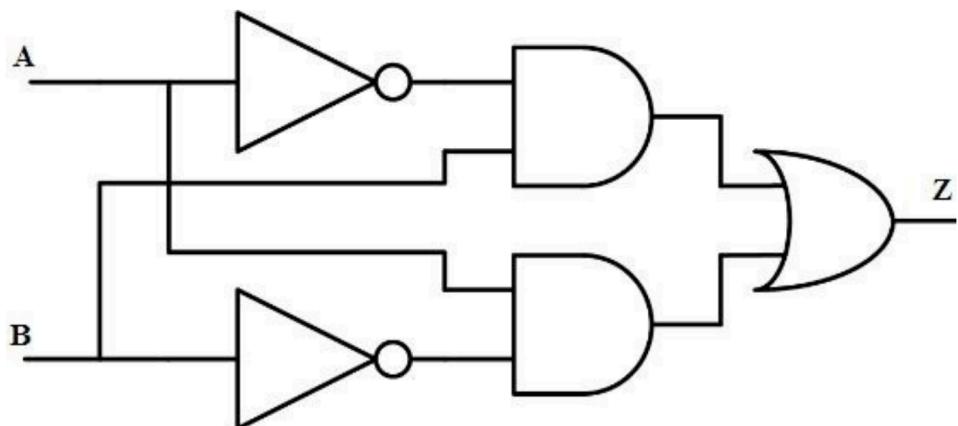
6.6 XOR logic gate



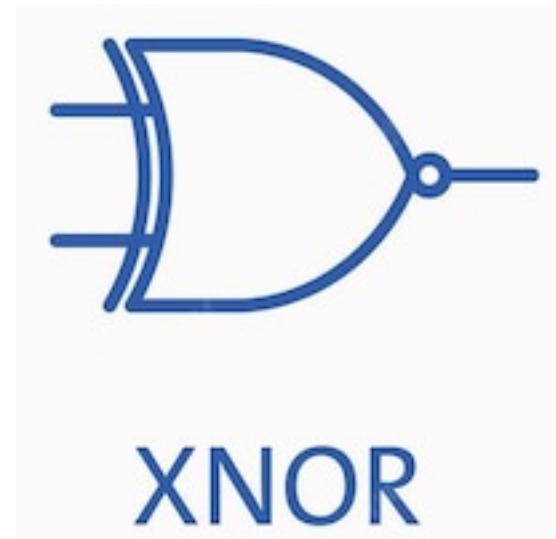
6.6.1 Truth table

Input 1 (X)	Input 2 (Y)	Output ($X \oplus Y$)
0	0	0
0	1	1
1	0	1
1	1	0

6.6.2 Circuit diagrams



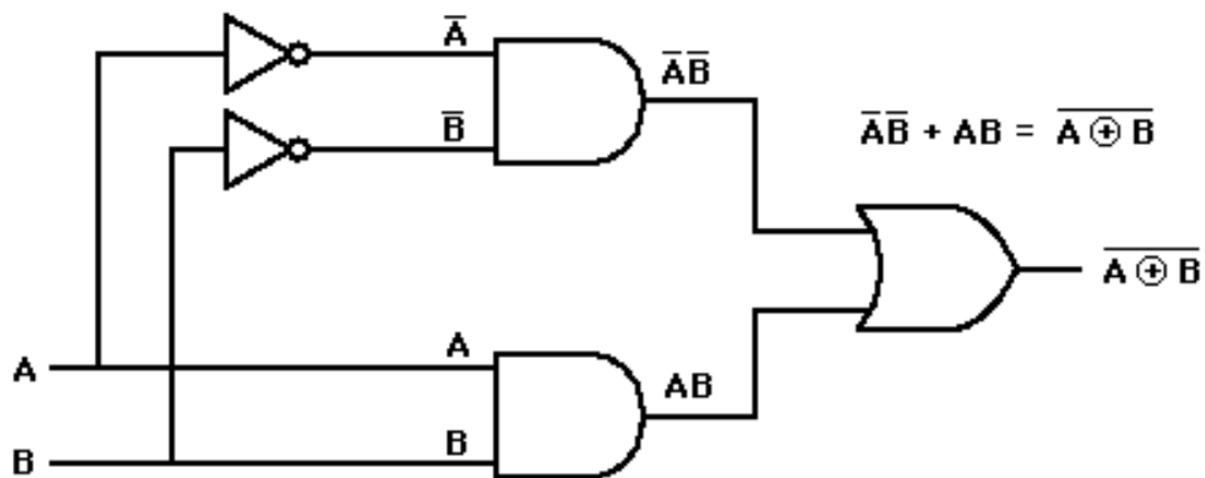
6.7 XNOR logic gate



6.7.1 Truth table

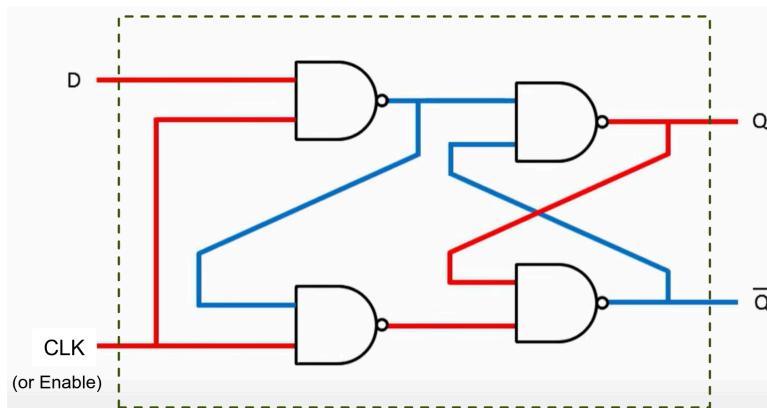
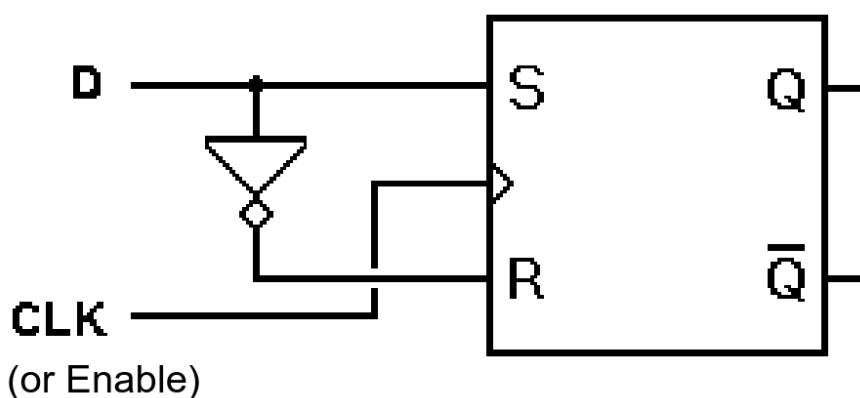
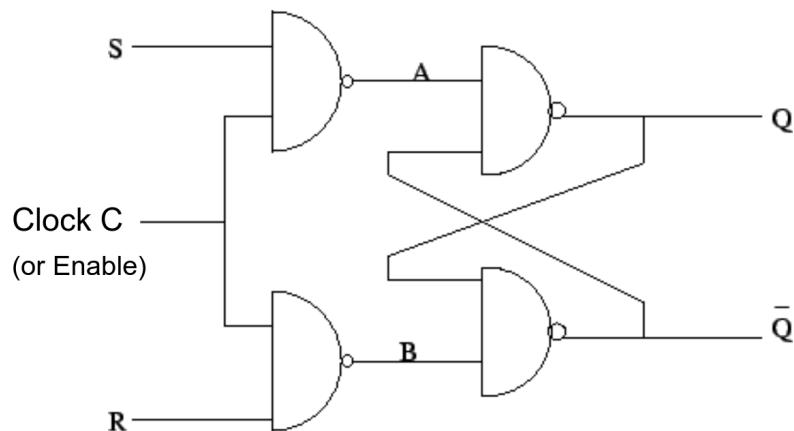
Input 1 (X)	Input 2 (Y)	Output ($\overline{X \oplus Y}$)
0	0	1
0	1	0
1	0	0
1	1	1

6.7.2 Circuit diagrams



6.8 D flip-flop

- The D flip-flop is a device that is able to store a single bit.
- It is usually used in CPU registers.

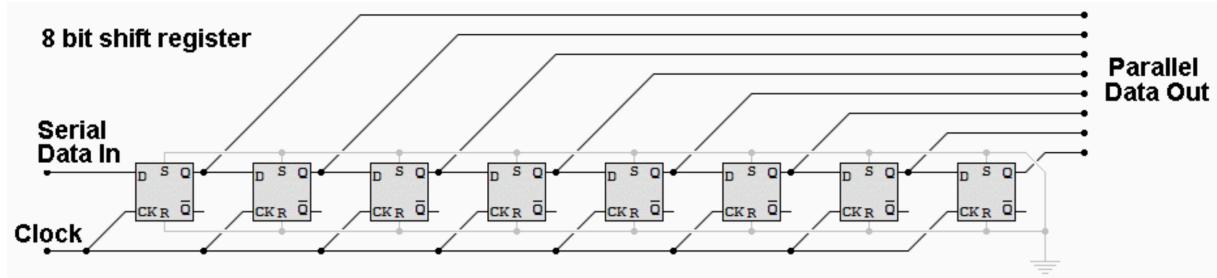


6.8.1 Truth table

S	R	CLK (or Enable)	$Q(t + 1)$	Comments
0	0	X	$Q(t)$	No change
0	1	↑	0	Reset
0	1	↑	1	Set
1	1	↑	?	Invalid

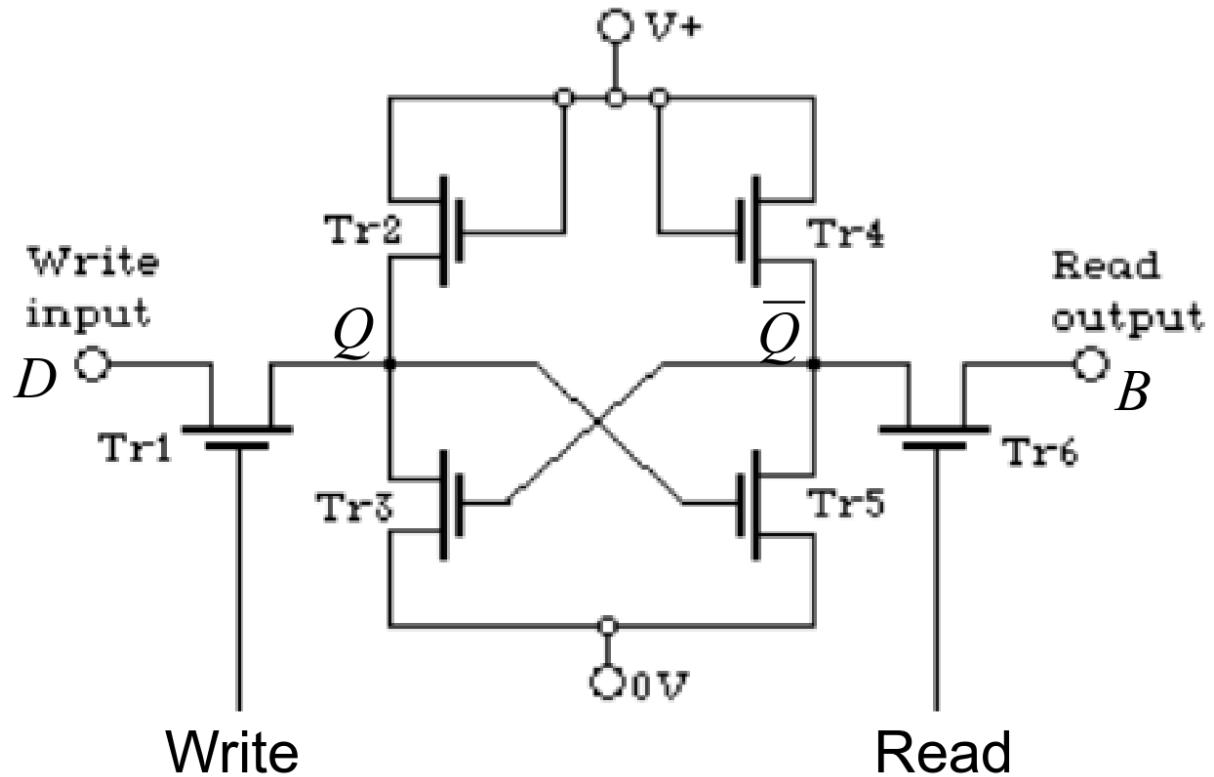
The ↑ shows that the device only responds on the rising edge.

6.9 8-bit shift register



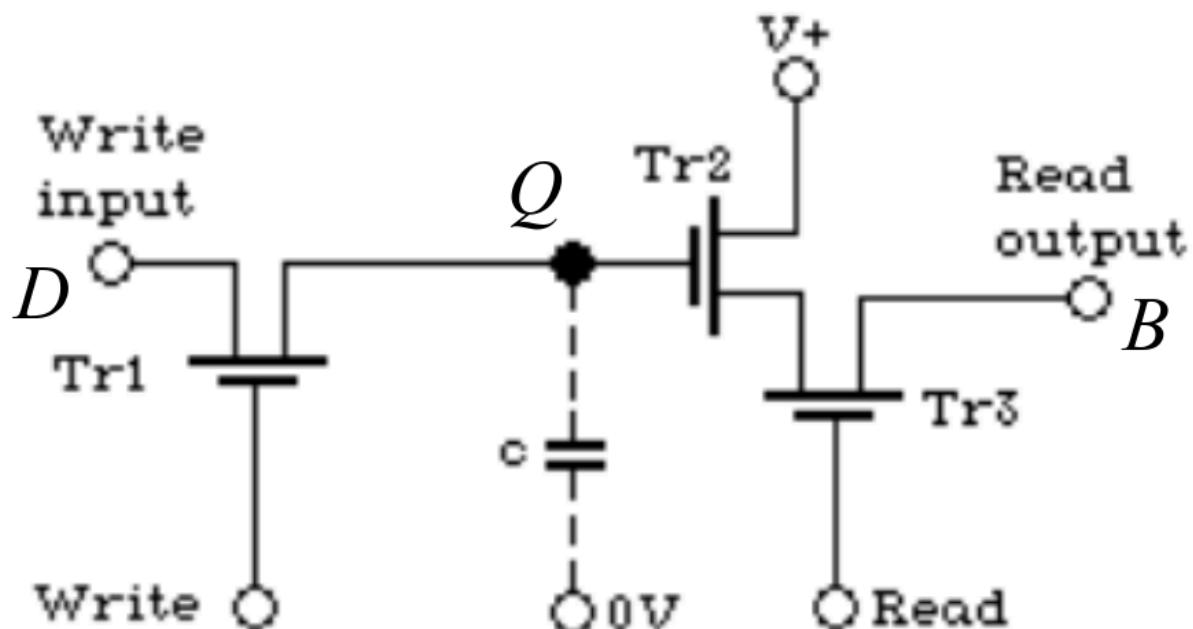
6.10 Single bit Static RAM (SRAM)

- Static RAM can continue to store memory even when the power is switched off, but it is more expensive than dynamic RAM.
- Write operation
 - “Write” line is in logic high.
 - Transistor Tr_1 is on.
 - $Q = D$
- Read operation
 - “Read” line is in logic high.
 - Transistor Tr_5
 - $B = 1 - Q$

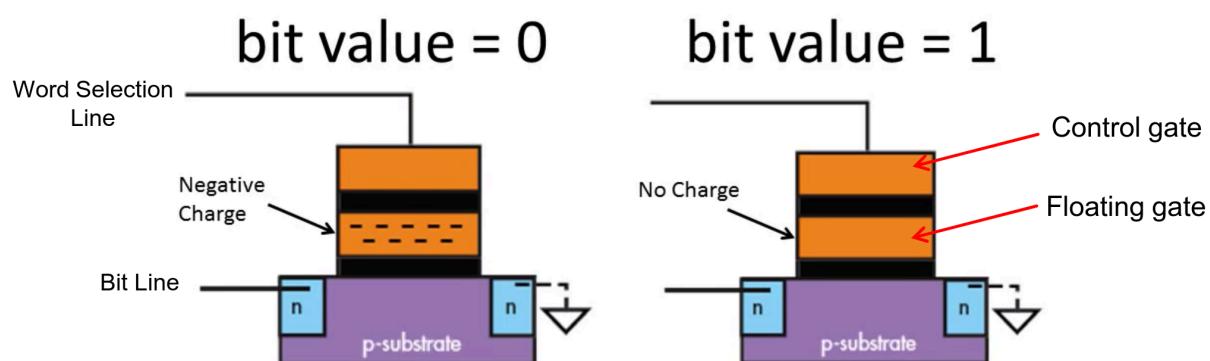


6.11 Single bit Dynamic RAM (DRAM)

- Dynamic RAM cannot store memory after it is powered off due to the capacitor losing its charge.
However, it is cheaper than static RAM due to having fewer transistors.
- Write operation
 - “Write” line is in logic high.
 - Transistor $Tr1$ is on.
 - $Q = D$
- Read operation
 - “Read” line is in logic high.
 - Transistor $Tr3$
 - $B = Q$



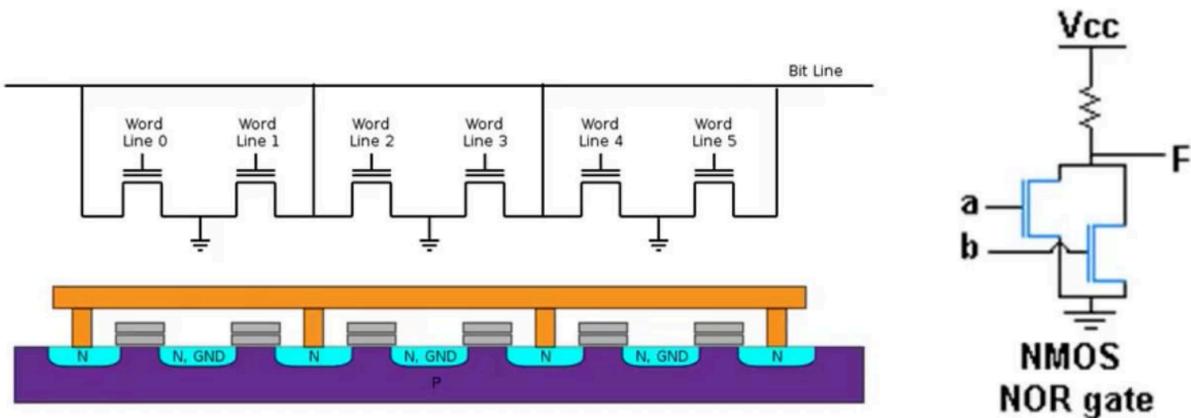
6.12 1-bit flash cell



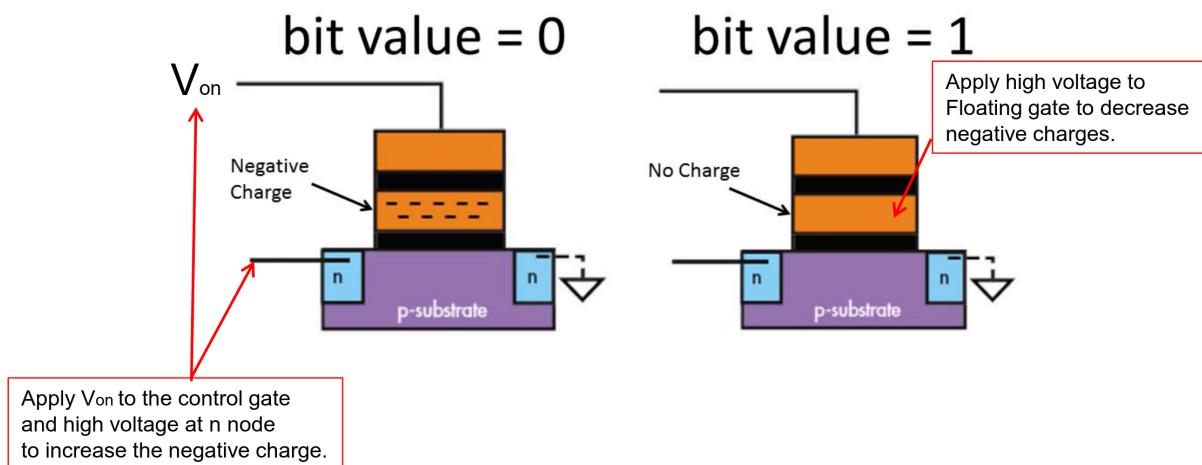
A 1-bit flash cell is made of a single level cell, which has:

- 2 charge states
- 1 bit per floating gate transistor

6.12.1 Line of flash cells



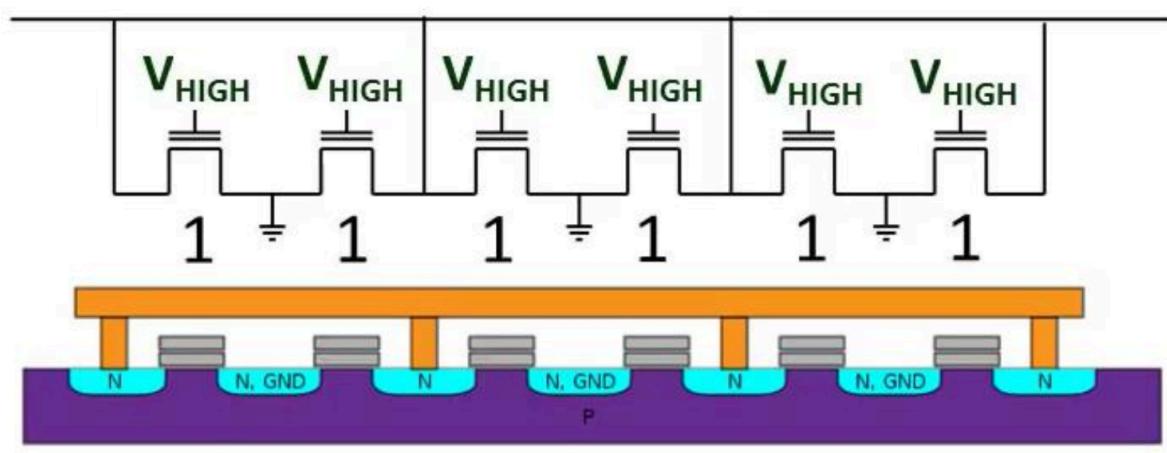
6.12.2 Writing



6.12.3 NOR erasing

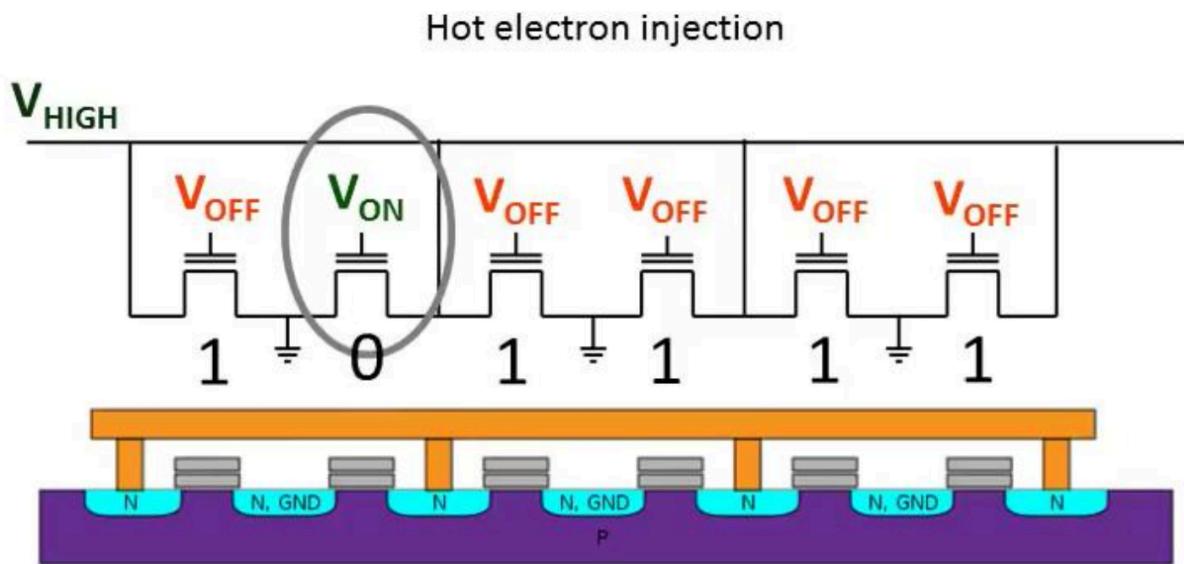
NOR erasing sets all bits to 1.

Block erasure via tunneling



6.12.4 NOR writing

NOR writing sets 0s.

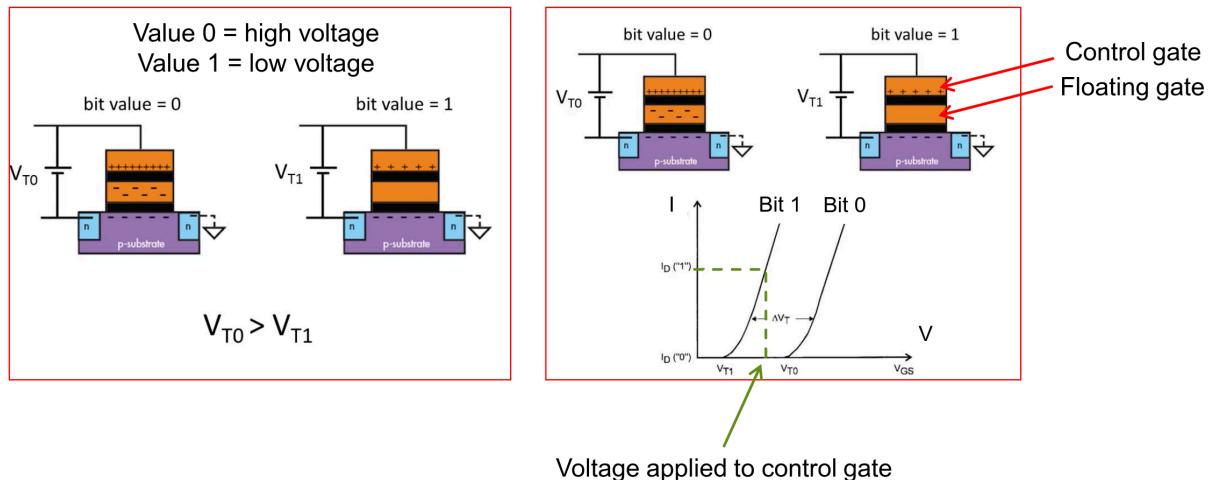


6.12.5 Reading

- To read from a flash cell, a voltage is applied to the control gate.
- The voltage is in between the voltage of the flash cell when the bit value is 0, and the voltage of the flash cell when the bit value is 1, i.e.

$$V_{T1} > V_{GS} > V_{T0}$$

- If there is no current, the output value is 0.
- If there is a current, the output value is 1.

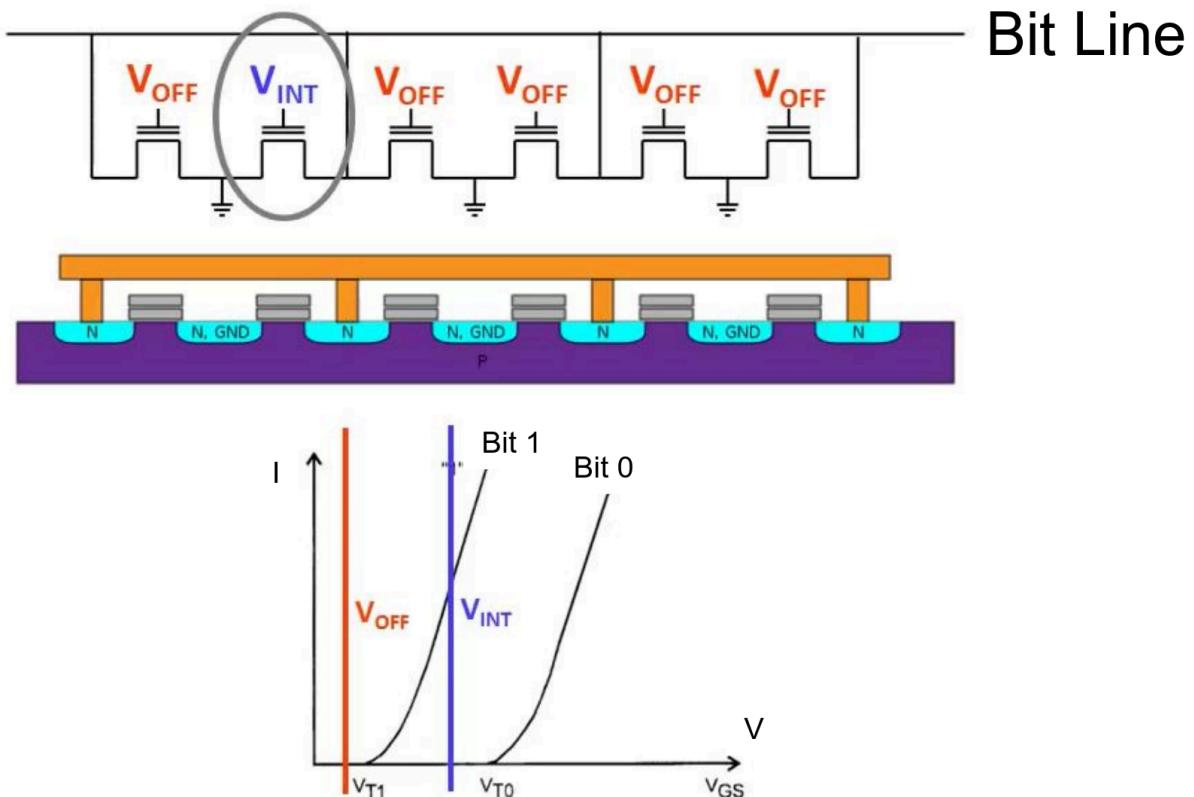


6.12.6 NOR reading

- NOR reading is used to read a single bit from a flash cell line.
- It also uses the same principles as the normal reading, which means the given voltage is in between the voltage of the flash cell when the bit value is 0, and the voltage of the flash cell when the bit value is 1, i.e.

$$V_{T1} > V_{GS} > V_{T0}$$

- Similarly, if there is no current, the output value is 0.
- If there is a current, the output value is 1.



7 Number representation

7.1 History of number systems

- Gottfried Wilhelm Leibniz (1646–1716) is the self-proclaimed inventor of the binary system and is considered as such by most historians of mathematics and mathematicians. However, systems related to binary numbers have appeared earlier in multiple cultures including Ancient Egypt, China, and India.
- Modern methods of writing decimals were invented less than 500 years ago. However, the use of decimals in various forms can be traced back thousands of years. The Babylonians used a number system based on 60 and extend it to deal with numbers less than 1. Some use of decimals was also made in ancient China, medieval Arabia and in Renaissance Europe.
- Swedish American engineer John Williams Nystrom developed the hexadecimal notation system in 1859.
- The eight digits of the octal system are 0, 1, 2, 3, 4, 5, 6, and 7. The octal system was first discovered in 1716 by Emanuel Swedenborg, although linguists speculate that it was discovered by the Proto-Indo-Europeans, a group of Neolithic people ancestors to the Indo-European people of the Bronze Age.

7.2 Binary number system

7.2.1 Format of binary numbers

- A binary number consists of a series of digits.
- A digit has its value and position in the series.
- One digit on the left is 2 times the digit on the right.
- For example, $1_{MSB}1100111000111_{LSB}$
- Hence, we can represent binary numbers in the following way:

$$V = D_{n-1}D_{n-2}\cdots D_1D_0$$

- Where:

$$D_i \in [0, 1], \quad i = 0, 1, 2, \dots \text{ or } i = -1, -2, \dots$$

$$V_n = D_{n-1}2^{n-1} + D_{n-2}2^{n-2} + \cdots + D_12^1D_02^0$$

7.2.2 Integer example

Binary numbers	Corresponding decimal values
00001111	$8 + 4 + 2 + 1 = 15$
11110000	$128 + 64 + 32 + 16 = 240$
11001100	$128 + 64 + 8 + 4 = 204$
00110011	$32 + 16 + 2 + 1 = 51$

7.2.3 Fractional example

Binary numbers	Corresponding decimal values
0.00001111	$0.0313 + 0.0156 + 0.0078 + 0.0039$
0.11110000	$0.5 + 0.25 + 0.125 + 0.0625$
0.11001100	$0.5 + 0.25 + 0.0313 + 0.0156$
0.00110011	$0.125 + 0.0625 + 0.0078 + 0.0039$

7.2.4 Binary addition

- Input:

$$V_1 = A_{n-1}2^{n-1} + A_{n-2}2^{n-2} + \cdots + A_12^1 + A_0 + 2^0$$

$$V_2 = B_{n-1}2^{n-1} + B_{n-2}2^{n-2} + \cdots + B_12^1 + B_0 + 2^0$$

- Output:

$$V_3 = C_{n-1}2^{n-1} + C_{n-2}2^{n-2} + \cdots + C_12^1 + C_0 + 2^0$$

- Rules:

- $0 + 0 = 0$
- $0 + 1 = 1$
- $1 + 0 = 1$
- $1 + 1 = 0$ with a carry of 1

$$\begin{array}{r}
 \text{1} \quad \text{1} \quad \text{1} \quad \text{0} \quad \text{0} \quad \text{0} \quad \text{Carry} \\
 1 \quad 0 \quad 1_2 \quad = \quad 5_{10} \quad \text{Augend} \\
 + \quad \quad \quad 1 \quad 1_2 \quad = \quad 3_{10} \quad \text{Addend} \\
 \hline
 1 \quad 10 \quad 10 \quad 10 \quad = \quad 8_{10}
 \end{array}$$

- Example:

- Augend:

11110000

- Addend:

11001100

- Result:

$$\begin{array}{r}
 \text{1} \quad \text{1} \quad \text{0} \quad \text{Carry} \\
 1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad \text{Augend} \\
 + \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad \text{Addend} \\
 \hline
 \textcolor{red}{1} \quad 1 \quad 0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0
 \end{array}$$

7.2.5 Binary subtraction

- Input:

$$V_1 = A_{n-1}2^{n-1} + A_{n-2}2^{n-2} + \dots + A_12^1 + A_0 + 2^0$$

$$V_2 = B_{n-1}2^{n-1} + B_{n-2}2^{n-2} + \dots + B_12^1 + B_0 + 2^0$$

- Output:

$$V_3 = C_{n-1}2^{n-1} + C_{n-2}2^{n-2} + \dots + C_12^1 + C_0 + 2^0$$

- Rules:

- $0 - 0 = 0$
- $0 - 1 = 1$ with a borrow of 1
- $1 - 0 = 1$
- $1 - 1 = 0$

$$\begin{array}{r}
 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad \text{Borrow} \\
 1 \quad 10 \quad 1_2 = 5_{10} \quad \text{Minuend} \\
 - \quad 1 \quad 1_2 = 3_{10} \quad \text{Subtrahend} \\
 \hline
 0 \quad 1 \quad 0_2 = 2_{10}
 \end{array}$$

- Example:

- Minuend:

$$11110000$$

- Subtrahend:

$$11001100$$

- Result:

$$\begin{array}{r}
 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad \text{Borrow} \\
 1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad \text{Minuend} \\
 - \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad \text{Subtrahend} \\
 \hline
 \color{red}{0} \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0
 \end{array}$$

7.3 Decimal number system

- A decimal number consists of a series of digits.
- A digit has its value and its position in the series.
- One digit on the left is 10 times the digit on the right.
- For example, 1234000.
- Hence, we can represent decimal numbers in the following way:

$$V = D_{n-1}D_{n-2}\cdots D_1D_0$$

- Where:

$$D_i \in [0, 1, 2, 3, 4, 5, 6, 7, 8, 9], \quad i = 0, 1, 2, \dots \text{ or } i = -1, -2, \dots$$

$$V_n = D_{n-1}10^{n-1} + D_{n-2}10^{n-2} + \cdots + D_110^1D_010^0$$

7.3.1 Integer example

Decimal numbers	Corresponding decimal values
00002234	$2000 + 200 + 30 + 4$
67890000	$60000000 + 7000000 + 800000 + 90000$
34008900	$30000000 + 400000 + 8000 + 900$
00890034	$800000 + 90000 + 30 + 4$

7.3.2 Fractional example

Decimal numbers	Corresponding decimal values
0.00002234	$0.00002 + 0.000002 + 0.000003 + 0.00000004$
0.67890000	$0.6 + 0.07 + 0.008 + 0.0009$
0.34008900	$0.3 + 0.04 + 0.00008 + 0.000009$
0.00890034	$0.008 + 0.009 + 0.0000003 + 0.00000004$

7.3.3 Decimal addition

- Input:

$$V_1 = A_{n-1}10^{n-1} + A_{n-2}10^{n-2} + \dots + A_110^1 + A_0 + 10^0$$

$$V_2 = B_{n-1}10^{n-1} + B_{n-2}10^{n-2} + \dots + B_110^1 + B_0 + 10^0$$

- Output:

$$V_3 = C_{n-1}10^{n-1} + C_{n-2}10^{n-2} + \dots + C_110^1 + C_0 + 10^0$$

- Rules:

- If $A_i + B_i > 9$, $C_i = A_i + B_i - 10$, with a carry of 1.
- Otherwise, $C_i = A_i + B_i$.

$$\begin{array}{r}
 \textcolor{blue}{1} \quad \textcolor{blue}{1} \quad \textcolor{blue}{0} \quad \text{Carry} \\
 9 \quad 5 \quad \text{Augend} \\
 + \quad \quad 1 \quad 6 \quad \text{Addend} \\
 \hline
 1 \quad 1 \quad 1
 \end{array}$$

- Example:

- Augend:

67890000

- Addend:

34008900

- Result:

$$\begin{array}{r}
 1 \quad 1 \quad 0 \quad \text{Carry} \\
 6 \quad 7 \quad 8 \quad 9 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad \text{Augend} \\
 + \quad 3 \quad 4 \quad 0 \quad 0 \quad 8 \quad 9 \quad 0 \quad 0 \quad 0 \quad \text{Addend} \\
 \hline
 \textcolor{red}{1} \quad 0 \quad 1 \quad 8 \quad 9 \quad 8 \quad 9 \quad 0 \quad 0
 \end{array}$$

7.3.4 Decimal subtraction

- Input:

$$V_1 = A_{n-1}10^{n-1} + A_{n-2}10^{n-2} + \dots + A_110^1 + A_0 + 10^0$$

$$V_2 = B_{n-1}10^{n-1} + B_{n-2}10^{n-2} + \dots + B_110^1 + B_0 + 10^0$$

- Output:

$$V_3 = C_{n-1}10^{n-1} + C_{n-2}10^{n-2} + \dots + C_110^1 + C_0 + 10^0$$

- Rules:

- If $A_i < B_i$, $C_i = 10 + A_i - B_i - 10$, with a borrow of 1.
- Otherwise, $C_i = A_i - B_i$.

$$\begin{array}{r}
 \textcolor{blue}{1} \quad \textcolor{blue}{0} \quad \text{Borrow} \\
 9 \quad \overset{1}{5}_{10} \quad \text{Minuend} \\
 - \quad 1 \quad 6_{10} \quad \text{Subtrahend} \\
 \hline
 7 \quad 9_{10}
 \end{array}$$

- Example:

- Minuend:

67890000

- Subtrahend:

34008900

- Result:

$$\begin{array}{r}
 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad \text{Borrow} \\
 6 \quad 7 \quad 8 \quad 9 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad \text{Minuend} \\
 - \quad 3 \quad 4 \quad 0 \quad 0 \quad 8 \quad 9 \quad 0 \quad 0 \quad 0 \quad \text{Subtrahend} \\
 \hline
 0 \quad 0 \quad 3 \quad 3 \quad 8 \quad 8 \quad 1 \quad 1 \quad 0 \quad 0
 \end{array}$$

7.3.5 Hexadecimal number system

- A hexadecimal number consists of a series of digits.
- A digit has its value and its position in the series.
- One digit on the left is 16 times the digit on the right.
- For example, ABCD5678.
- Hence, we can represent hexadecimal numbers in the following way:

$$V = D_{n-1}D_{n-2}\cdots D_1D_0$$

- Where:

$$D_i \in [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F], \quad i = 0, 1, 2, \dots \text{ or } i = -1, -2, \dots$$

$$V = D_{n-1}16^{n-1} + D_{n-2}16^{n-2} + \cdots + D_116^1 + D_016^0$$

7.3.6 Integer example

Hexadecimal numbers	Corresponding decimal values
00005688	$20480 + 1536 + 128 + 8$
ABCD0000	$2684400000 + 184549376 + 12582912 + 851968$
CD007800	$3221200000 + 218103808 + 28672 + 2048$
007800CD	$7340032 + 524288 + 192 + 13$

7.3.7 Fractional example

Hexadecimal numbers	Corresponding decimal values
0.00005688	$0.0000047684 + 0.00000035763 + 0.000000029802 + 0.0000000018626$
0.ABCD0000	$0.625 + 0.043 + 0.0029 + 0.00019836$
0.CD007800	$0.75 + 0.0508 + 0.0000066757 + 0.00000047684$
0.007800CD	$0.0017 + 0.00012207 + 0.000000044703 + 0.0000000030268$

7.3.8 Hexadecimal addition

- Input:

$$V_1 = A_{n-1}16^{n-1} + A_{n-2}16^{n-2} + \dots + A_116^1 + A_0 + 16^0$$

$$V_2 = B_{n-1}16^{n-1} + B_{n-2}16^{n-2} + \dots + B_116^1 + B_0 + 16^0$$

- Output:

$$V_3 = C_{n-1}16^{n-1} + C_{n-2}16^{n-2} + \dots + C_116^1 + C_0 + 16^0$$

- Rules:

- If $A_i + B_i > F$, $C_i = A_i + B_i - 16$, with a carry of 1
- Otherwise, $C_i = A_i + B_i$

1	0	1	1	0	Carry	
F	0	B	A	Augend		
+	E	9	A	D	Addend	
						Sum
1	D	A	6	7		

- Example:

- Augend:

ABCD0000

- Addend:

007800CD

- Result:

0	0	1	1	0	0	0	0	0	Carry	
A	B	C	D	0	0	0	0	0	Augend	
+	0	0	7	8	0	0	C	D	Addend	
										Sum
0	A	C	4	5	0	0	C	D		

7.3.9 Hexadecimal subtraction

- Input:

$$V_1 = A_{n-1}16^{n-1} + A_{n-2}16^{n-2} + \dots + A_116^1 + A_0 + 16^0$$

$$V_2 = B_{n-1}16^{n-1} + B_{n-2}16^{n-2} + \dots + B_116^1 + B_0 + 16^0$$

- Output:

$$V_3 = C_{n-1}16^{n-1} + C_{n-2}16^{n-2} + \dots + C_116^1 + C_0 + 16^0$$

- Rules:

- If $A_i < B_i$, $C_i = 16 + A_i - B_i$, with a borrow of 1
- Otherwise, $C_i = A_i - B_i$

0	0	1	1	1	Borrow
D	E	A	9	Minuend	
-	4	F	B	D	Subtrahend
					<hr/>
0	8	E	E	C	

- Example:

- Minuend:

67890000

- Subtrahend:

34008900

- Result:

0	0	0	0	1	1	0	0	0	Borrow
6	7	8	9	0	0	0	0	0	Minuend
-	3	4	0	0	8	9	0	0	Subtrahend
									<hr/>
0	3	3	8	8	7	7	0	0	

7.3.10 Hexadecimal ASCII table

For example, 0x42 refers to the character “B”.

LEAST SIGNIFICANT BITS	Hexadecimal Codes		MOST SIGNIFICANT BITS					
	0	1	2	3	4	5	6	7
0	Null	Data Link Escape	Space	0	@	P	‘	p
1	Start of Heading	Device Control 1	!	1	A	Q	a	q
2	Start of Text	Device Control 2	“	2	B	R	b	r
3	End of Text	Device Control 3	#	3	C	S	c	s
4	End of Transmit	Device Control 4	\$	4	D	T	d	t
5	Enquiry	Neg Acknowledge	%	5	E	U	e	u
6	Acknowledge	Synchronous Idle	&	6	F	V	f	v
7	Bell	End of Trans Block	,	7	G	W	g	w
8	Backspace	Cancel	(8	H	X	h	x
9	Horizontal Tab	End of Medium)	9	I	Y	i	y
A	Line Feed	Substitute	*	:	J	Z	j	z
B	Vertical Tab	Escape	+	;	K	[k	{
C	Form Feed	File Separator	,	<	L	\	l	
D	Carriage Return	Group Separator	-	=	M]	m	}
E	Shift Out	Record Separator	.	>	N	^	n	~
F	Shift In	Unit Separator	/	?	O	_	o	Delete

7.4 Conversion look-up table

Hex	Binary	Decimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
A	1001	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15

7.4.1 Conversion from binary to hexadecimal

- Rules:
 - 1 digit of hexadecimal corresponds to 4 digits of binary.
 - Group every 4 digits of binary together and replace it with the corresponding digit of hexadecimal.
- Example:
 - Binary input:

$\overbrace{1111}^F \overbrace{1110}^E \overbrace{1100}^C \overbrace{1000}^8$

- Hexadecimal output:
 - $F = 1111$
 - $E = 1110$
 - $C = 1100$
 - $8 = 1000$
 - Hexadecimal = 0xFEC8

7.4.2 Conversion from hexadecimal to binary

- Rules:
 - 1 digit of hexadecimal corresponds to 4 digits of binary.
 - Expand every digit of hexadecimal into a series of 4 digits of binary and replace it with corresponding 4 digits of binary.
- Example:
 - Hexadecimal input:

0x \overbrace{F}^{1111} \overbrace{E}^{1110} $\overbrace{8}^{1000}$ $\overbrace{9}^{1001}$ \overbrace{A}^{1010} \overbrace{B}^{1011} $\overbrace{5}^{0101}$ $\overbrace{6}^{0110}$

- Binary output:
 - $F = 1111$ $E = 1110$ $8 = 1000$ $9 = 1001$
 - $A = 1010$ $B = 1011$ $5 = 0101$ $6 = 0110$
 - Binary = 1111 1110 1000 1001 1010 1011 0101 0110

7.4.3 Conversion from decimal to binary, integer part

- Decimal input:

$$V_{\text{decimal}} = D_{n-1}10^{n-1} + D_{n-2}10^{n-2} + \dots + D_110^1 + D_010^0$$

- Binary output:

$$V_{\text{binary}} = B_{n-1}2^{n-1} + B_{n-2}2^{n-2} + \dots + B_12^1 + B_02^0$$

- Process:

- Divide the input by 2. The remainder is the first digit of the binary output.
- Divide the quotient by 2. The remainder is the second digit of the binary output.
- Continue the process until the **quotient becomes zero**.

- Example:

- Input: 57 in decimal
 - 1. $\frac{57}{2} = 28$, Remainder = 1 (Binary number will end with 1)
 - 2. $\frac{28}{2} = 14$, Remainder = 0
 - 3. $\frac{14}{2} = 7$, Remainder = 0
 - 4. $\frac{7}{2} = 3$, Remainder = 1
 - 5. $\frac{3}{2} = 1$, Remainder = 1
 - 6. $\frac{1}{2} = 0$, Remainder = 1 (Binary number will start with 1)
- Output: 111001 in binary

7.4.4 Conversion from decimal to binary, fractional part

- Process:

- Multiply the input by 2. The decimal number before the decimal point is the first digit of the binary output after the decimal point.
- Multiply the fractional part from the previous result by 2. The decimal number before the decimal point is the second digit of the binary output after the decimal point.
- Continue the process until the quotient becomes **zero** or a **periodic number**, which means the number in binary keeps repeating.
- Read the number starting from the **top**, the first result is the first digit after the decimal point.

- Example:

- Input: 0.375 in decimal
 - 1. $0.375 \times 2 = 0.75$, Fractional part = 0.75 with carry = 0
 - 2. $0.75 \times 2 = 1.50$, Fractional part = 0.50 with carry = 1
 - 3. $0.50 \times 2 = 1.00$, Fractional part = 0.00 with carry = 1
- Output: 0.011 in binary

- Faster method:

- Multiply the fractional part by 2 to the power of the number of decimal places required, i.e.

$$\text{Fractional part} \times 2^n$$

Where n is the number of decimal places required.

- Round the result to the nearest whole number and convert the result to binary.

- Example:

- Input: 0.375 in decimal, which has 3 decimal places, so $n = 3$:
 - 1. $0.375 \times 2^3 = 3$
 - 2. $3_{10} = 011_2$
- Output: 0.011 in binary

7.4.5 Example of converting a decimal number with decimals to binary

Input: 43.167 in decimal.

- For the integer part:
 1. $\frac{43}{2} = 21$, Remainder = 1
 2. $\frac{21}{2} = 10$, Remainder = 1
 3. $\frac{10}{2} = 5$, Remainder = 0
 4. $\frac{5}{2} = 2$, Remainder = 1
 5. $\frac{2}{2} = 1$, Remainder = 0
 6. $\frac{1}{2} = 0$, Remainder = 1
- For the fractional part:
 1. $0.167 \times 2 = 0.334$
 2. $0.334 \times 2 = 0.668$
 3. $0.668 \times 2 = 1.336$
 4. $0.336 \times 2 = 0.672$
 5. $0.672 \times 2 = 1.344$

Output: 101011.00101 in binary.

7.4.6 Conversion from decimal to hexadecimal, integer part

- Decimal input:

$$V_{\text{decimal}} = D_{n-1}10^{n-1} + D_{n-2}10^{n-2} + \dots + D_110^1 + D_010^0$$

- Hexadecimal output:

$$V_{\text{hexadecimal}} = H_{n-1}16^{n-1} + H_{n-2}16^{n-2} + \dots + H_116^1 + H_016^0$$

- Process:
 - Divide the input by 16. The remainder is the first digit of the hex output.
 - Divide the quotient by 16. The remainder is the second digit of the hex output.
 - Continue the process until the quotient becomes zero.
- Example:
 - Input: 35243 in decimal
 1. $\frac{35243}{16} = 2202$, Remainder = 11 → B (hex number will end with B)
 2. $\frac{2202}{16} = 137$, Remainder = 10 → A
 3. $\frac{137}{16} = 8$, Remainder = 9
 4. $\frac{8}{16} = 0$, Remainder = 8 (hex number will start with 8)

Therefore, collecting the remainders, $35243_{10} = 89AB_{16}$. To check:

$$(8 \times 16^3) + (9 \times 16^2) + (10 \times 16^1) + (11 \times 16^0) = 35243_{10}$$

- Output: 89AB in hexadecimal

7.4.7 Conversion from decimal to hexadecimal, fractional part

- Process:
 - Multiply the input by 16. The decimal number before the decimal point is the first digit of the binary output after the decimal point.
 - Multiply the fractional part from the previous result by 16. The decimal number before the decimal point is the second digit of the binary output after the decimal point.
 - Continue the process until the quotient becomes zero or a periodic number
 - Read the number starting from the top, the first result is the first digit after the decimal point.
- Example 1:
 - Input: 0.375 in decimal
 - 1. $0.375 \times 16 = 6.0$
 - Output: 0.6 in hexadecimal
- Example 2:
 - Input: 0.987 in decimal
 - 1. $0.987 \times 16 = 15.792 \rightarrow F$
 - 2. $0.792 \times 16 = 12.672 \rightarrow C$
 - 3. $0.672 \times 16 = 10.752 \rightarrow A$
 - 4. $0.752 \times 16 = 12.032 \rightarrow C$
 - Output: 0.FCAC in hexadecimal

7.5 Representation of integers

7.5.1 Binary-coded decimal of integers

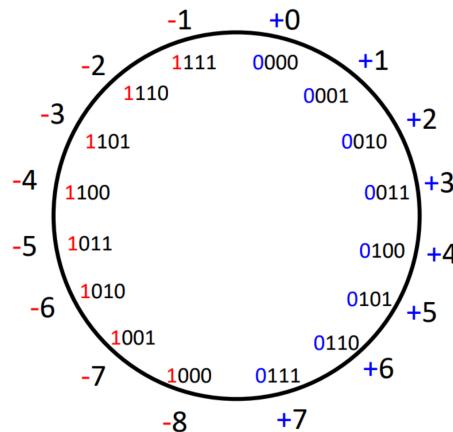
- Rules:
 - Each digit of decimal is represented by four digits of binary.
 - 0 = 0000, 1 = 0001, 2 = 0010, 3 = 0011, 4 = 0100
 - 5 = 0101, 6 = 0110, 7 = 0111, 8 = 1000, 9 = 1001
- Examples:
 - Decimal(9879) = Binary(1001 1000 0111 1001)
 - Decimal(12345) = Binary(0001 0010 0011 0100 0101)

7.5.2 Signed magnitude format of integers

- Rules:
 - Allocate the most-significant bit (MSB) to represent the signs.
 - Allocate the remaining bits to represent the values.
- Examples:
 - Binary with 4 bits:
 - Max = 0 111 → 7
 - Min = 1 111 → -7
 - Binary with 8 bits:
 - Max = 0 111 1111 → 127
 - Min = 1 111 1111 → -127
 - Binary with 16 bits:
 - Max = 0 111 1111 1111 1111 → 32767
 - Min = 1 111 1111 1111 1111 → -32767
 - Binary with 32 bits:
 - Max = 0 111 1111 1111 1111 1111 1111 1111 → 2,147,483,647
 - Min = 1 111 1111 1111 1111 1111 1111 1111 → -2,147,483,647

7.5.3 Two's complement format of integers

- Rules:
 - For a binary number of N bits, the positive numbers are represented by the bits from 0 (LSB) to $N - 2$ and bit $N - 1$ (MSB) is zero.
 - For a binary number of N bits, the negative numbers are represented by the two's complements of their positive numbers.
 - A two's complement is equal to bit-wise complement plus 1.
- Examples:
 - 4 bit binary of integer 6 = 0110
 - 4 bit binary of integer $-6 = 1001 + 1 = 1010$
 - 8 bit binary of integer 102 = 0110 0110
 - 8 bit binary of integer $-102 = 1001\ 1001 + 1 = 1001\ 1010$
- Diagram:



- 8 positive numbers
- 8 negative numbers
- $+8 = 1000$
- $-8 = 0111 + 1 = 1000$
- $+8 = -8$
- So, we ignore +8

7.5.4 Reasons to use two's complement format

Addition:

- Addition not dependent on the signs of operand.
- No need to compare magnitudes to determine sign of result.

Subtraction:

- Subtraction is treated as an addition.
- Add the negative of the subtrahend to the minuend.

Simple implementation:

- Adder unit.
- Negation circuit unit.

The simpler addition and subtraction scheme makes two's complement the most common choice for integer number systems within digital systems.

7.6 Representation of floating point numbers

7.6.1 Binary-coded decimal of floating-point numbers

- Rules:
 - Each digit of decimal is represented by 4 digits of binary.
 - 0 = 0000, 1 = 0001, 2 = 0010, 3 = 0011, 4 = 0100
 - 5 = 0101, 6 = 0110, 7 = 0111, 8 = 1000, 9 = 1001
- Examples:
 - Decimal(9879.34) = Binary(1001 1000 0111 1001).Binary(0011 0100)
 - Decimal(12345.5) = Binary(0001 0010 0011 0100 0101).Binary(0101)

7.6.2 Exponent-mantissa format of floating point numbers

Using the 32-bit ANSI/IEEE 754 standard format.

Sign bit (S)	Exponent (E)	Mantissa (M) or Fraction
(1 bit)	(8 bits)	(23 bits)

- Sign bit (S)
 - 0 denotes positive number, 1 denotes negative number.
- Exponent (E)
 - Represents both positive and negative exponents.
 - A bias value of 127_{10} must be added to all exponents, regardless of sign.
- Mantissa or Fraction (M)
 - Represents the leading significant bits in the number.
 - It is stored in the normalised form.

7.6.3 Conversion of decimals to exponent-mantissa format

- Procedure:
 - Convert decimals to binary format.
 - Normalise binary format.
 - Determine exponent.
 - Determine mantissa.
- Example of 2.75 in decimal:
 1. $2.75_{10} = 10.11_2$
 2. In normalised form: $1.011_2 \times 2^1$
 3. Express in 23-bit mantissa without leading 1_2 :

$$M = 011\ 0000\ 0000\ 0000\ 0000\ 0000$$

4. Express 8-bit exponent (E) with added bias:

$$E = 1_{10} + 127_{10} = 128_{10} = 1000\ 0000_2$$

5. Express sign bit (S):

$$S = 0 \quad (\text{positive number})$$

- Example of -0.0625 in decimal:
 1. $-0.0625_{10} = -0.0001_2$
 2. In normalised form: $1.0_2 \times 2^{-4}$
 3. Express in 23-bit mantissa without leading 1_2 :

$$M = 000\ 0000\ 0000\ 0000\ 0000\ 0000$$

4. Express 8-bit exponent (E) with added bias:

$$E = -4_{10} + 127_{10} = 123_{10} = 0111\ 1011_2$$

5. Express sign bit (S):

$$S = 1 \quad (\text{negative number})$$

- Example of -417680 in decimal:
 1. $-417680_{10} = -110\ 0101\ 1111\ 1001\ 0000_2$
 2. In normalised form: $1.10010111110010000_2 \times 2^{18}$
 3. Express in 23-bit mantissa without leading 1_2 :

$$M = 100\ 1011\ 1111\ 0010\ 0000\ 0000$$

4. Express 8-bit exponent (E) with added bias:

$$E = 18_{10} + 127_{10} = 145_{10} = 1010\ 0001_2$$

5. Express sign bit (S):

$$S = 1 \quad (\text{negative number})$$

7.6.4 Zero and infinity in exponent-mantissa format

- Representing $+/-0.0$ in 32-bit ANSI/IEEE 754 standard format

Sign bit (S)	Exponent (E)	Mantissa (M) or Fraction
0/1	0000 0000	000 0000 0000 0000 0000

- Representing $+/-\infty$ in 32-bit ANSI/IEEE 754 standard format

Sign bit (S)	Exponent (E)	Mantissa (M) or Fraction
0/1	1111 1111	000 0000 0000 0000 0000

8 Nature of programming

8.1 Programming versus writing

Programming (using Arm Keil):

- You have a solution in mind.
- You organise your solution.
- You compose your program.
- You test your program.
- You deploy your program.

Writing (using LaTeX or Microsoft Word):

- You have a story in mind.
- You organise your story.
- You compose your texts.
- You proof-read your texts.
- You publish your texts.

8.2 How to organise your solution?

1. Plan arithmetic or logical computations underlying your solution.
2. Allocate memory resources which are indispensable for the implementation of the planned computations.

8.2.1 Example of $(x = (a + b) - c)$

Allocation of memory for the program:

```
EXPORT Start
AREA progA, CODE, READONLY

var_a DCD 5 ; a = 5
var_b DCD 6 ; b = 6
var_c DCD 7 ; c = 7
var_x DCD 0 ; Initialise x to 0
```

Planning of actual computation:

```
Start ADR R4, var_a ; Get the memory address of var_a
      LDR R0, [R4] ; Get the value of var_a
      ADR R4, var_b ; Get the memory address of var_b, reusing R4
      LDR R1, [R4] ; Get the value of var_b
      ADD R3, R0, R1 ; Compute var_a + var_b
      ADR R4, var_c ; Get the memory address of var_c
      LDR R2, [R4] ; Get the value of var_c
      SUB R3, R3, R2 ; Complete the computation of x = (a + b) - c
      ADR R4, var_x ; Get the memory address of var_x
      STR R3, [R4] ; Store the value of x
stop  B stop
      END
```

8.3 Procedure of planning computations

1. Describe your solution in the form of mathematics or logic.
2. Transform your solution into algorithms which consist of a series of arithmetic and logical computations.
3. Draw flowcharts which connect arithmetic and logical computations in terms of their inputs and outputs.
4. Describe flow charts with the use of a programming language.

8.3.1 Example of transforming a solution into an algorithm

Solution:

$$ax^2 + bx + c = 0$$

$$x = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

$$x = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

Algorithm:

1. First do:

$$y = b^2 - 4ac$$

2. Then do:

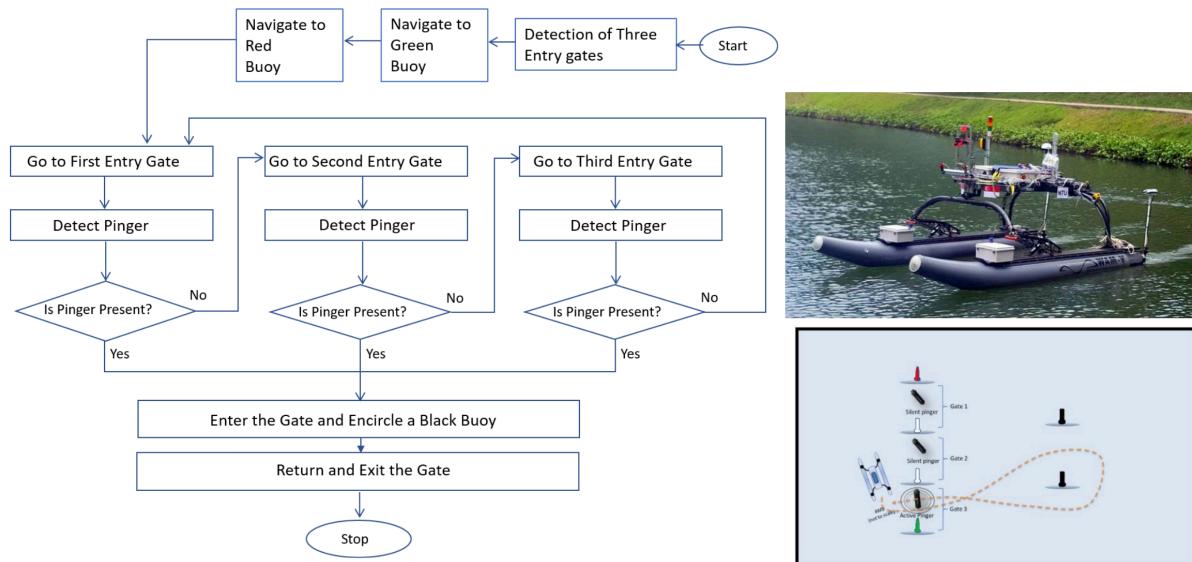
$$z = \sqrt{y}$$

3. Then do:

$$x_1 = \frac{-b + z}{2a}$$

$$x_2 = \frac{-b - z}{2a}$$

8.3.2 Example of flowchart corresponding to algorithm



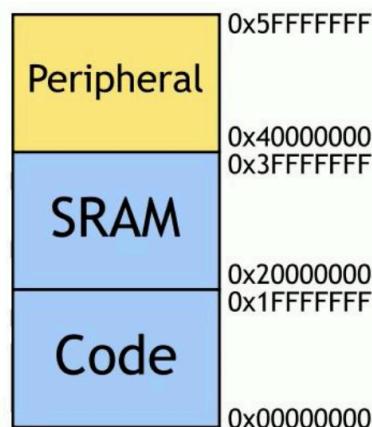
9 Allocation of memory

9.1 Why allocate memory?

- Hardware can support multiple applications running concurrently.
- The ALU is being shared among these applications, which means the application has full access to the ALU during the time when the ALU is allocated to it.
- However, memory units are generally not shared.
- Hence, each application must take care of memory allocation explicitly.

9.2 Memory allocation scenarios

1. Allocation of memory for housing a program itself.
2. Allocation of memory for housing constants, parameters and variables of a program.



- The allocation of memory for the program itself is stored in the code section of ARM memory.
- The allocation of memory for the variables and constants used by a program is stored in the static RAM or SRAM.
- On-chip code, data, and I/O are located in the first 1.5 GiB of memory space.
- Each is allocated 0.5 GiB.
- May use physically separate buses for each space.

9.3 Memory allocation example

```
EXPORT      Start
AREA       progB, CODE, READONLY
var_a     DCD 1      ; a = 1
var_b     DCD 15     ; b = 15
var_z     DCD 0      ; Initialise z to 0

Start          ; z = (a << 2) | (b & 15)

ADR R4, var_a    ; Get the memory address of var_a
LDR R0, [R4]      ; Get the value of var_a
MOV R0, R0, LSL #2 ; Bitwise shift var_a left by 2: R0 = a << 2
ADR R4, var_b    ; Get the memory address of var_b
LDR R1, [R4]      ; Get the value of var_b

; Perform bitwise AND on var_b and 152: r_1 = (b & 152)
AND R1, R1, #15

; Perform bitwise OR on R0 and R1: R1 = (a << 2) | (b & 152)
ORR R1, R0, R1
ADR R4, var_z    ; Get the memory address of var_z

; Store the value of R1 in the memory address of var_z
STR R1, var_z

stop    B  stop
END
```

10 ARM programming tools

10.1 Overview

- The compiler converts from C files (.c) to assembler files (.s).
- The assembler converts from assembler files (.s) to object files (.o).
- The linker converts from object files (.o) to executable files (.exe).
- The linker can link object files (.o) with archive files (.a)

10.2 Assembler directives for memory destination (GCC and ArmClang)

Mnemonic and syntax	Description
.bss <i>symbol</i> , <i>size in bytes</i> [, <i>alignment</i> [, <i>bank offset</i>]]	Reserves <i>size</i> in bytes in the .bss (uninitialised data) section.
.data	Assembles into the .data (initialised data) section.
.sect “ <i>section name</i> ”	Assmebles into a named (initialised) section
.text	Assembles into a .text (executable code) section.
<i>symbol</i> .usect “ <i>section name</i> ”, <i>size in bytes</i> [, <i>alignment</i> [, <i>bank offset</i>]]	Reserves <i>size</i> in bytes in a named (uninitialised) section.
.space	Reserves a block of bytes.
.cstring { <i>expr₁</i> “ <i>string₁</i> ”}[, ..., { <i>expr_n</i> “ <i>string_n</i> ”}]	Initialises one or more text strings.
.double <i>value₁</i> [, ..., <i>value_n</i>]	Initialises one or more 64-bit, IEEE double-precision floating-point numbers.
.field <i>value</i> [, <i>size</i>]	Initialises a field of <i>size</i> bits (1-32) with <i>value</i> .
.float <i>value₁</i> [, ..., <i>value_n</i>]	Initialises one or more 32-bit, IEEE, single-precision floating-point numbers.
.half <i>value₁</i> [, ..., <i>value_n</i>]	Initialises one or more 16-bit integers (half-word).
.int <i>value₁</i> [, ..., <i>value_n</i>]	Initialises one or more 32-bit integers.
.long <i>value₁</i> [, ..., <i>value_n</i>]	Initialises one or more 32-bit integers.
.short <i>value₁</i> [, ..., <i>value_n</i>]	Initialises one or more 16-bit integers (half-word).
.string { <i>expr₁</i> “ <i>string₁</i> ”}[, ..., { <i>expr_n</i> “ <i>string_n</i> ”}]	Initialises one or more text strings.
.ubyte <i>value₁</i> [, ..., <i>value_n</i>]	Initialises one or more successive unsigned bytes in the current section.
.uchar <i>value₁</i> [, ..., <i>value_n</i>]	Initialises one or more successive unsigned bytes in the current section, but it is mainly use for ASCII characters.
.uhalf <i>value₁</i> [, ..., <i>value_n</i>]	Initialises one or more unsigned 16-bit integers (half-word).
.uint <i>value₁</i> [, ..., <i>value_n</i>]	Initialises one or more unsigned 32-bit integers.
.ulong <i>value₁</i> [, ..., <i>value_n</i>]	Initialises one or more unsigned 32-bit integers.
.ushort <i>value₁</i> [, ..., <i>value_n</i>]	Initialises one or more unsigned 16-bit integers (half-word).
.uword <i>value₁</i> [, ..., <i>value_n</i>]	Initialises one or more unsigned 32-bit integers.
.word <i>value₁</i> [, ..., <i>value_n</i>]	Initialises one or more 32-bit integers.

The SRAM can only hold data, while the code section can hold both instructions and data.

10.2.1 Example 1

In this example, code is assembled into the .data and .text sections.

```
1          ****
2          ** Reserve space in .data.      **
3          ****
4 00000000          .data
5 00000000          .space 0CCh
6
7          ****
8          ** Assemble into .text.      **
9          ****
10 00000000         .text           ; Constant into .data
11     00000000 INDEX    .set    0
12 00000000 E3A00000      MOV     R0, #INDEX
13
14          ****
15          ** Assemble into .data.      **
16          ****
17 000000cc          Table: .data
18 000000cc FFFFFFFF      .word   -1       ; Assemble 32-bit
19                           ; constant into .data.
20
21 000000d0 FF          .byte   0FFh     ; Assemble 8-bit
22                           ; constant into .data.
23
24          ****
25          ** Assemble into .text.      **
26          ****
27 00000004          .text
28 00000004 000000CC" con: .field  Table, 32
29 00000008 E51F100C      LDR     R1, con
30 0000000c E5912000      LDR     R2, [R1]
31 00000010 E0802002      ADD     R2, R0, R2
```

10.2.2 Example 2

This example uses the .int directive to initialise words.

```
1 00000000          .space 73h
2 00000000          .bss   PAGE, 128   ; Reserve 128 bytes for pointer PAGE
3 00000080          .bss   SYMPTR, 4    ; Reserve 4 bytes for pointer SYMPTR
4 00000074 E3A00056 INST: MOV    R0, #056h
5 00000078 0000000A      .int   10, SYMPTR, -1, 35 + 'a', INST, "abc"
  0000007c 00000080-
  00000080 FFFFFFFF
  00000084 00000084
  00000088 00000074'
  0000008c 00000061
  00000090 00000062
  00000094 00000063
```

10.2.3 Example 3

- This example shows how the `.long` directive initialises words.
- The symbol `DAT1` points to the first word that is reserved.

```
1 00000000 0000ABCD  DAT1: .long    0ABCDh, 'A' + 100h, 'g', 'o'  
00000004 00000141  
00000008 00000067  
0000000c 0000006F  
2 00000010 00000000'          .long    DAT1, 0AABBCCDDh  
00000014 ABBCCDD  
3 00000018             DAT2:
```

10.2.4 Example 4

- In this example, the `.word` directive is used to initialise words.
- The symbol `WORDX` points to the first word that is reserved.

```
1 00000000 00000C80  WORDX: .word    3200, 1 + 'AB', -0AFh, 'X'  
00000004 00004242  
00000008 FFFFFF51  
0000000c 00000058
```

10.3 Assembler directives for controlling conditional computations (GCC and ArmClang)

Mnemonic and syntax	Description
<code>.if condition</code>	Assembles code block if the <i>condition</i> is true.
<code>.else</code>	Assembles code block if the <code>.if condition</code> is false. When using the <code>.if</code> construct, the <code>.else</code> construct is optional.
<code>.elseif condition</code>	Assembles code block if the <code>.if condition</code> is false and the <code>.elseif condition</code> is true. When using the <code>.if</code> construct, the <code>.elseif</code> construct is optional.
<code>.endif</code>	Ends <code>.if</code> code block.
<code>.loop [count]</code>	Begins repeatable assembly of a code block. The loop count is determined by the <i>count</i> .
<code>.break [end condition]</code>	Ends <code>.loop</code> assembly if <i>end condition</i> is true. When using the <code>.loop</code> construct, the <code>.break</code> construct is optional.
<code>.endloop</code>	Ends <code>.loop</code> code block.

10.3.1 Example

The example below shows conditional assembly:

Memory for Data	1 00000001 SYM1 .set 1	2 00000002 SYM2 .set 2	3 00000003 SYM3 .set 3	4 00000004 SYM4 .set 4
Memory for Code	5	6 If_4: .if SYM4 = SYM2 * SYM2	.byte SYM4 ; Equal values	
	7 00000000 04	.else	.byte SYM2 * SYM2 ; Unequal values	
	8	.endif		
	9	10 If_5: .if SYM1 <= 10	.byte 10 ; Less than / equal	
	11	.else	.byte SYM1 ; Greater than	
	12	.endif		
	13 00000001 0A	14 If_6: .if SYM3 * SYM2 != SYM4 + SYM2	.byte SYM3 * SYM2 ; Unequal value	
	15	.else	.byte SYM4 + SYM2 ; Equal values	
	16	.endif		
	17	18 If_7: .if SYM1 = SYM2	.byte SYM1	
	19	.elseif SYM2 + SYM3 = 5	.byte SYM2 + SYM3	
	20	.endif		
	21 00000002 08			
	22			
	23			
	24			
	25			
	26			
	27 00000003 05			
	28			

10.4 Directive for creating reusable block of codes (GCC and ArmClang)

A macro definition is a series of source statements in the following format:

```
macro_name .macro [parameter_1][, ..., parameter_n]
    model_statements_or_macro_directives
    [.mexit]
    .endm
```

- **macro_name** is the name of the macro. You must place the name in the source statement's label field. Only the first 128 characters of a macro name are significant. The assembler places the macro name in the internal opcode table, replacing any instruction or previous macro definition with the same name.
- **.macro** is the directive that identifies the source statement as the first line of a macro definition. You must place **.macro** in the opcode field.
- **parameter_1, parameter_n** are optional substitution symbols that appear as operands for the **.macro** directive.
- **.mexit** is a directive that functions as a **goto .endm**. The **.mexit** directive is useful when error testing confirms that macro expansion fails and completing the rest of the macro is unnecessary.
- **.endm** is the directive that terminates the macro definition.

10.4.1 Example

Macro definition: The following code defines a macro, add3, with 4 parameters:

```
*      add3
*
*      ADDRP = P1 + P2 + P3

add3  .macro P1, P2, P3, ADDRP
      ADD ADDRP, P1, P2
      ADD ADDRP, ADDRP, P3
      .endm
```

Macro call: The following code calls the add3 macro with 4 arguments.

```
add3 R1, R2, R3, R0
```

Macro expansion: The following code shows the substitution of the macro definition for the macro call. The assembler substitutes R1, R2, R3, and R0 for the P1, P2, P3, and ADDRP parameters of add3.

```
ADD R0, R1, R2
ADD R0, R0, R3
```

11 Memory-mapped I/O devices

11.1 Data flow

- Data consists of values, symbols, addresses and instructions.
- Data flows from source memory to destination memory.
- Memory is a space which has:
 - Location
 - Address
 - Label

11.2 Basic unit of input/output data

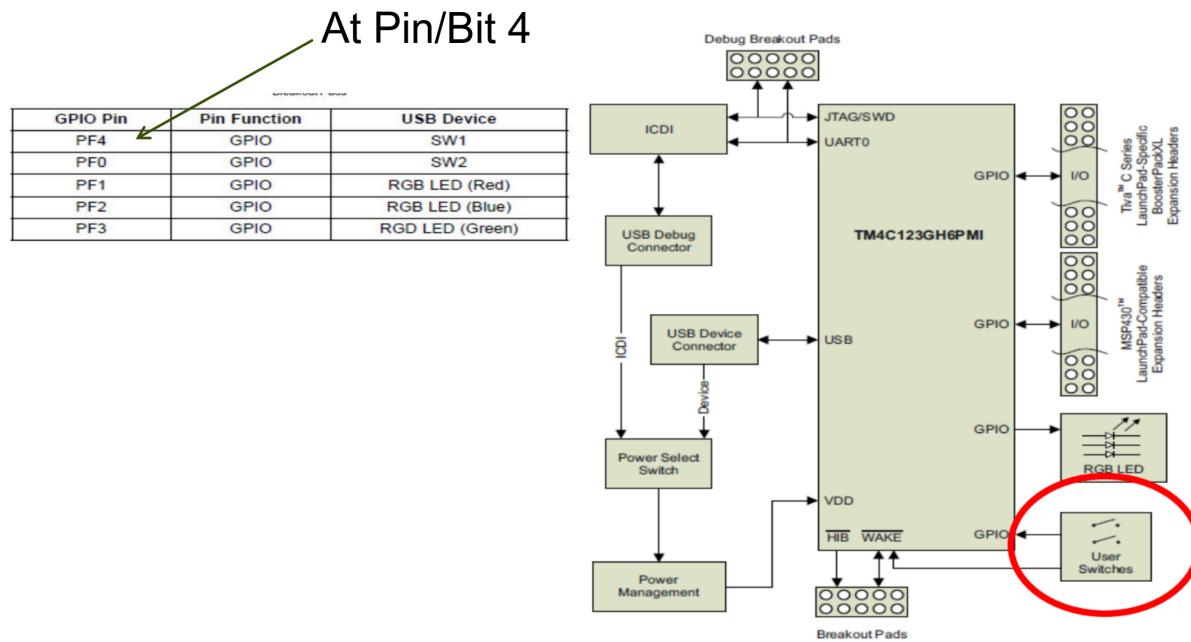
- A port has 8 pins.
- Hence, the basic unit of input/output data is one byte, or 8-bits.
- However, inside ARM Cortex, the data bus transfers 4-bytes or 32-bits per cycle.

11.3 Memory allocation

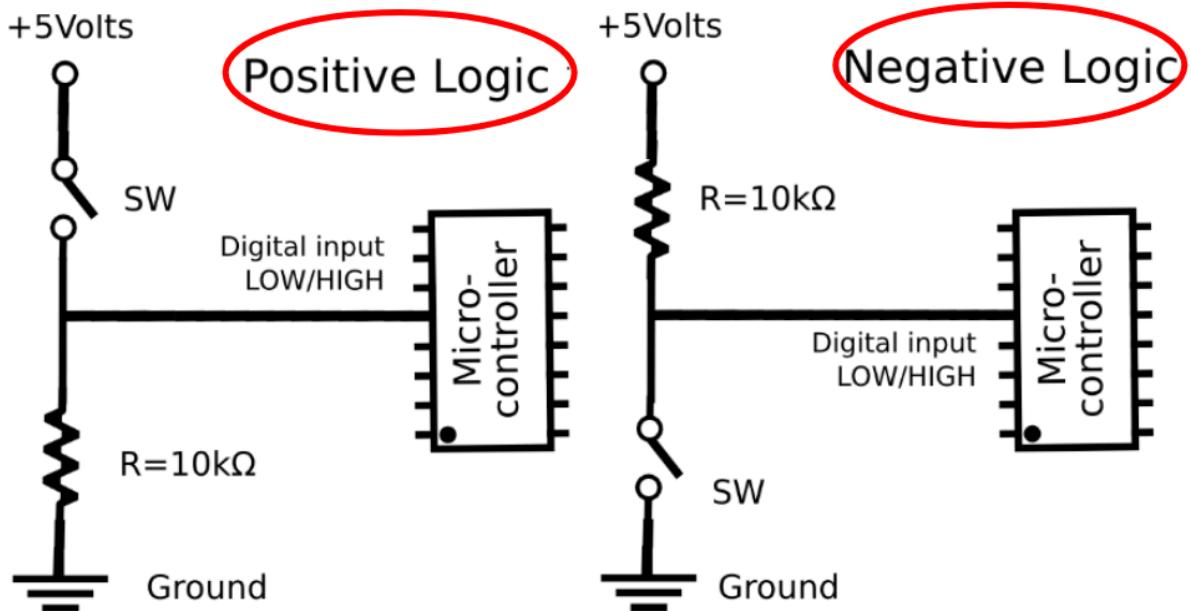
- Think of memory as a vector, like `memory[address]`.
- To store **variable** values, assign a memory label to a variable value, which indicates:
 - A fixed memory address, which is **useful** to users.
 - Multiple instances of a value, which is **needed** by users.
 - Example:
`TravelledDistance DCD 0`
- To store **fixed** values, assign a memory label to a fixed value, which indicates:
 - Multiple memory addresses, which is **not useful** to users.
 - Single instance of value, which is **needed** by users.
 - Example:
`GravityAcceleration EQU 9.8`

11.4 Example of internal data input from port F

Address of port F's data is R4 = 0x40025FFF.



Types of logic switches:



Sample code:

```
loop          BL  SSR_On   ; Call the function to switch on the LED

waitForpress1
    ; Read the status of the port PF4
    LDR R0, [R4]

    ; Check if the button is pressed
    CMP R0, #0x10

    ; Continue the loop if the button is not pressed
    BEQ waitForpress1

waitForrelease1
    ; Read the status of the port PF4
    LDR R0, [R4]

    ; Check if the button is released
    CMP R0, #0x10

    ; Continue the loop if the button is not released
    BNE waitForrelease1

    ; Call the function to switch off the LED
    BL SSR_Off

; Function to switch on LED
waitForpress2
    ; Read the status of the port PF4
    LDR R0, [R4]

    ; Check if the button is pressed
    CMP R0, #0x01

    ; Continue the loop if the button is not pressed
    BEQ waitForpress2

waitForrelease1
    ; Read the status of the port PF4
    LDR R0, [R4]

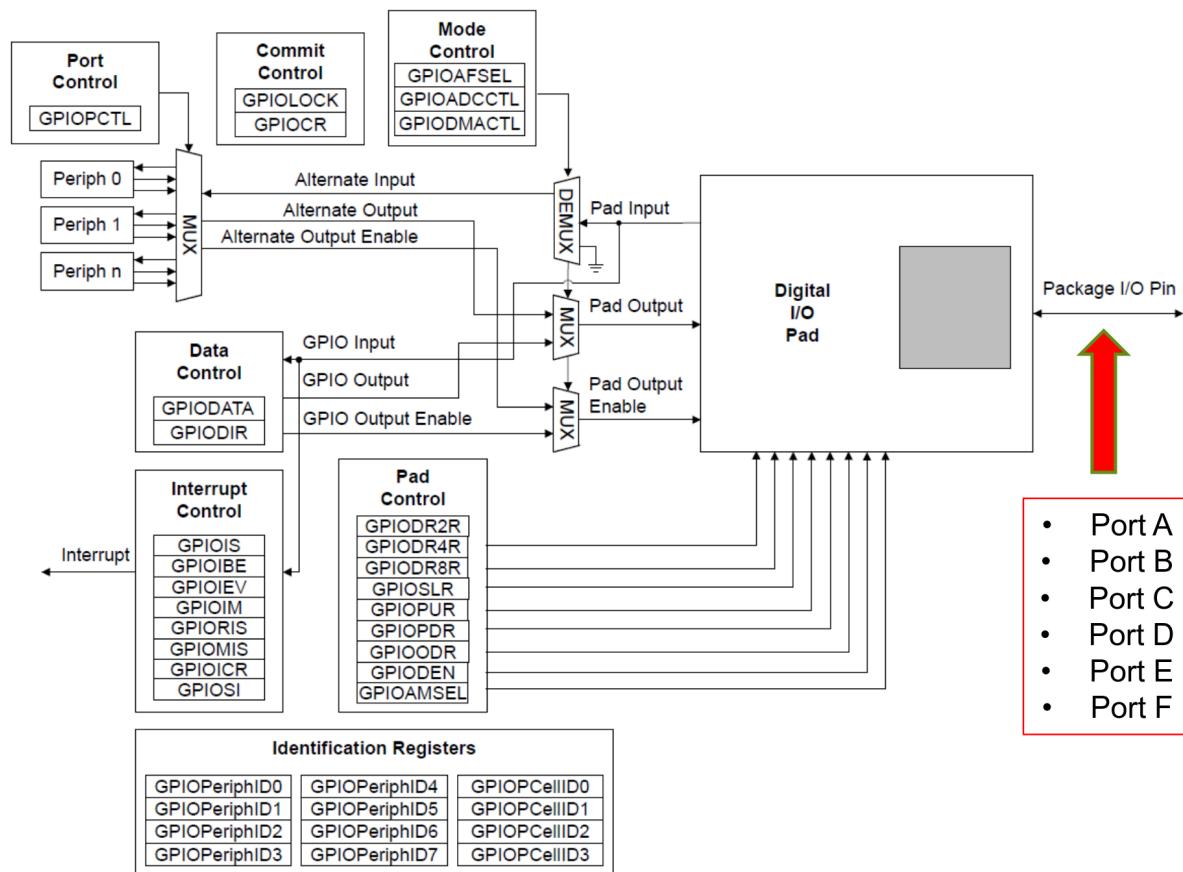
    ; Check if the button is released
    CMP R0, #0x01

    ; Continue the loop if the button is not released
    BNE waitForrelease2

    ; Go back to the start to run an infinite loop
    B loop
```

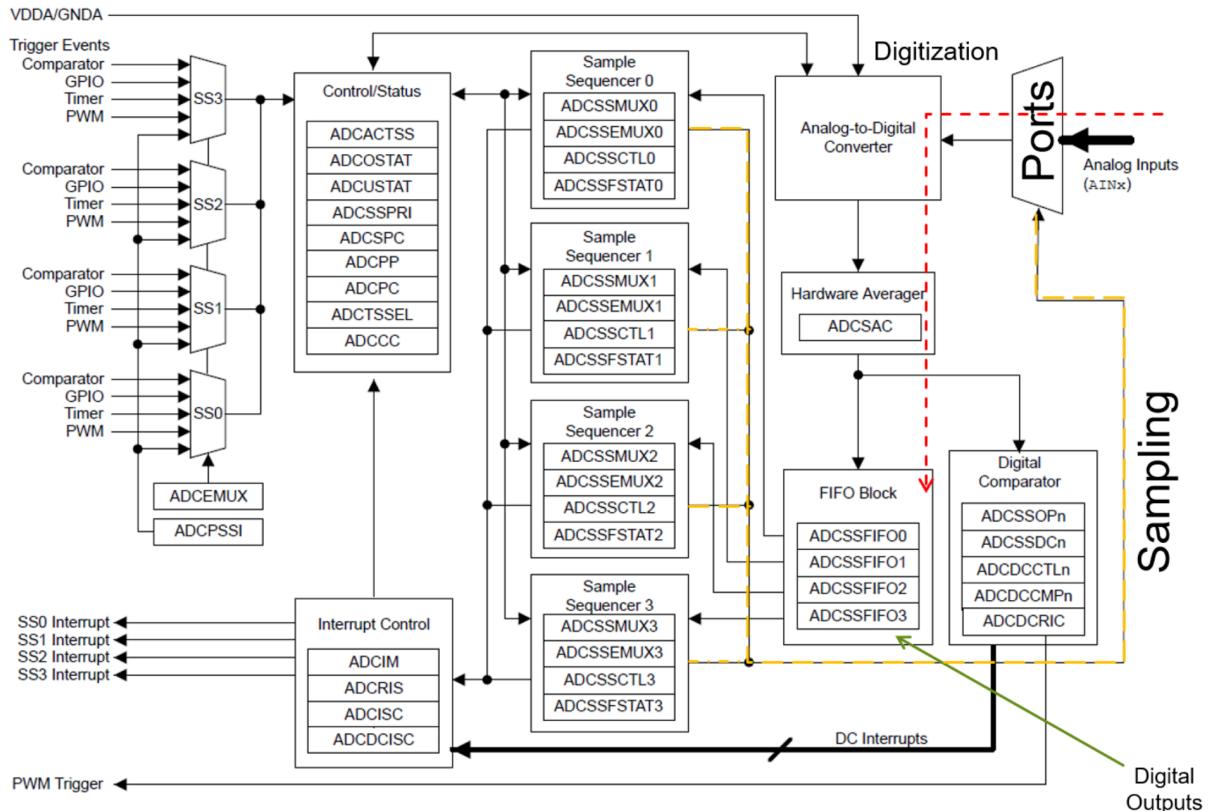
11.5 TM4C123G GPIO module

- Each GPIO port is a separate hardware instantiation of the same physical block.
- The TM4C123G allows for one interrupt per port.



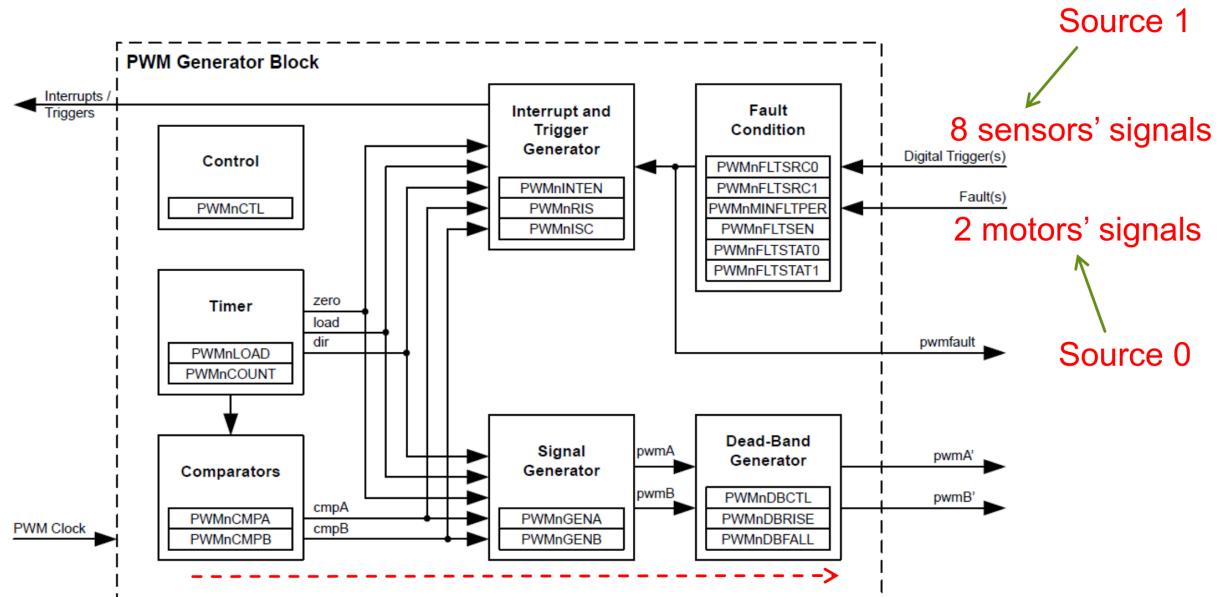
11.6 TM4C123G ADC module

Note that some pins can receive analogue values.



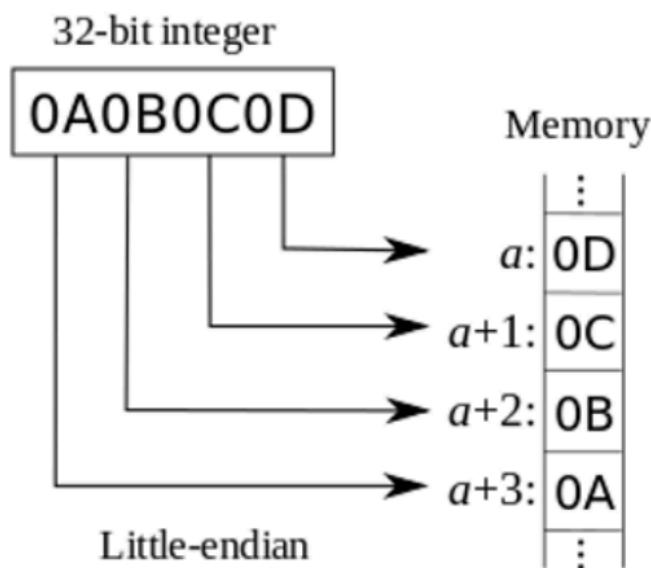
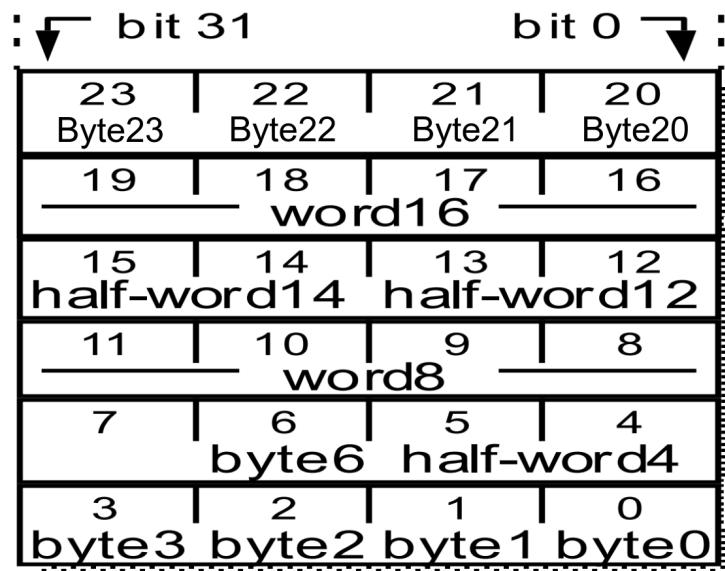
11.7 TM4C123G PWM module

TM4C123G has two PWM modules, and 4 PWM generators per module.



11.8 Data organisation nomenclature

- Data is organised into 8-bits or one byte, which is the basic unit of data.
- Data organised into 16-bits is called a half-word.
- Data organised into 32-bits is called a word.
- The size of a word depends on the size of the address bus. A 32-bit address bus would be able to handle 32-bit data.
- ARM Cortex is a 32-bit microcontroller, so a word is 32-bits.
- A memory unit in hardware is 1-byte, while a memory unit in software is 4-bytes by default.



11.9 Address indexing methods

The addressing mode is formed from two parts:

- The base register.
- The offset.

The base register can be any one of the general-purpose registers (R0 - R16), including the program counter register (PC), which allows for PC-relative addressing for position-independent code.

The offset takes one of three formats:

1. Intermediate

The offset is an unsigned number that can be added to or subtracted from the base register.

Immediate offset addressing is useful for accessing data elements that are a fixed distance from the start of the data object, such as structure fields, stack offsets and input/output registers.

For the word and unsigned byte instructions, the immediate offset is a 12-bit number. For the half-word and signed byte instructions, it is an 8-bit number.

2. Register

The offset is a general-purpose register (not the program counter, PC) that can be added to or subtracted from the base register. Register offsets are useful for accessing arrays or blocks of data.

3. Scaled register

The offset is a general-purpose register (not the program counter, PC) shifted by an immediate value, then added to or subtracted from the base register. The same shift operations used for data-processing instructions can be used (Logical Shift Left, Logical Shift Right, Arithmetic Shift Right and Rotate Right), but Logical Shift Left is the most useful as it allows an array indexed to be scaled by the size of each array element.

Scaled register offsets are only available for the word and unsigned byte instructions.

Types of address indexing:

- Offset, where the base register and offset are added or subtracted to form the memory address.
- Pre-indexed, where the base register and offset are added or subtracted to form the memory address. The base register is then updated with this new address, to allow automatic indexing through an array or memory block.
- Post-indexed, where the value of the base register alone is used as the memory address. The base register and offset are added or subtracted, and this value is stored back in the base register to allow automatic indexing through an array or memory block.

11.9.1 Pre-indexing without write-back

Pre-indexing without write-back is to use [base + offset] and **not update** the base address afterwards.

- Data: memory[base + offset]
- Base address register is **not updated**
- Code:

```
LDR R0, [R1, #4] ; C equivalent: R0 = memory[R1 + 4]
```

11.9.2 Pre-indexing with write-back

Pre-indexing without write-back is to use [base + offset] and **update** the base address afterwards.

- Data: memory[base + offset]
- Base address register: base + offset
- Code:

```
LDR R0, [R1, #4]! ; C equivalent: R0 = memory[R1 + 4]; R1 = R1 + 4
```

11.9.3 Post-indexing

Post-indexing is to use [base] and update the base address **before** using it.

- Data: memory[base]
- Base address register: base + offset
- Code:

```
LDR R0, [R1], #4 ; C equivalent: R1 = R1 + 4; R0 = memory[R1]
```

11.9.4 Example 1: Pre-indexed address without write-back

```
LDR R11, [R1, R2] ; Load R11 from the address in R1 + R2  
STRB R10, [R7, -R4] ; Store byte from R10 to the address in R7 - R4
```

11.9.5 Example 2: Pre-indexed address with write-back

```
LDR R11, [R3, R5, LSL #2] ; Load R11 from the address in R3 + (R5 * 4)  
LDR R1, [R0, #4]! ; Load R1 from R0 + 4, then R0 = R0 + 4  
STRB R10, [R7, #-1]! ; Store byte from R7 to R6 - 1, then R6 = R6 - 1
```

11.10 ARM Cortex M4 GPIO ports

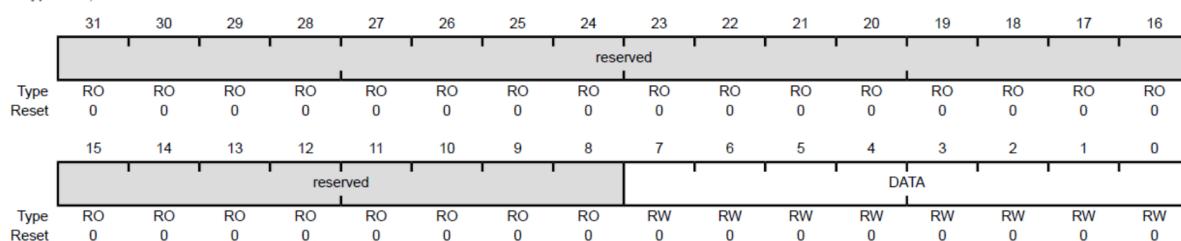
- ARM Cortex M4 has 6 maskable data registers (ports A to F).
- Note that the last 12 bits in the GPIO ports are not used for the address. They are instead used to make input and output data. By default, use you choose the offset of 0xFFFF.

GPIO Data (GPIOData)

```

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
Offset 0x000
Type RW, reset 0x0000.0000

```



- Bits 8 to 31 are reserved and are read-only. Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
- Bits 0 to 7 are GPIO data bits and are read-write. The register is virtually mapped to 256 locations in the address space. To facilitate the reading and writing of data to these registers by independent drivers, the data read from and written to the registers are masked by the eight address lines, bits 2 to 9. Reads from this register return its current state. Writes to this register only affect bits that are not masked by bits 2 to 9 and are configured as outputs.

11.10.1 Read and write example

Figure 10-3. GPIOData Write Example

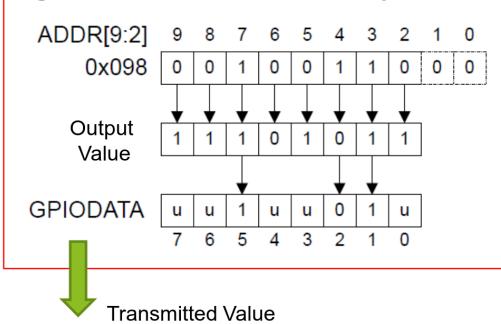
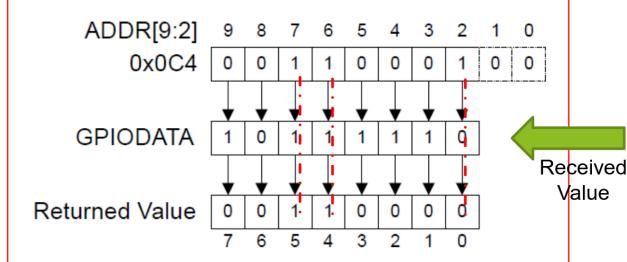
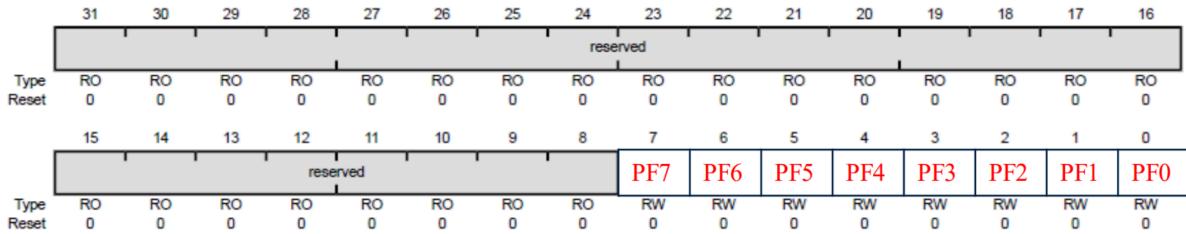


Figure 10-4. GPIOData Read Example



11.10.2 Example with port F

GPIO Data Port F address: 0x40025000



```

PORT_F_PIN_0 EQU 0x40025 004 ; ADR[9:0] = 0 000 0100
PORT_F_PIN_1 EQU 0x40025 008 ; ADR[9:0] = 0 000 1000
PORT_F_PIN_2 EQU 0x40025 010 ; ADR[9:0] = 0 001 0000
PORT_F_PIN_3 EQU 0x40025 020 ; ADR[9:0] = 0 010 0000

```

Masking port F:



This mask will result in an address of 0x40025038, which represents port F pins 1-3, as 0000 0011 1000 in binary is 038 in hexadecimal.

Port F pins:

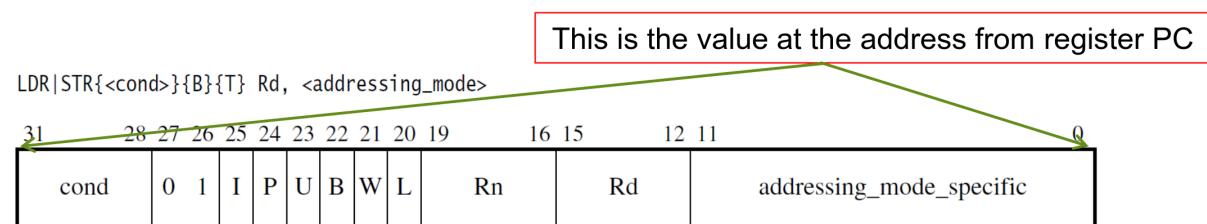
Port F register at 0x40025 000												
	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0x38	0	0	0	0	0	0	0	0	0	1	X	X
	0	0	0	0	0	0	0	0	1	0	X	X
	0	0	0	0	0	0	0	1	0	0	X	X
	0	0	0	0	0	0	1	0	0	0	X	X

11.11 Encoding of instructions in ARM

	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				
Data processing immediate shift	cond [1]	0	0	0	opcode	S	Rn	Rd	shift amount	shift	0	Rm																								
Miscellaneous instructions: See Figure A3-4	cond [1]	0	0	0	1	0	x	x	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	x	x	x	x						
Data processing register shift [2]	cond [1]	0	0	0	opcode	S	Rn	Rd	Rs	0	shift	1	Rm																							
Miscellaneous instructions: See Figure A3-4	cond [1]	0	0	0	1	0	x	x	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	x	x	x	x							
Multiplies: See Figure A3-3 Extra load/stores: See Figure A3-5	cond [1]	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	1	x	x	x					
Data processing immediate [2]	cond [1]	0	0	1	opcode	S	Rn	Rd	rotate			immediate																								
Undefined instruction	cond [1]	0	0	1	1	0	x	0	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x					
Move immediate to status register	cond [1]	0	0	1	1	0	R	1	0	Mask	SBO	rotate	immediate																							
Load/store immediate offset	cond [1]	0	1	0	P	U	B	W	L	Rn	Rd	immediate																								
Load/store register offset	cond [1]	0	1	1	P	U	B	W	L	Rn	Rd	shift amount	shift	0	Rm																					
Media instructions [4]: See Figure A3-2	cond [1]	0	1	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	x	x	x	x	x					
Architecturally undefined	cond [1]	0	1	1	1	1	1	1	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	1	1	1	x	x	x	x				
Load/store multiple	cond [1]	1	0	0	P	U	S	W	L	Rn		register list																								
Branch and branch with link	cond [1]	1	0	1	L					24-bit offset																										
Coprocessor load/store and double register transfers	cond [3]	1	1	0	P	U	N	W	L	Rn	CRd	cp_num	8-bit offset																							
Coprocessor data processing	cond [3]	1	1	1	0	opcode1				CRn	CRd	cp_num	opcode2	0	CRm																					
Coprocessor register transfers	cond [3]	1	1	1	0	opcode1	L			CRn	Rd	cp_num	opcode2	1	CRm																					
Software interrupt	cond [1]	1	1	1	1					swi number																										
Unconditional instructions: See Figure A3-6		1	1	1	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		

11.11.1 LDR and STR instructions for data I/O

- Load instructions load a single value from memory and write it to a general-purpose register.
- Store instructions read a value from a general-purpose register and store it to memory.
- Load (input to ALU) and store (output) words and unsigned bytes.



Where:

- I, P, U and W are bits that distinguish between different types of <addressing_mode>.
- L bit distinguishes between a load (L = 1) and a store instruction (L = 0)
- B bit distinguishes between an unsigned byte (B = 1) and a word (B = 0) access.
- Rn specifies the base register used by <addressing_mode>.
- Rd specifies the register whose contents are to be loaded or stored.

11.11.1.1 For sizes other than a word

Load (input to ALU) and store (output from ALU) **half-words, double-words, or unsigned bytes**.

LDR|STR{<cond>}D|H|SH|SB Rd, <addressing_mode>

31	28	27	26	25	24	23	22	21	20	19	16	15	12	11	8	7	6	5	4	3	0
cond	0	0	0	P	U	I	W	L		Rn		Rd		addr_mode	1	S	H	1	addr_mode		

Where:

- `addr_mode` are addressing mode specific bits.
- `I`, `P`, `U` and `W` are bits that specify the type of addressing mode.
- `L`, `S` and `H` are bits that combine to specify signed or unsigned loads or stores, as well as double-word, half-word or byte accesses.
- `Rn` specifies the base register used by the addressing mode.
- `Rd` specifies the register whose contents are to be loaded or stored.

12 Digital input and output (GPIO)

12.1 Digital signals

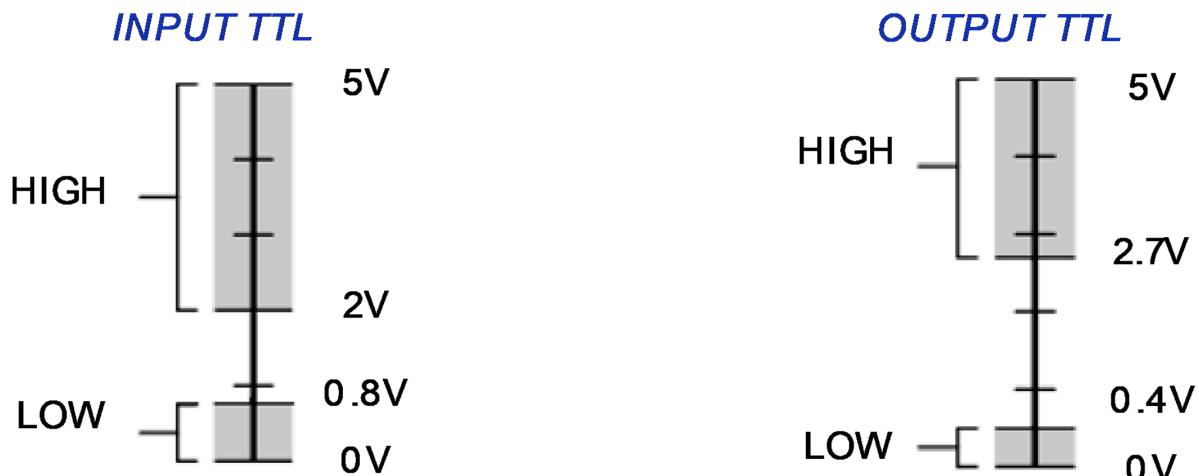
12.1.1 Logic levels

Logic	Voltage
0	0 V
1	5 V

12.1.2 Transistor-transistor logic (TTL) levels

V_{OH}	The minimum output voltage level a TTL device will provide for a HIGH signal.
V_{IH}	The minimum input voltage level to be considered a HIGH.
V_{OL}	The maximum output voltage level a device will provide for a LOW signal.
V_{IL}	The maximum input voltage level to still be considered a LOW.

12.1.3 Acceptable input digital signals (TTL levels)



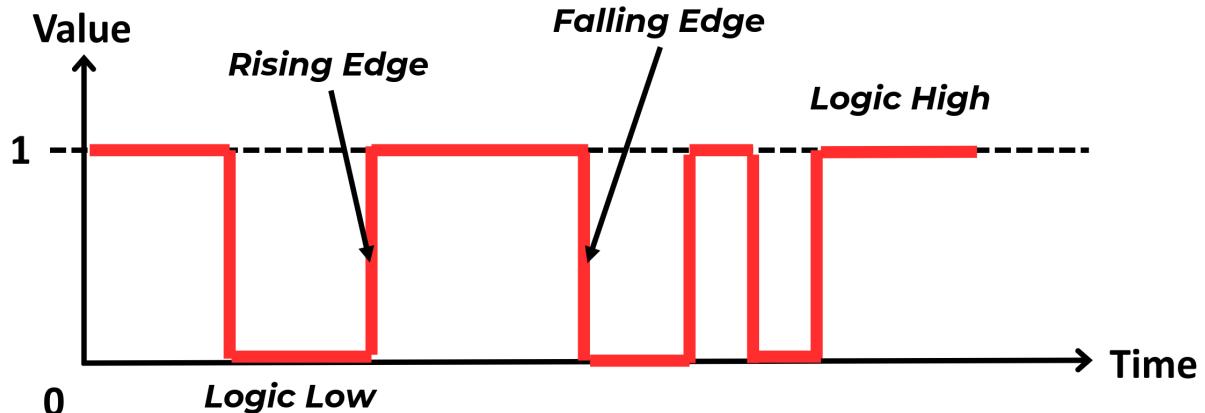
12.1.4 Representation and events of digital signals

Representation:

- Time series of 1s and 0s.
- Time series of square waves.

Events:

- Rising edge
- Falling edge
- Level of logic high
- Level of logic low



12.1.5 Signal lines of digital signals

- These are physical wires which transmit digital signals from one end to another end.
- When a microcontroller (MCU) or device sends out a digital signal through a signal line to be HIGH or LOW, such an operation is called a digital output.

12.1.6 Tri-state logic

- In digital devices a device, which can output 1, 0, or high impedance to signal lines, is called a tri-state device.
- Three-state, tri-state, or 3 state logic allows an output or input pin to assume a high impedance state.
- This effectively removes the output from the circuit, in addition to the 0 and 1 logic levels.
 - This is needed if more than 2 devices are sharing the same signal.
 - This allows multiple devices to share the same output lines.

12.1.7 Schmitt filters or triggers

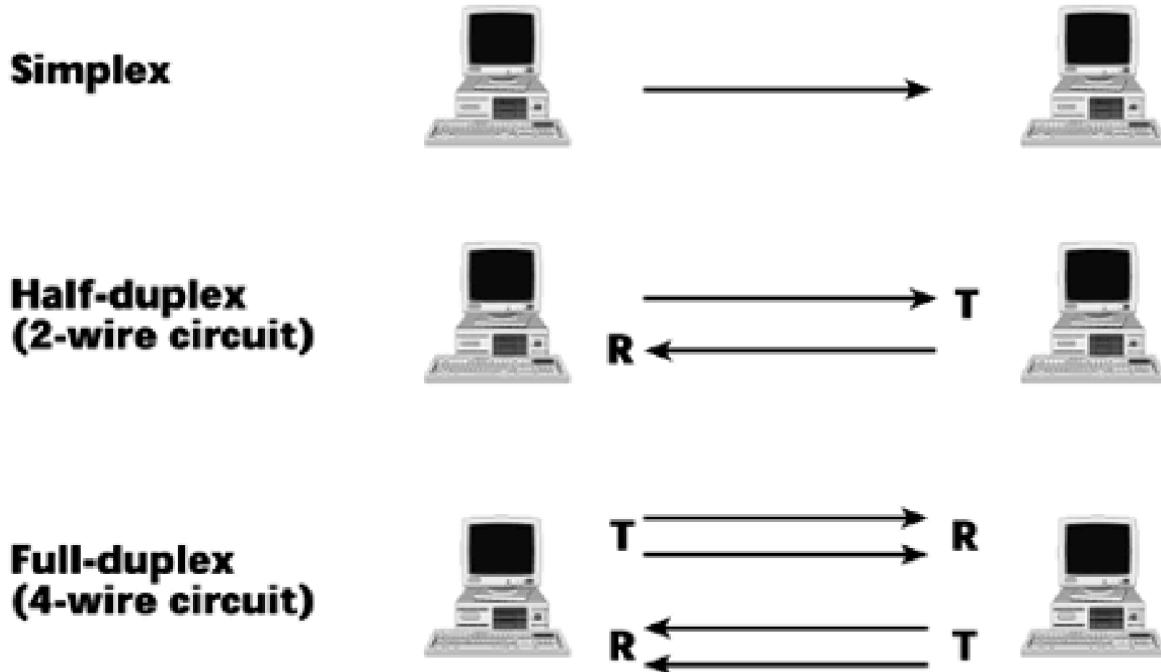
- Digital signals can be contaminated by noises due to various reasons.
- The input of digital signals at the receiver must have the ability to filter out the noises. One type of device for such purpose is called a Schmitt filter or trigger.

12.1.8 Principle of Schmitt filters

- If $V_{in} > V_{max}$, then the output logic is HIGH.
- If $V_{in} < V_{min}$, then the output logic is LOW.

12.1.9 Communication of digital signals

- Asynchronous serial output
- Synchronous serial output

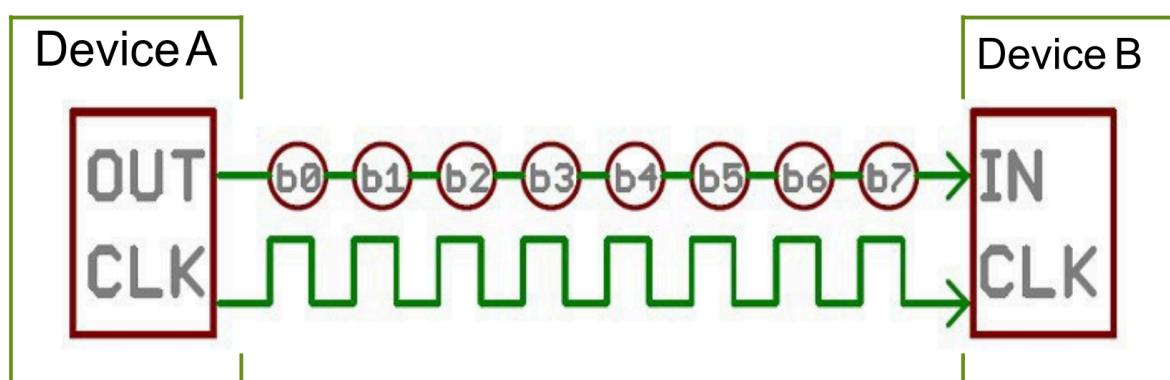


12.2 Synchronous digital input and output

12.2.1 Working principle of synchronous serial input and output

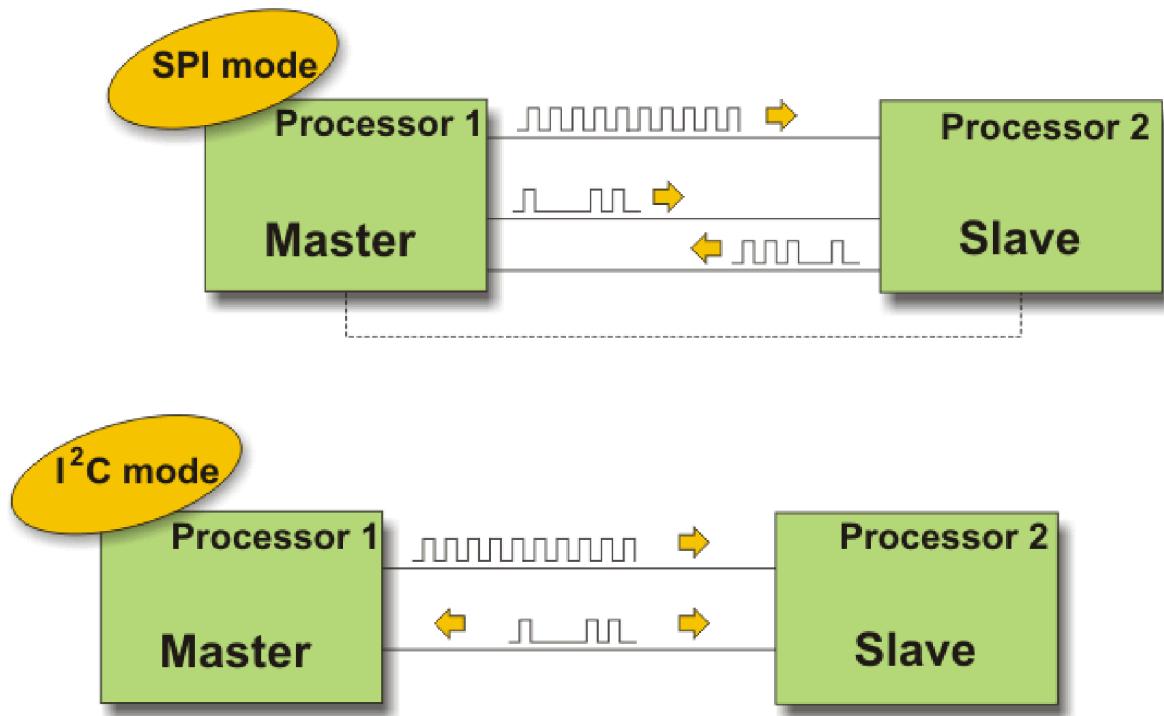
The input unit consists of:

- Shift registers
- Data registers
- Control registers
- Status registers
- Data line, clock line

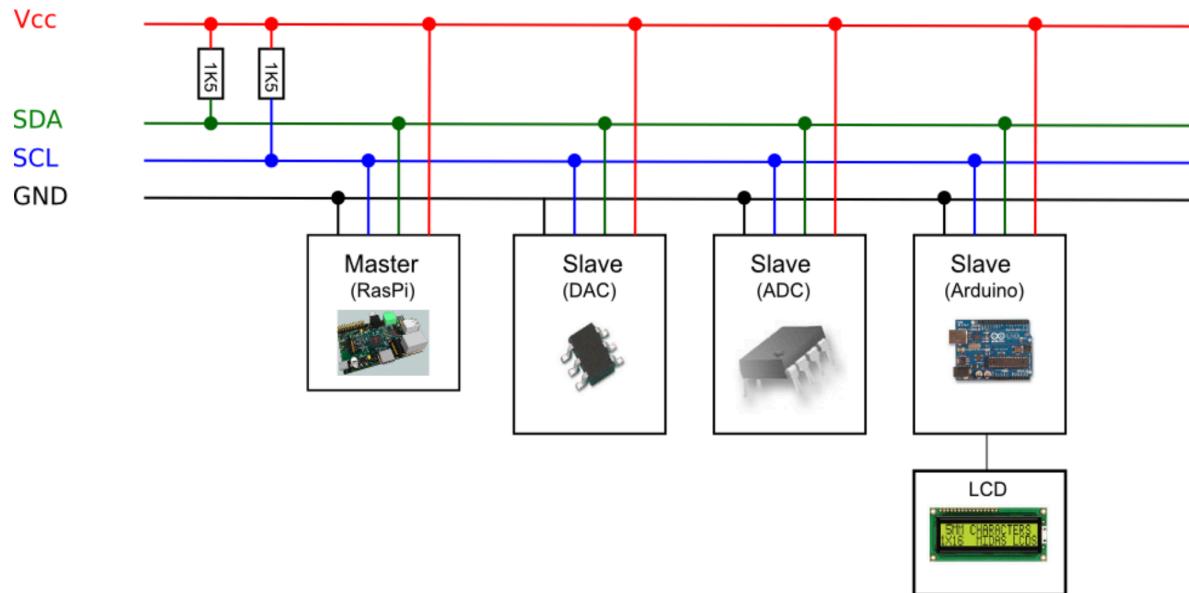


- Frame sync signal send start-pulse which tells the clock line to produce eight clock pulses.
- Each clock pulse causes one bit to be shifted into the receiver (Device B).
- Frame sync signal sends stop-pulse, and the content of the shift register is copied to the data register.

12.2.2 SPI and I²C example



12.2.3 Example of an I²C bus



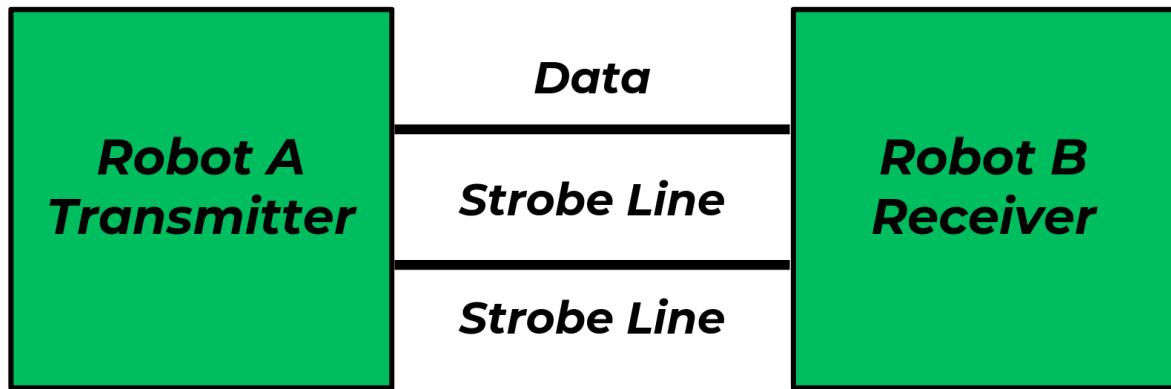
Where:

- SDA refers to Serial Data
- SCL refers to Serial Clock

12.2.4 Working principle of synchronous parallel input and output

The input unit consists of:

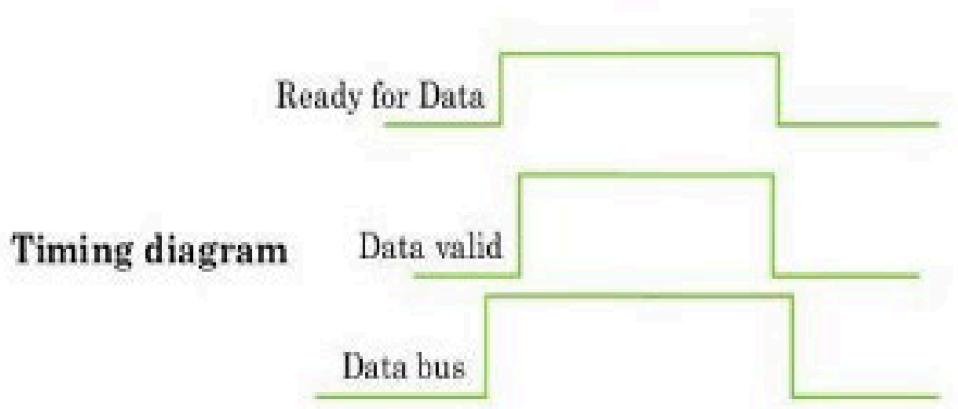
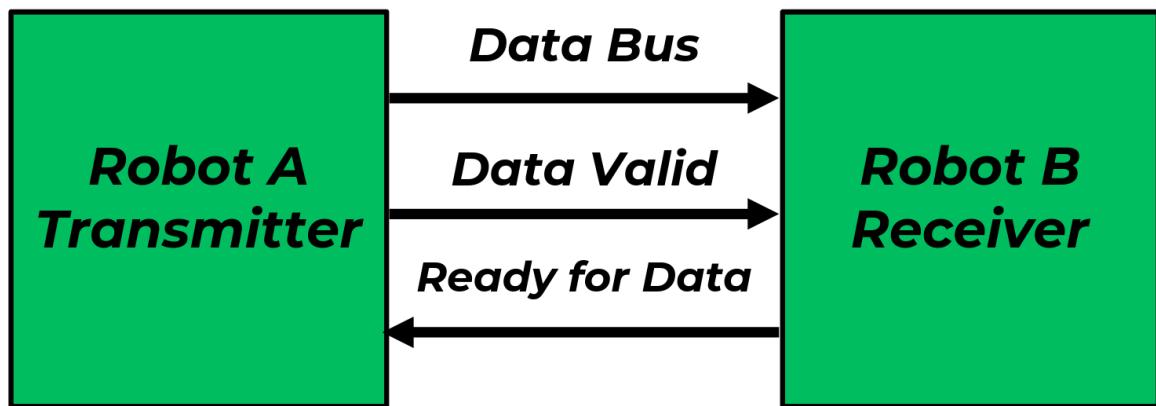
- Data registers
- Control registers
- Status registers
- Input port (8 pins, 1 byte)
- Strobe lines



If the receiver initiates the communication, it first indicates that the receiver is ready to input one byte. The transmitter then sends one byte to the data bus and indicates that one byte is ready to be read. The receiver inputs one byte from the data bus. ([Suitable for reading sensory input](#)).

If the transmitter initiates the communication, it sends one byte to the data bus and indicates that one byte is ready to be read. The receiver inputs one byte from the data bus ([Suitable for sending commands to motors](#)).

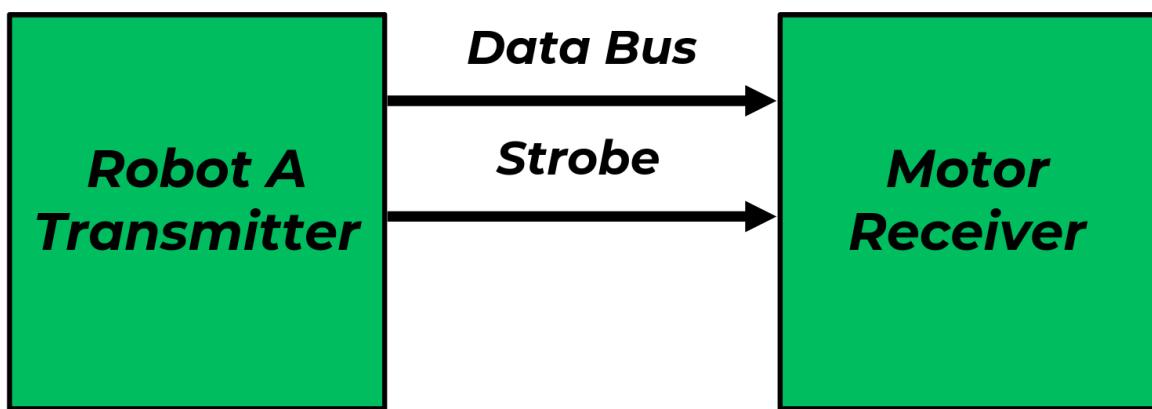
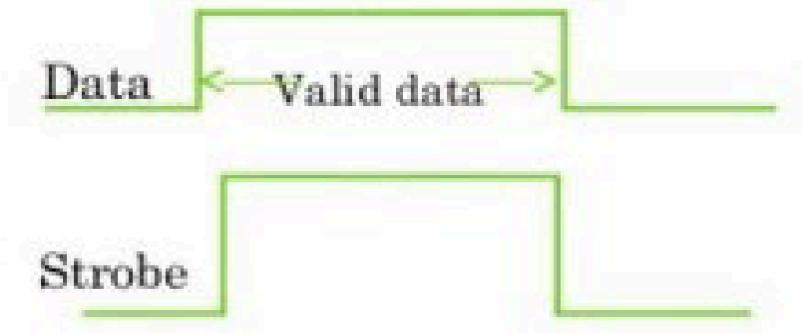
12.2.5 Receiver initiated parallel input



- Receiver first initiates that it is ready to get input.
- Transmitter sends a byte to the data bus.
- Transmitter indicates that one byte is ready on the bus.
- Receiver inputs one byte.
- The process repeats.

12.2.6 Transmitter initiated parallel input

Timing diagram



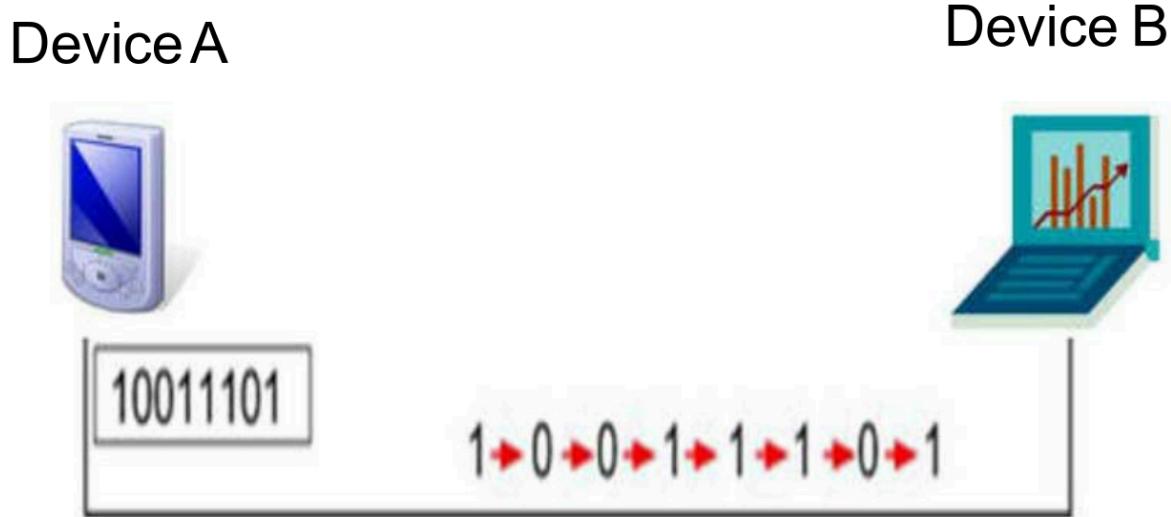
- Transmitter sends a byte to the data bus.
- Transmitter indicates that one byte is ready on the bus.
- Receiver inputs one byte.
- The process repeats.

12.3 Asynchronous digital input and output

12.3.1 Working principle of asynchronous digital input and output (data line)

The input unit consists of:

- Shift registers
- Latch registers
- Control registers
- Status registers
- Data line



- One byte is used for formatted additional bits, such as START bits, STOP bits and parity bits.
- Transmitter and receiver are configured to be running at the same clock frequency.
- The communication is initiated by the transmitter and is triggered by the START bit.
- The START bit wakes up the receiver which will receive the series of bits until the last stop bit.
The data is then latched into the latch register.

12.3.2 Working principle of asynchronous digital input and output (data line)

The input units consist of:

- Data registers
- Control registers
- Status registers
- Input port (8 pins, 1 byte)

	R/W (x)	Features							
PORTA	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0	Bit name
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	

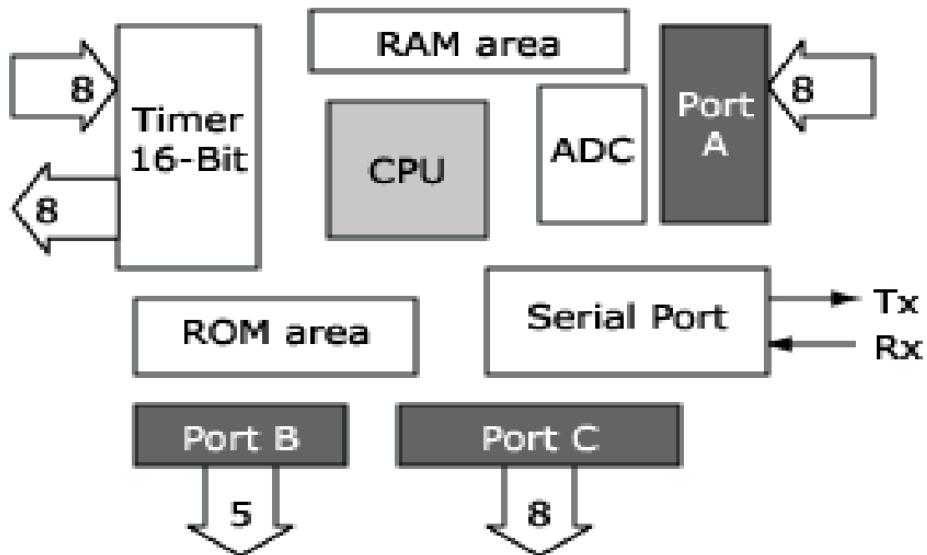
	R/W (1)	Features							
TRISA	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	Bit name
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	

Legend

R/W	Readable/Writable bit
(x)	After reset, bit is unknown
(1)	After reset, bit is set

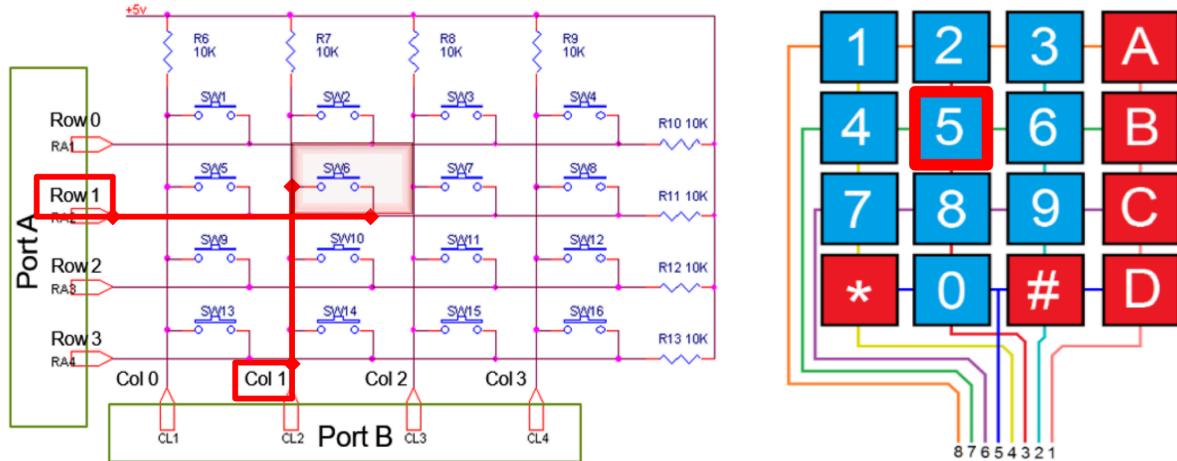
The CPU uses the control registers to configure the input port.

- The tri-state register (TRIS), controls the direction of pins in the input and output port.
- The CPU reads one byte from the input port.
- The CPU decodes the content.
- The CPU acts according to the content.



12.3.2.1 Example: Keypad

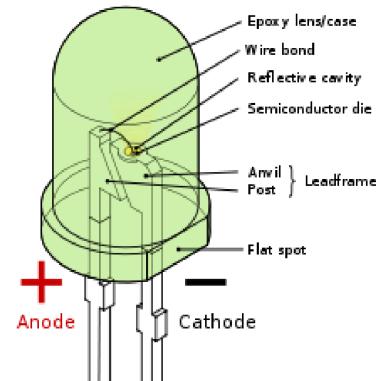
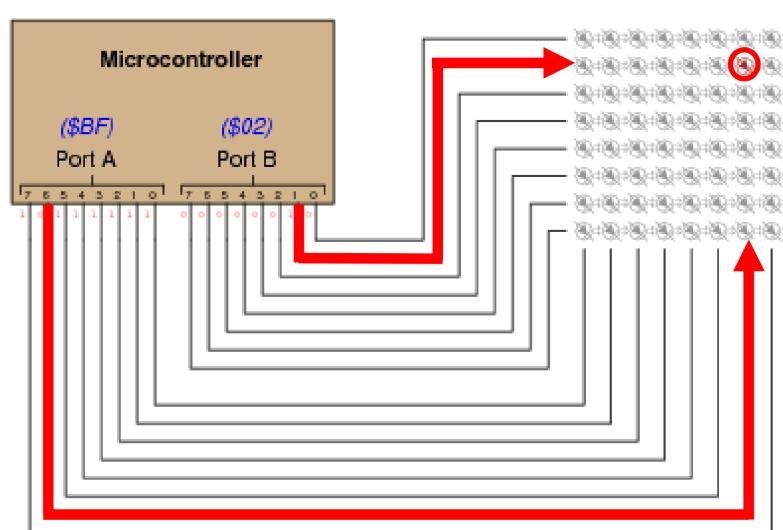
- The CPU sequentially outputs 1110, 1101, 1101, 0111 to port A.
- The CPU inputs data from port B.
- By default, all keys are not pressed.
- When a key is pressed, the corresponding column line will be in logic low (i.e. “0”).



When port A is 1101, and port B is 1101, the keypad registers 5.

12.3.2.2 Example: LED dot matrix

- CPU outputs \$BF (0xBF) to port A and \$02 (0x02) to port B.
- The LED at row 1 and column 6 will light up.



13 Analogue input and outputs

13.1 Types of analogue inputs

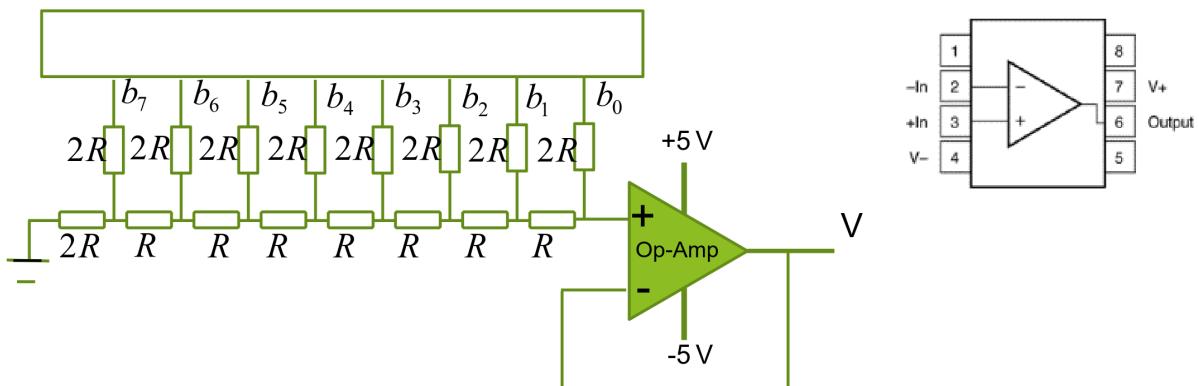
1. Visual
2. Auditory
3. Motion

13.2 Types of analogue outputs

1. Lights or displays
2. Sounds
3. Motion, motors and sensors

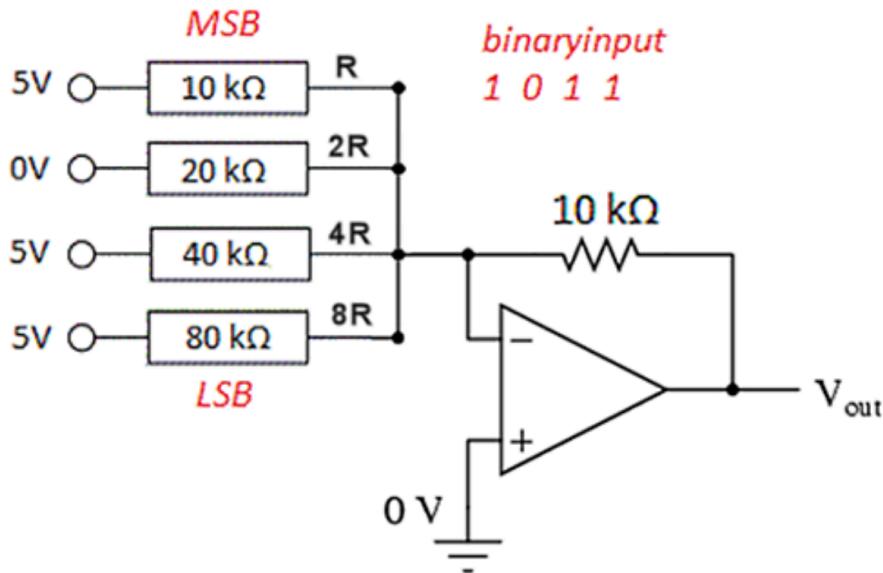
13.3 Working principle of a digital to analogue converter (DAC)

1. Configure the digital to analogue converter.
 2. Read a digital number.
 3. Convert the digital number into an analogue value.
 4. Output the result.
- The digital to analogue converter has data registers, control registers and status registers.
 - After configuration, the digital to analogue converter reads a digital number from the data register.
 - The data register is connected to a network of resistors.
 - The output voltage from the resistor network is connected to an operational amplifier (op-amp).
 - The voltage output from the operational amplifier is the analogue output value.



13.3.1 Exercise

A 4-bit DAC converts these digital numbers into analogue voltages. If we use the circuit shown below, what is the analogue voltage if the digital value is 1011?



$$V_{\text{out}} = -10k \times \left(\frac{5}{10k} + \frac{0}{20k} + \frac{5}{40k} + \frac{5}{80k} \right)$$

$$V_{\text{out}} = -5 + 0 + \frac{5}{4} + \frac{5}{8}$$

$$V_{\text{out}} = -5 + 1.25 + 0.625$$

$$V_{\text{out}} = -6.875 \text{ V}$$

13.4 Specification of DACs

- Range of input value: N bits
- Range of output value: V_{min} (V) to V_{max} (V)
- DAC's digital resolution:

$$\text{Digital resolution} = \frac{\text{Analogue Range}}{\text{Digital Range}}$$

- DAC's analogue resolution:

$$\text{Digital resolution} = \frac{\text{Digital Range}}{\text{Analogue Range}}$$

13.4.1 Exercise

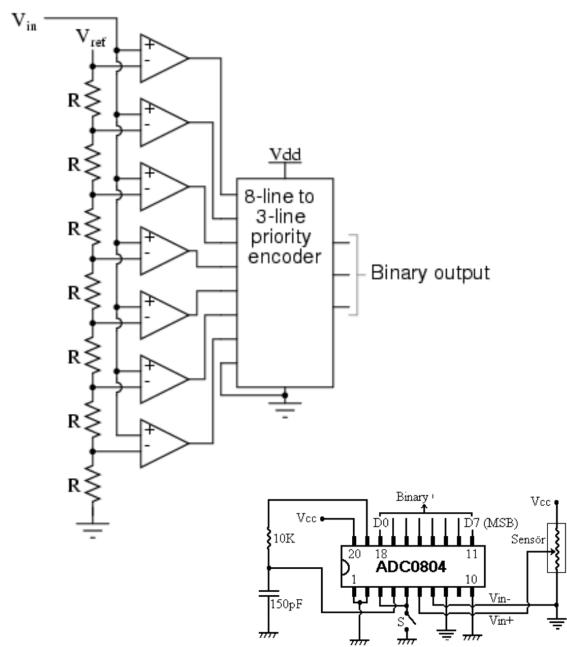
A microcontroller's DAC takes 12-bit numbers as input and produces an analogue output which varies from 0.0 V to 5.0 V. What is the DAC's digital and analogue resolution?

$$\text{Digital resolution} = \frac{5.0 - 0.0}{2^{12}} = 1.2 \times 10^{-3} \text{ V bits}^{-1}$$

$$\text{Analogue resolution} = \frac{2^{12}}{5.0 - 0.0} = 819 \text{ bits V}^{-1}$$

13.5 Working principle of an analogue to digital converter (ADC)

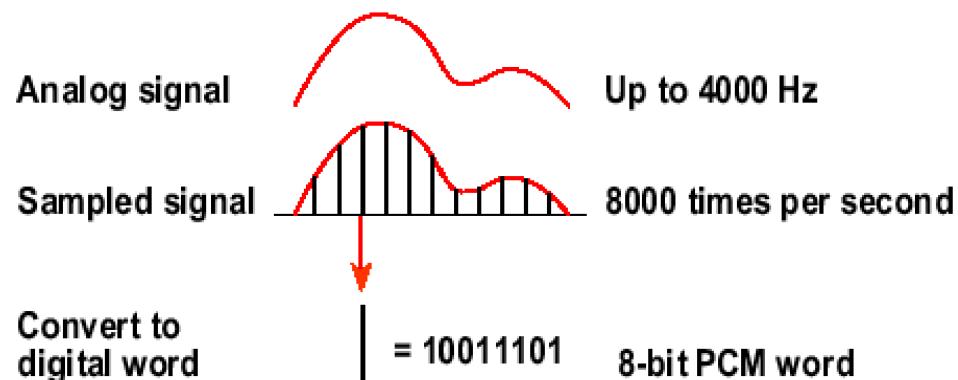
- The analogue to digital converter has data registers, control registers and status registers.
- After configuration, the analogue to digital converter runs a counter which incrementally outputs a digital number.
- The digital number is converted to an analogue voltage by a digital to analogue converter.
- The digital to analogue voltage is compared with the input's analogue voltage.
- When the above two voltages are equal, the analogue to digital converter stops the counter.
- The digital number of the counter is the output of the analogue to digital converter.



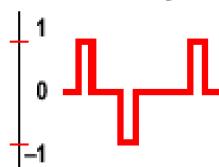
13.5.1 Example: Speech signals

Pulse-code modulation (PCM) is a method used to digitally represent sampled analogue signals. It is the standard form of digital audio in computers.

The 64-kbps voice channel from PCM



8 bits X 8000 samples = 64 Kbps = a digital voice channel



MEMS microphones produce an electrical signal from the change in capacitance caused by the movement of a membrane relative to a stationary plate.



13.6 Specification of ADCs

- Range of input value:
 - High reference voltage (V_{refh}) to low reference voltage (V_{refl})
- Range of output value: N bits
- ADC's digital resolution:

$$\text{Digital resolution} = \frac{\text{Analogue Range}}{\text{Digital Range}}$$

- ADC's analogue resolution:

$$\text{Digital resolution} = \frac{\text{Digital Range}}{\text{Analogue Range}}$$

13.6.1 Exercise

A rotary potentiometer outputs voltages which are within the interval of [0.0, 3.0] (V). We use a 12-bit ADC to convert the analogue values of the potentiometer into digital numbers. What is the ADC's digital and analogue resolution?

$$\text{Digital resolution} = \frac{3.3 - 0.0}{2^{12}} = 0.8 \times 10^{-3} \text{ V bits}^{-1}$$

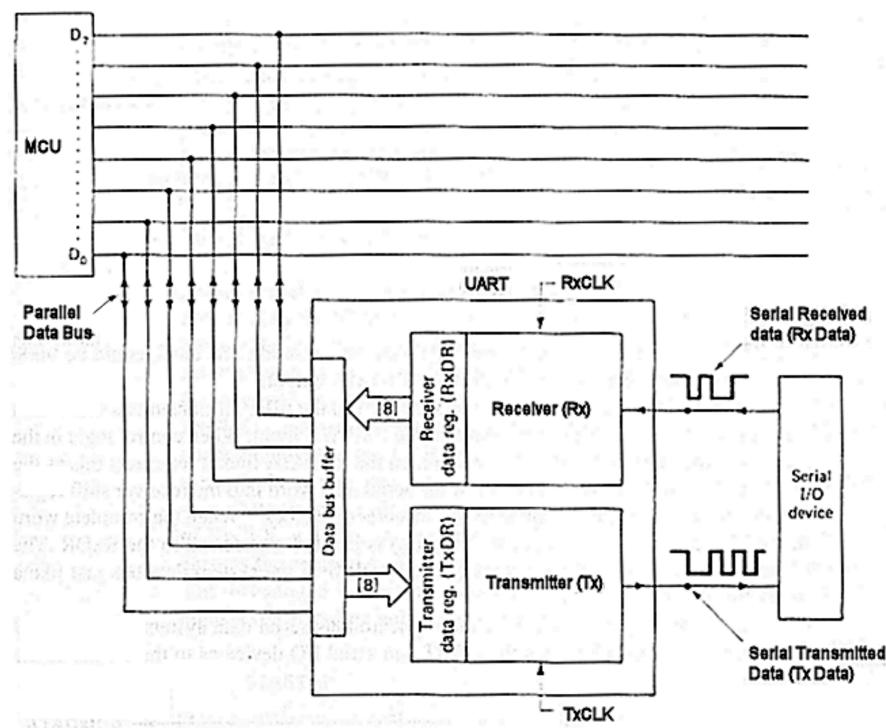
$$\text{Analogue resolution} = \frac{2^{12}}{3.3 - 0.0} = 1241 \text{ bits V}^{-1}$$

14 Universal asynchronous receiver/transmitter (UART)

- UART is the interface device that implements the transmission of serial data between two devices.
- Data terminal equipment (DTE) is the computer.
- Data communication equipment (DCE) is the device, like a modem, printer, etc.
- It is used for serial and parallel conversions for communication between microcontrollers (MCU) and serial I/O devices.
- Its purpose is to translate bytes of data into serial forms suitable for transmitting (TX) and converting the serial data back to bytes of data, or receiving (RX).
- UARTs are commonly included in microcontrollers.
- Data format and transmission speeds are configurable.
- The data format follows the communication standards of EIA, RS232, RS485, etc.
- Communications may be in:
 - Simplex, which is one direction only.
 - Duplex, which is both directions simultaneously.
 - Half duplex, which is RX and TX taking turns, one at a time.
- Has 3 lines, Tx, Rx and ground (GND).

14.1 Basic elements of a parallel/serial interface

- A serial receiver (Rx).
 - It converts a serial format into parallel format, and store it in the receiver data register (RxDR) for eventual transmission to the MCU.
- A serial transmitter (Tx).
 - It takes a parallel word from the transmitter data register (TxDR) and converts it into a serial format for transmission.
- A bidirectional data bus buffer.
 - It passes data from the microcontroller to the transmitter data register, or from the receiver data register to the microcontroller over the system data bus.
- External clocks such as the receiver clock (RxCLK) and the transmitter clock (TxCLK).



14.2 UART connection to external device

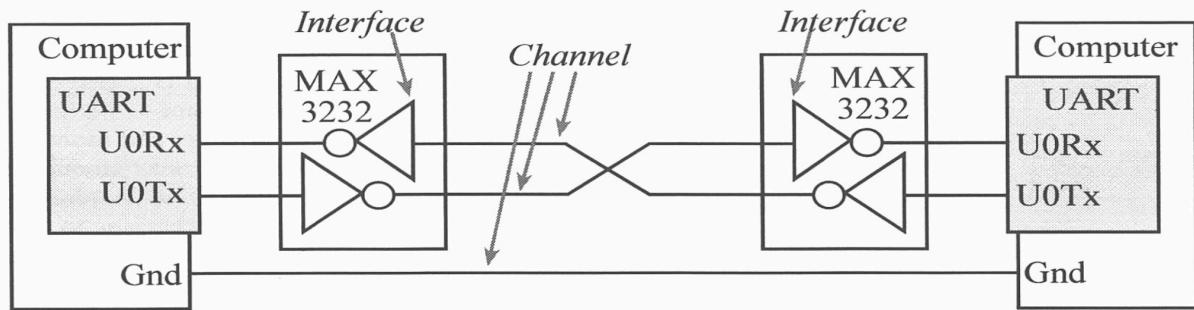


Figure 1: Serial channel connecting two data terminal equipment (DTE) devices.

To improve bandwidth, increase range, and reduce noise, we use the TI MAX3232 chip line driver to bring the voltages to transistor-transistor logic (TTL) requirements.



MAX3232 is a 3 V to 5.5 V multichannel RS-232 line driver/receiver with ± 15 kV ESD protection.

14.3 Data communication (serial)

- Asynchronous communication
 - Transmitter can send data to the receiver at any time.
 - The time delay between the transmission of two words may be indeterminate.
 - Transmitter's clock need not be synchronised with the receiver's clock.
- Synchronous communication
 - Transmitter sends data to receiver continually.
 - Data words are sent in blocks, and separated by sync characters.
 - Sync characters are used by the receiver to synchronise its clock to that of the transmitter.
 - Transmitter sends sync characters continually when there is no data to send.

14.4 Asynchronous serial format (data framing)

- A data frame is a complete and nonvisible packet of information, in bits.
- A data frame includes information, which is data or characters, and overhead, like start, stop, and error checking bits.
- A data frame is the smallest data packet that can be transmitted and understood.

14.4.1 Standard format

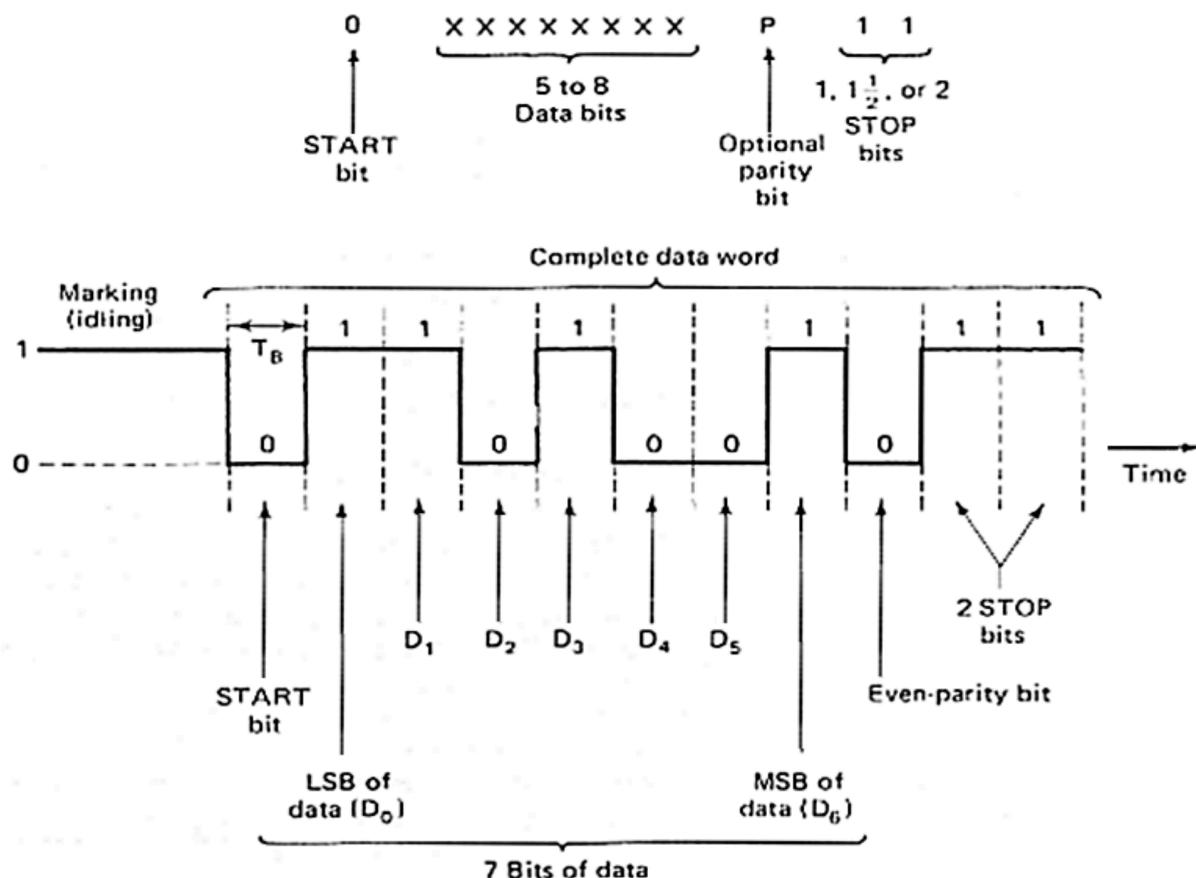
Bit number	1	2	3	4	5	6	7	8	9	10	11
	Start bit	5–8 data bits									Stop bit(s)
	Start	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7		Stop

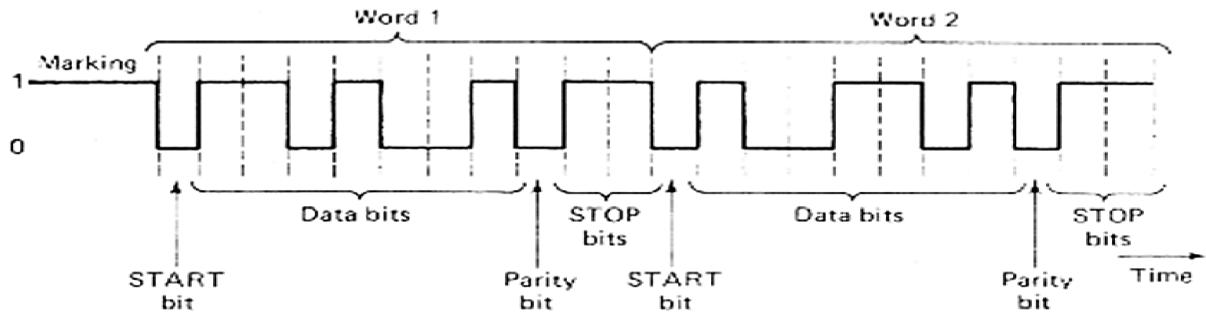
- A START bit at index 0.
- 5 - 8 data bus, and the least significant bit (LSB) is transmitted first.
- Optional parity bit for error detection, which can either be even or odd parity.
- Example of an even parity check:
 - Sum of all bits in 1 transmitter data word must be even.
 - There is an error detected at the receiver if the sum is not even.
 - 7 bit data words, like ASCII, and 1 parity bit:

$\underbrace{1010 \quad 100\color{red}{1}}_{\text{Odd}}$ $\underbrace{1000 \quad 1110}_{\text{Even}}$

The bits in red are the parity bits.

- 1 or 2 stop bits.





- The START bit signals to the receiver that a new character is coming.
- The next 5 to 8 bits represent the character.
- If a parity bit is used, it would be placed after all the data bits.
- The next one or two bits always in the mark (logic high, i.e. "1") condition and are called the stop bits. They signal to the receiver that the character is completed.
- Since the START bit is logic low (0), and the stop bit is logic high (1) there are always at least two guaranteed signal changes between characters.

14.5 ASCII format

ASCII stands for American Standard Code for Information Interchange.

	MSB									
	HEX	0	1	2	3	4	5	6	7	
HEX	BIN	000	001	010	011	100	101	110	111	
0	0000	(NUL)	(DLE)	Space	0	@	P	'	p	
1	0001	(SOH)	(DC1)	!	1	A	Q	a	q	
2	0010	(STX)	(DC2)	"	2	B	R	b	r	
3	0011	(ETX)	(DC3)	#	3	C	S	c	s	
4	0100	(EOT)	(DC4)	\$	4	D	T	d	t	
5	0101	(ENQ)	(NAK)	%	5	E	U	e	u	
6	0110	(ACK)	(SYN)	&	6	F	V	f	v	
7	0111	(BEL)	(ETB)	'	7	G	W	g	w	
8	1000	(BS)	(CAN)	(8	H	X	h	x	
9	1001	(HT)	(EM))	9	I	Y	i	y	
A	1010	(LF)	(SUB)	*	:	J	Z	j	z	
B	1011	(VT)	(ESC)	+	;	K	[]	k	{	
C	1100	(FF)	(FS)	,	<	L	\	l		
D	1101	(CR)	(GS)	-	=	M]	m	}	
E	1110	(SO)	(RS)	.	>	N	^	n	~	
F	1111	(SI)	(US)	/	?	O	_	o	DEL	

14.6 Baud rate (data transmission speed)

- Baud rate is the rate of data transmission in serial communication.

$$\text{Baud rate} = \frac{1}{\text{Time between transition (Baud)}}$$

Data rate = Rate of data transmission (bits s⁻¹ or Mb s⁻¹)

- In most simple serial communication, baud rate is equal to the data rate.
- Common baud rates include:
 - ▶ 115200
 - ▶ 57600
 - ▶ 19200
 - ▶ 14400
 - ▶ 9600
 - ▶ 4800
 - ▶ 2400
 - ▶ 1200
 - ▶ etc.

14.6.1 Example

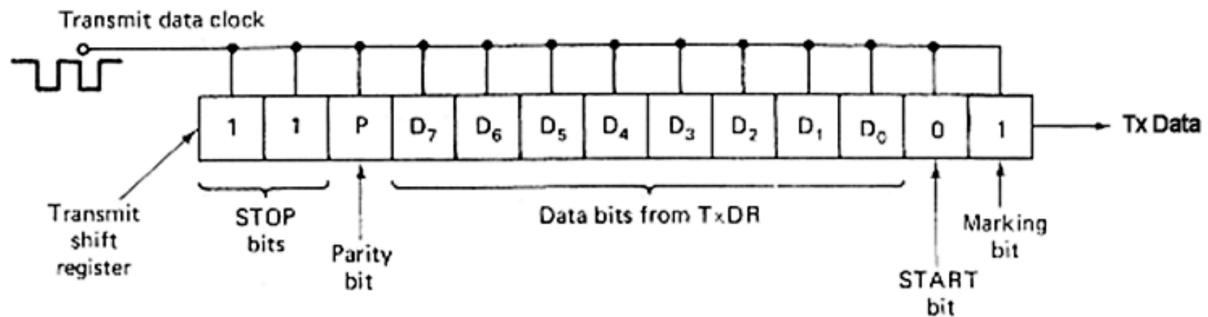
If the baud rate is equal to the data rate of 9600 baud, what is the time duration of 1 bit?

$$\text{Data rate} = 9600 \text{ bits s}^{-1}$$

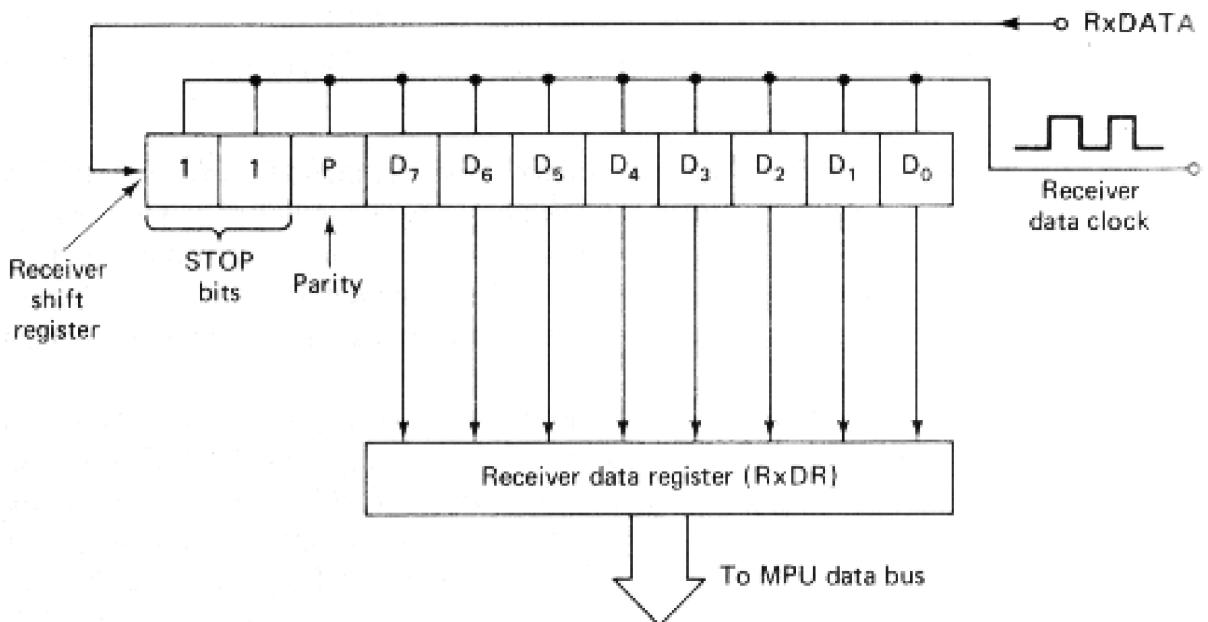
$$\text{Bit time} = \frac{1}{9600} = 104.17 \mu\text{s}$$

14.7 UART data transmission

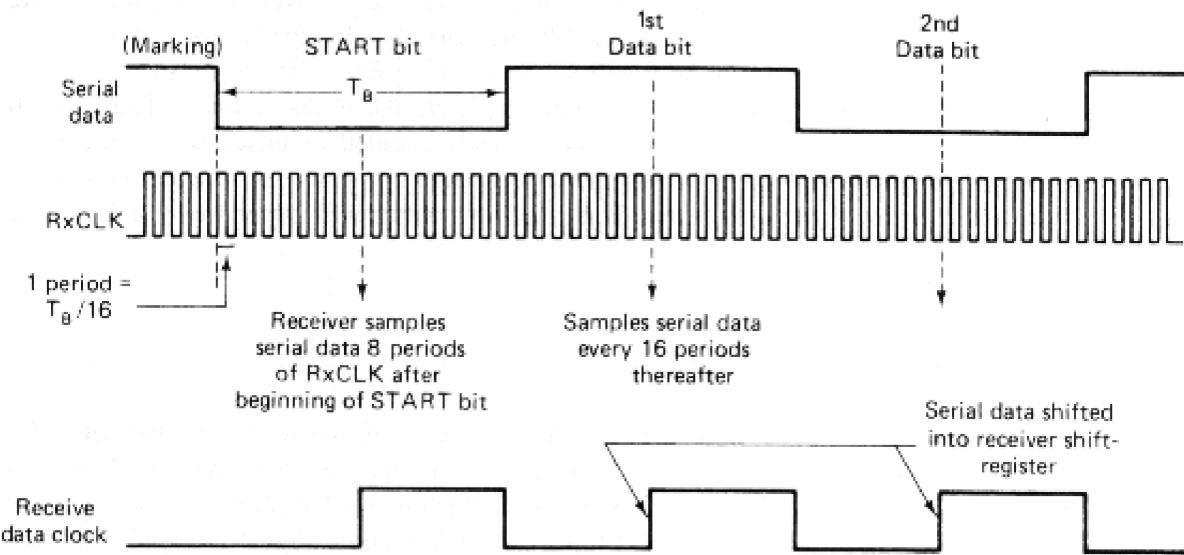
- To transmit a data word from the MCU to a serial output device:
 - MCU sends the data word to the UART's transmitter data register (TxDR).
 - Control logic in the Tx section takes this data word, frames it with a START bit, parity bit, and stop bits, and then places it in the transmit shift register.
 - Contents of this shift register are shifted right at a rate determined by the transmit data clock, where the clock frequency is the baud rate.



- To send a data word from a serial input device to the microcontroller:
 - Serial input device transmit a serial data word to the UART's receiver section through the RxDATA input.
 - Upon receiving a START bit on the RxDATA line, the Rx section shifts the remainder bits of the data word into the receiver shift register.
 - The transmission rate is determined by the receiver data clock.
 - When the complete word is in the register, the data portion (D₇ - D₀) is transferred in parallel to the RxDR.
 - The microcontroller reads the contents of RxDR like other memory locations.



14.8 Syncing receiver section with serial data



To ensure reliable data transmission, the receiver needs to sample the signal at a rate that is high enough to capture the changes in the signal.

The baud rate determines the frequency of the signal changes, so the sampling rate needs to be a multiple of the baud rate.

Sampling at 16× the baud rate provides several benefits:

1. **Accurate data recovery:** Sampling at 16× ensures that the receiver can accurately detect the start and stop bits, as well as the data bits.
2. **Noise tolerance:** Sampling at a higher rate helps to reduce the impact of noise on the signal, making the transmission more reliable.
3. **Clock recovery:** The 16× sampling rate allows the receiver to recover the clock signal from the transmitter, ensuring proper synchronisation.

While other sampling rates are possible, 16× has become the standard for UART communication due to its balance between accuracy, noise tolerance, and clock recovery.

- External clock of UART is 16 times faster than the baud rate, i.e. 1 period of RxCLK = $\frac{T_B}{16}$
 - For example:

$$\text{Baud rate} = 9600$$

$$\text{RxCLK} = 16 \times 9600 = 153.6 \text{ kHz}$$

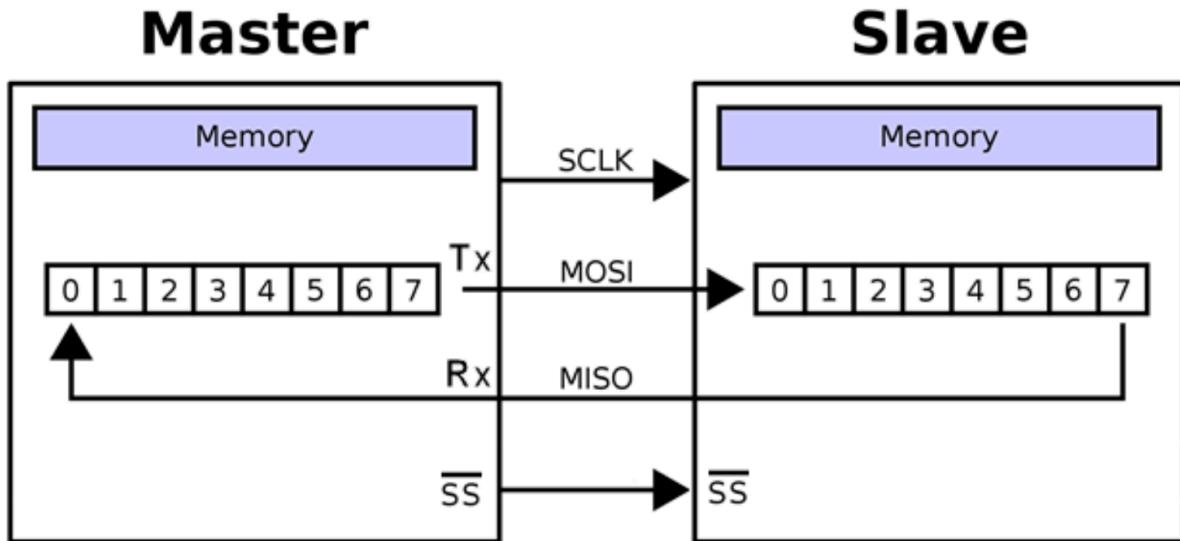
- RxCLK set to divide by 16 (MOD 16) when samples are shifted to the shift register. This makes the frequency of the received data clock be the same as the baud rate.

14.8.1 Syncing procedure

1. After sensing the first negative going transition (falling edge), the receiver waits for 8 cycles of RxCLK, then samples to see if the serial input is still low, to ensure that it is a START bit.
2. Receiver samples the serial data at every 9th cycle of the 16 RxCLK cycles (~1 bit time)
3. Each sample is shifted into the receiver shift register on the rising edge of the receiver's data clock.
4. UART checks if STOP bits are 1s, if not, a **framing error** flag is set.
5. UART checks for parity to be even, if not, a **parity error** flag is set.
6. Data bits are transferred in parallel to the RxDR.
7. Receiver continues to shift data bits into the receiver shift register.
8. This second word will not be transferred into the RxDR until the first word is read by the microcontroller.
9. If the third word arrives before the first word is cleared, it will overwrite the second word. An **overrun error** flag will be set.
10. This can be avoided with the correct baud rate setting.

15 Synchronous serial interface (SSI)

- Another common name for synchronous serial interface is serial peripheral interface (SPI).
- SSI is widely used as a serial interface between an absolute position sensor and a microcontroller.
- SSI uses a clock pulse train from a microcontroller to initiate a gated output form the sensor.
- Operates as master or slave.
- Channels can operate as one master to a single or multiple slaves.
- Communication is always in duplex mode.



- Tx (Master out, Slave in, MOSI) data line is driven by the master and received by the slave.
- Rx (Master in, Slave out, MISO) data line is driven by the slave and received by the master.

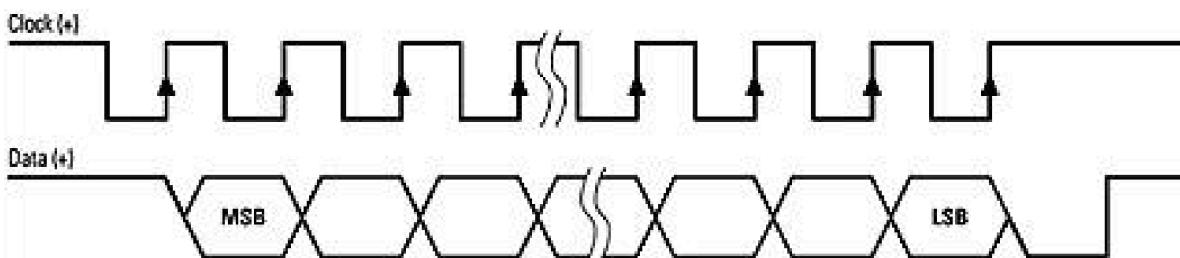


Figure 2: SSI timing diagram

- Transmitting devices uses the edge of one clock to change the output and the receiving device uses other edge to accept the data.
- Classified as synchronous devices as one hardware clock is shared between the devices.

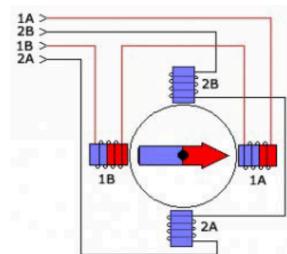
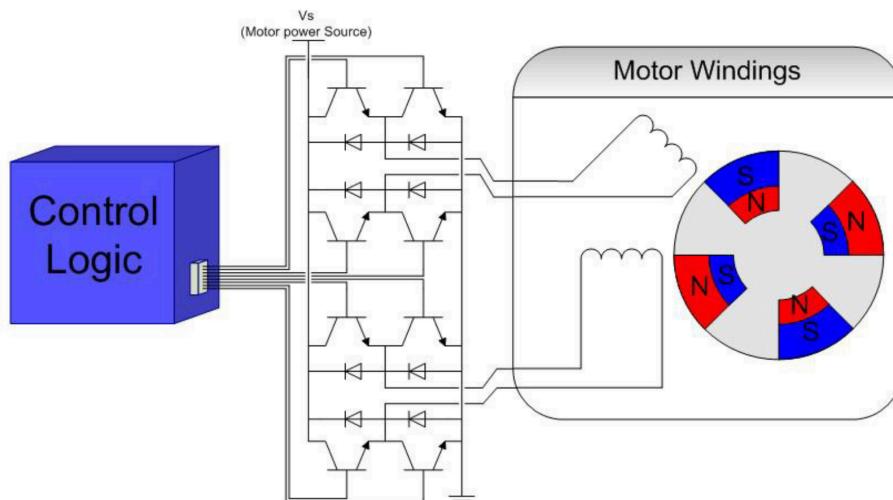
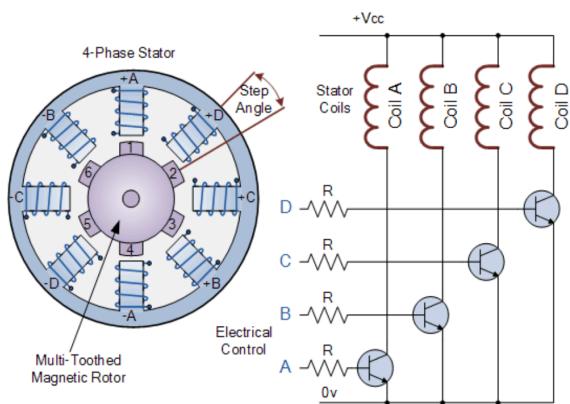
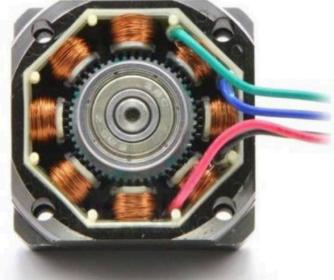
15.1 UART vs SSI

UART	SSI
Asynchronous protocol	Synchronous protocol
Two devices operate at the same frequency (baud rate) but have 2 separate clocks.	Two devices operate with the same clock with one clock frequency.
Two clocks must have frequency within 5% of each other	The master creates the clock and the slave uses the clock to latch the data, either for input or output.
Lines: Tx, Rx and Gnd	Lines: Tx, Rx, Clk, Fss (optional)

16 Stepper motors

16.1 Working principle

- The rotating body is called the rotor.
- The outer body is called the stator.
- The rotor is made of a permanent magnet.
- The stator has a set of electromagnets.
- When one pair of electromagnets is on, the rotator will make one step motion order to align with the stator's magnetic field.
- By changing the sequence of energising the stator's electromagnet, we can get the rotator to make stepwise motions.



16.2 Advantages

- Easy to control as each pulse produces a respective step angle of the motor.
- Mounting space is small.
- Stepper motor will hold their position when stationary, which requires power.
- The stepper motor driver board is cheaper compared to other drivers like PWM or servo motor driver boards.

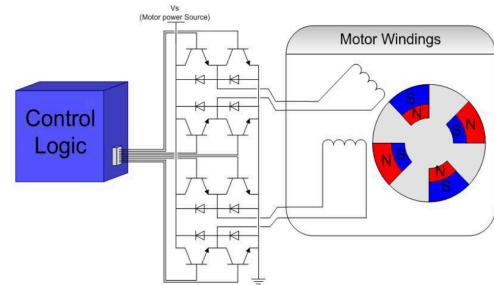
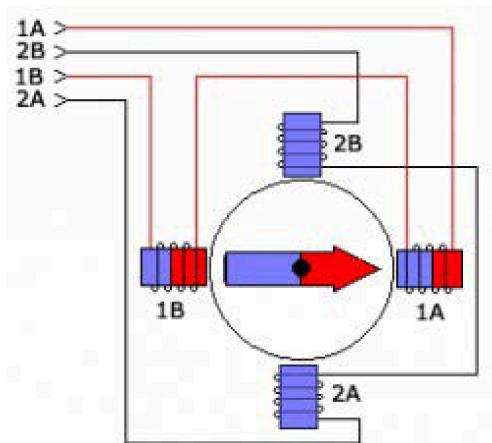
16.3 Disadvantages

- The design of the driver is not an efficient one.
- It requires a lot of wiring for the application.
- Larger stepper motors require heat sinks to dissipate heat from the stepper motor when they are stationary.

16.4 Applications

- Printers (2D and 3D)
- Robotic arms
- Autonomous vehicles

16.5 Output control



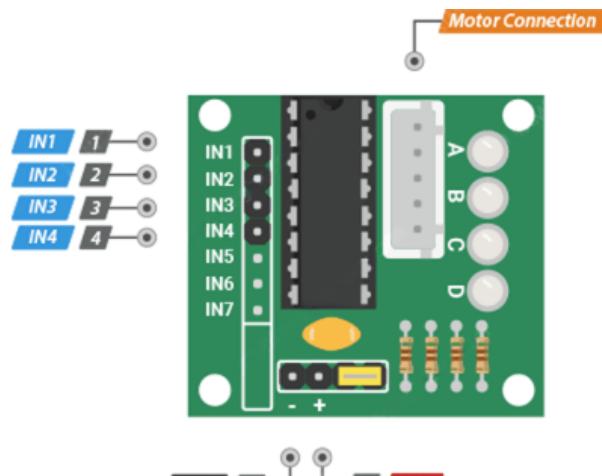
Phase 1:

1. 1A: Powered, 1B: Depowered
2. 1A: Depowered, 1B: Powered

Phase 2:

1. 2A: Powered, 2B: Depowered
2. 2A: Depowered, 2B: Powered

16.5.1 Example



ULN2003 Driver Pinout

Figure 3: ULN2003 stepper motor driver board

- IN1 - IN4 pins are used to drive the motor. Connect them to a digital output pins on the TM4C123G Launchpad (e.g. PC4, PC5, PC6, PC7).
- GND is a common ground pin.
- VDD pin supplies power for the motor. Connect it to an external 5 V power supply. Do not use the 5 V power from the TM4C123G Launchpad.
- The motor connector is where the stepper motor plus into. The connector is keyed, so it only goes one way.

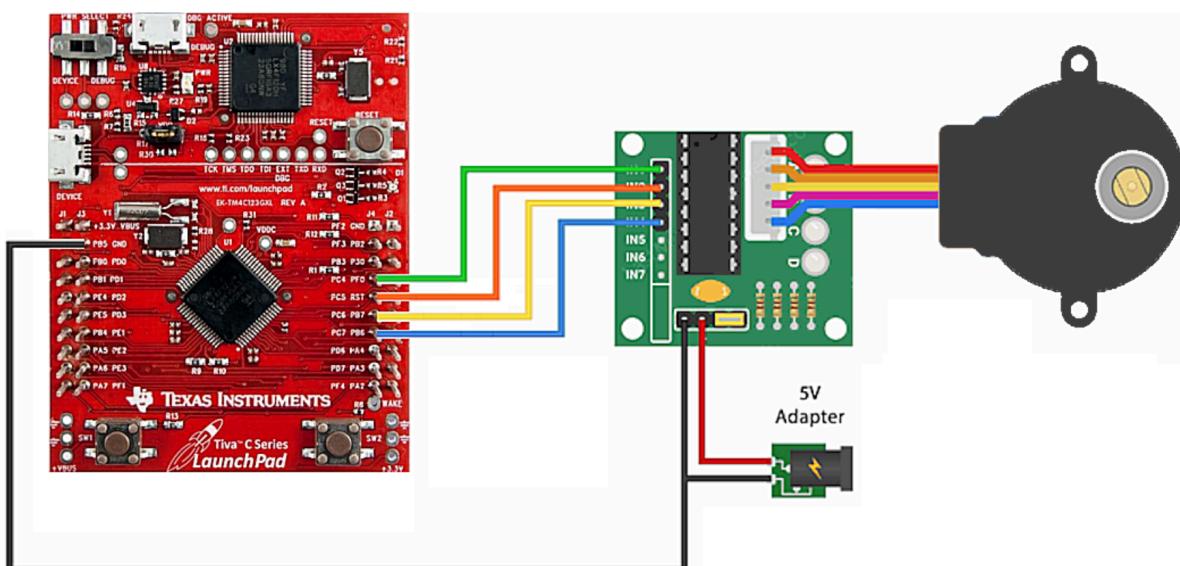
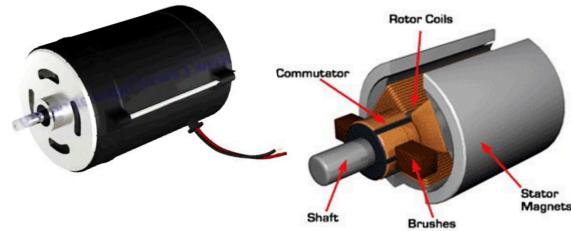
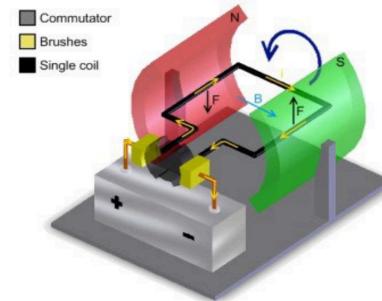


Figure 4: TM4C123G, ULN2003 stepper driver, and stepper motor connection diagram.

17 Brushed DC motors

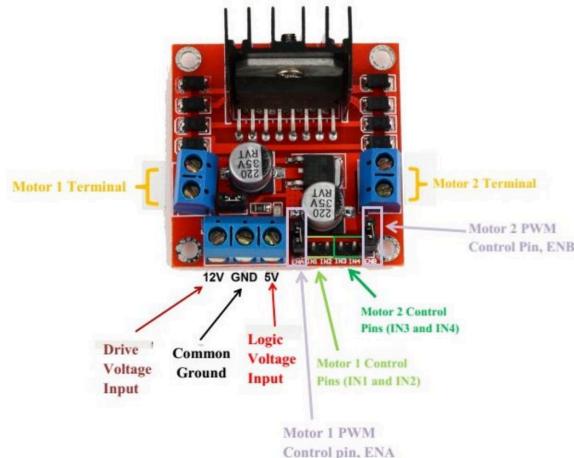
17.1 Working principle

- The stator is made of a permanent magnet.
- The rotor has a set of rectangular coils.
- When the rectangular coils are parallel to the magnetic field, current will pass through the coils. Also, the edges perpendicular to the magnetic flux will produce magnetic forces, which make the coils rotate in one direction.
- The motion of the rotor will bring another rectangular coil into being parallel with the magnetic field. The process repeats.



17.2 L289N H-bridge motor driver

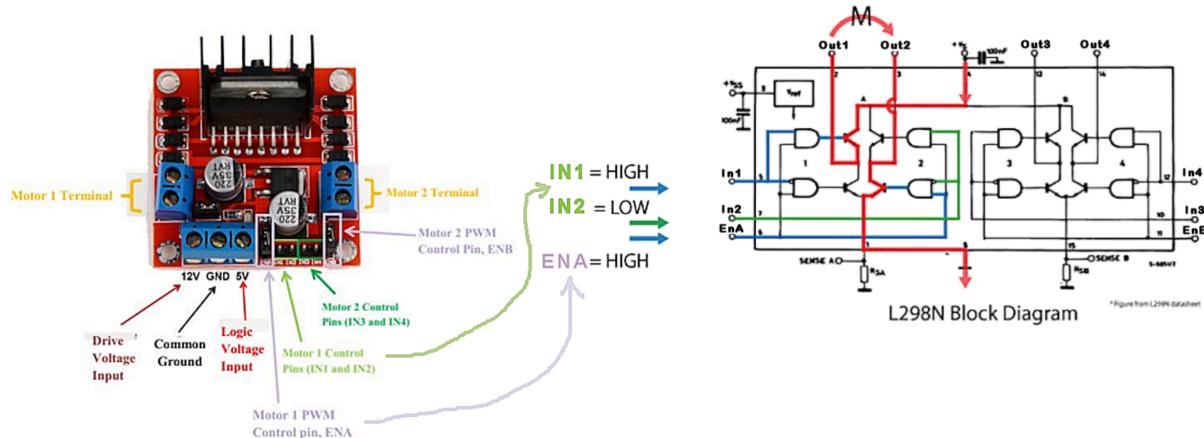
- There are two terminals with two connections. Hence, the driver can support 2 DC motors at a time.
- IN1 and IN2 are the direction control pins for the motor 1, which controls the first internal H-bridge within the driver.
- IN3 and IN4 are the direction control pins for the motor 2, which controls the second internal H-bridge within the driver.
- ENA and ENB controls motor 1 and motor 2 respectively. Keeping these pins on logical high will let the DC motors rotate. Pulling these pins to a low state will turn off the motors.
- By removing jumpers, we can apply a PWM signal to each pin instead of just an on/off signal.



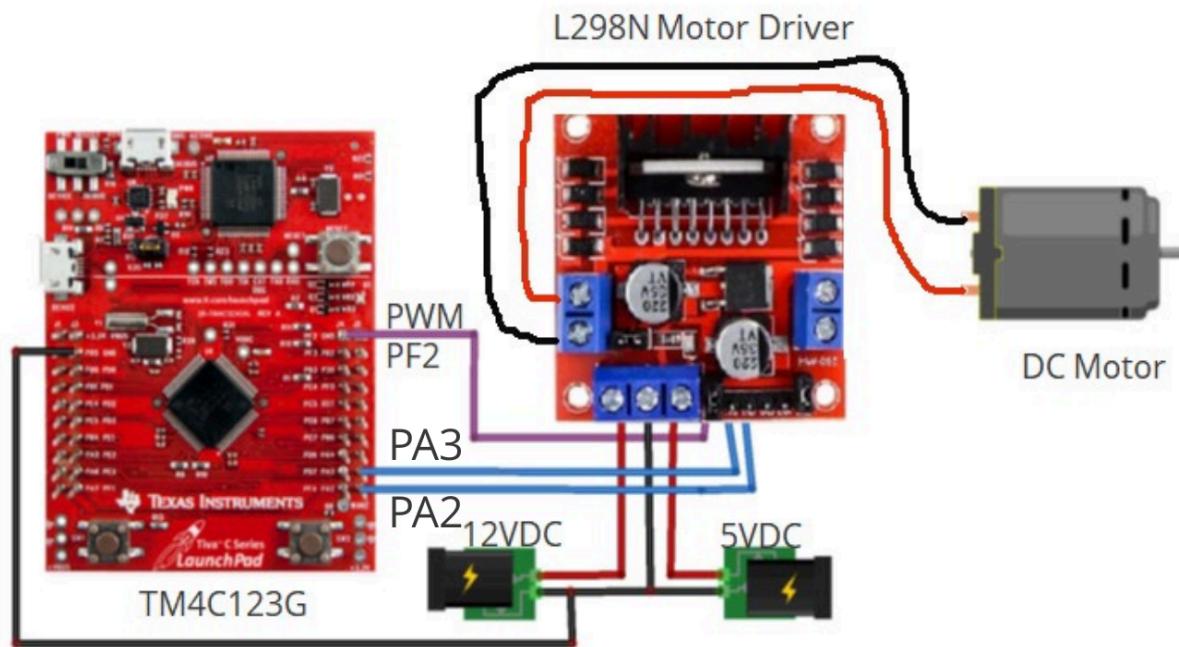
17.2.1 Output control

17.2.1.1 PWM logic control

- Clockwise (CW) rotation:
 - ▶ IN1 high.
 - ▶ IN2 low.
 - ▶ ENA supplies PWM pulses.
- Counter-clockwise (CCW) rotation:
 - ▶ IN1 low.
 - ▶ IN2 high.
 - ▶ ENA supplies PWM pulses.

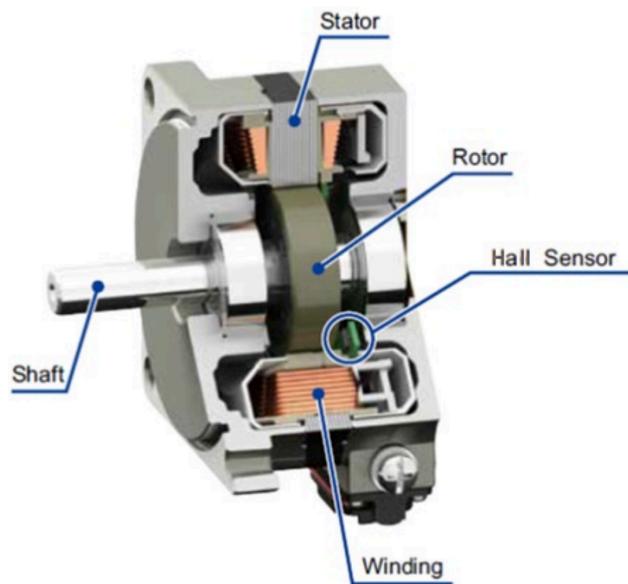


17.2.1.2 PWM power control



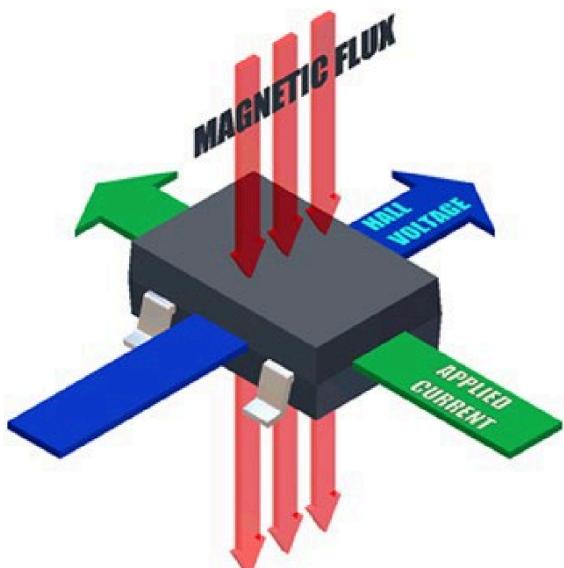
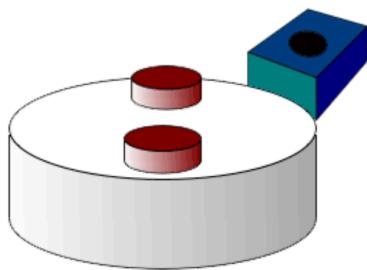
- To drive a DC motor using a TM4C123G with a L298N motor driver, three signals to the L298N motor driver from the TM4C123G are required.
- Two signals to control the direction (e.g. GPIO PA2 and PA3) to provide active high/low to the L298N and one PWM signal (PF2).

18 Brushless DC motors

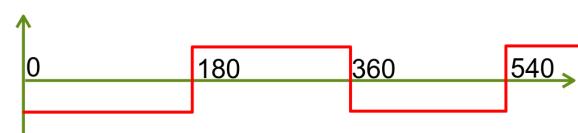


18.1 Hall effect sensor

When a magnetic flux passes through the sensor, a voltage is produced as output.



Hall Effect Sensor



18.2 Working principle

- The rotor is made of a permanent magnet.
- The stator has a set of electromagnets.
- The stator's electromagnets are sequentially excited to create a rotating magnetic field.
- The stator's rotating magnetic field will cause the rotator to follow the rotation.
- The hall sensors are employed to provide the position feedback so as to synchronise the commutation of excitations.

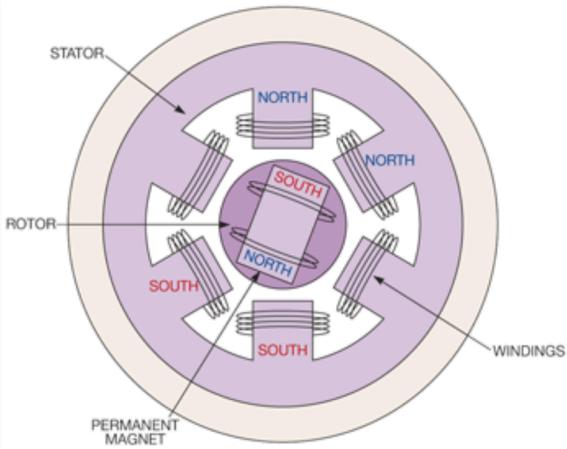
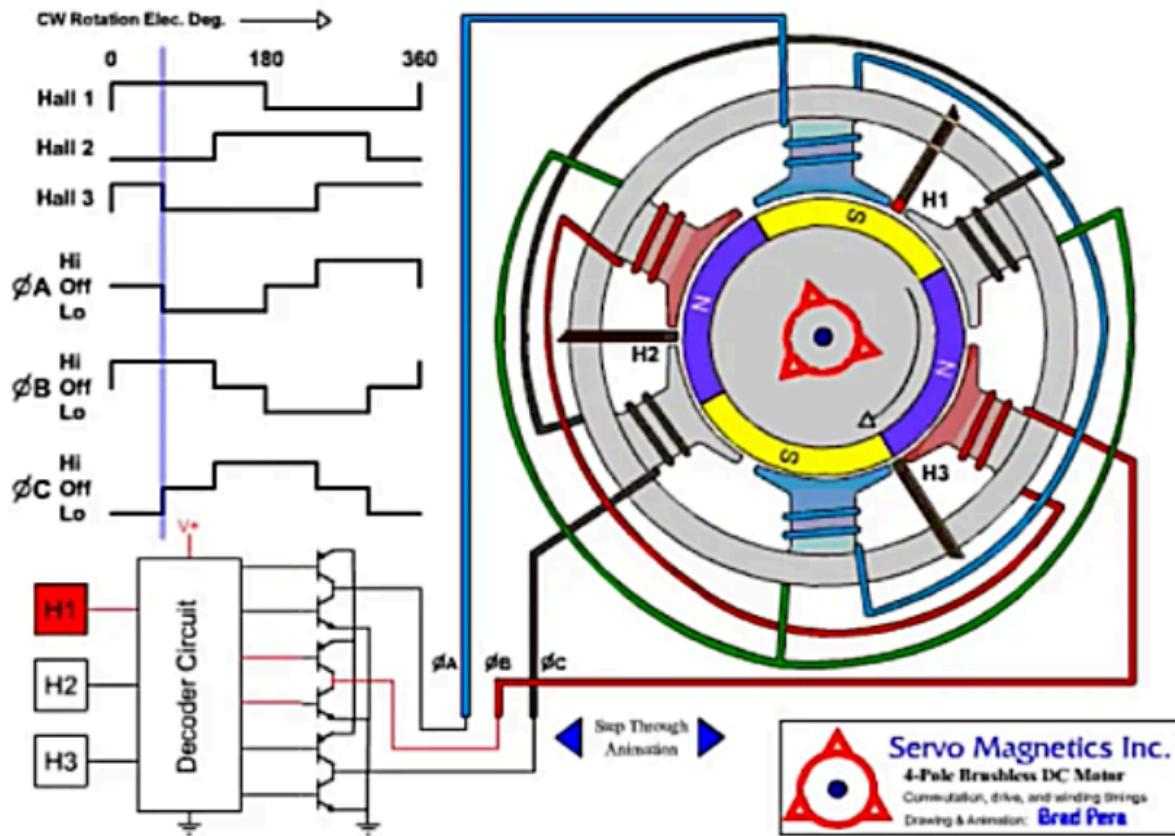
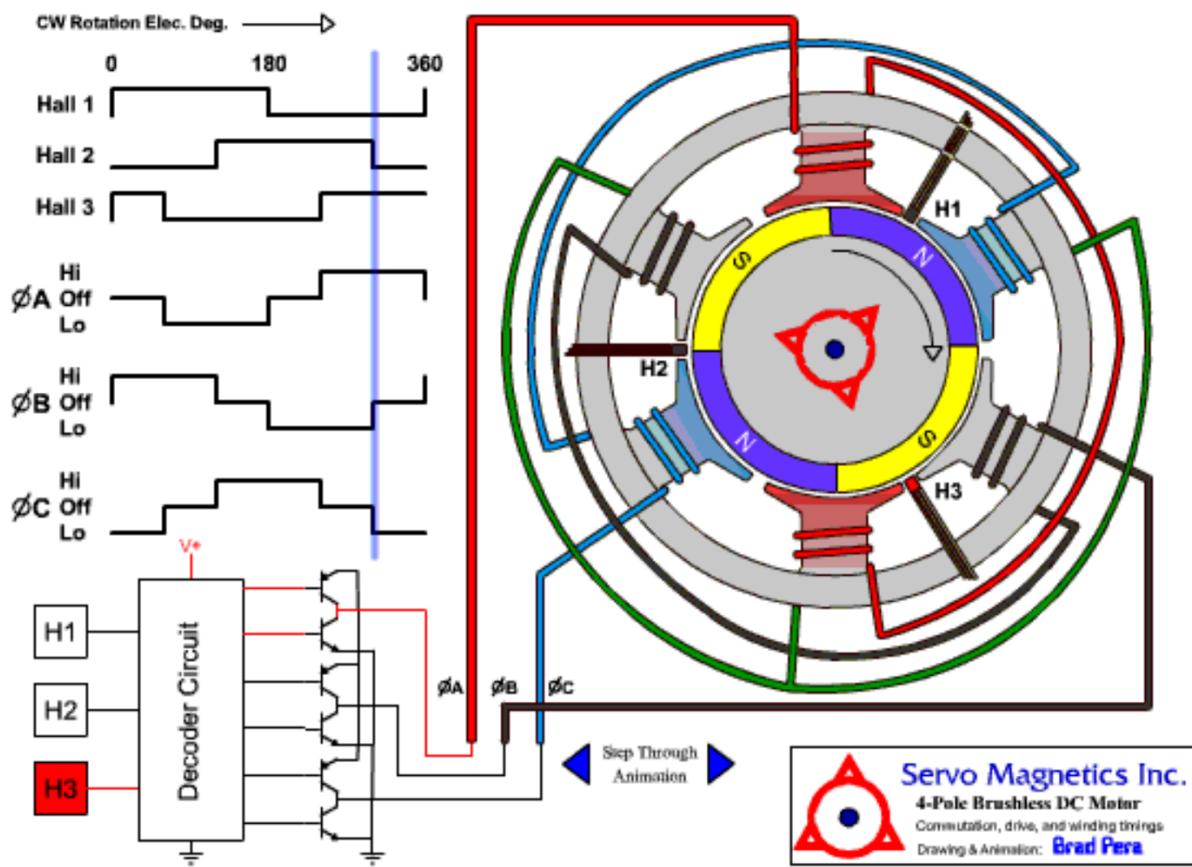
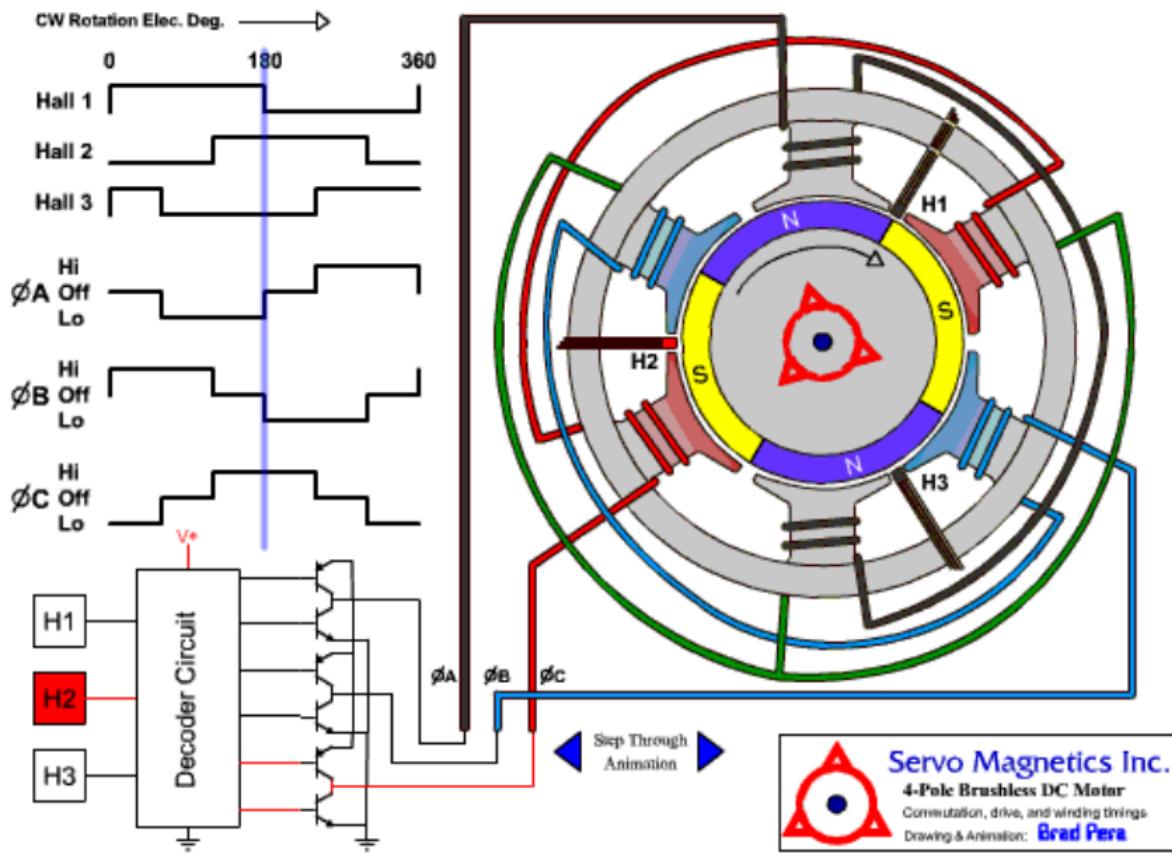


Figure 5: Brushless DC motors have a rotor with a permanent magnet containing a north and south poles. The stator comprises multiple electromagnets.

18.3 Sequence





18.4 Advantages

Brushless DC motors are more efficient as its velocity is determined by the frequency at which the current is supplied, not the voltage.

1. As brushes are absent, the mechanical energy loss due to friction is less, which enhances efficiency.
2. Brushless DC motors can operate at high-speeds under any condition.
3. There is no sparking and there is much less noise during operation.
4. More electromagnets could be used on the stator for more precise control.
5. Brushless DC motors accelerate and decelerate easily as they have low rotor inertia.
6. They are also high performance motors that provide a large torque per cubic inch over a large speed range.
7. Brushless DC motors do not have brushes, which make them more reliable, last longer, and require very little maintenance.
8. There are no ionising sparks from the commutator, and electromagnetic interference is also reduced.
9. Such motors could be cooled by conduction, and no air flow is required to cool the insides of the motor.

18.5 Disadvantages

1. Brushless DC motors cost more than a brushed DC motor.
2. The power of the motor is limited, as too much heat weakens the magnets and may damage the insulation of the winding.

19 TM4C123GH6PM General Purpose Input/Output (GPIO)

19.1 Key features

- Six ports: ports A, B, C, D, E and F.
- Ports A to F are accessed through Advanced Peripheral Bus (APB).
- 43 programmable pins: 7:0 PA - PD, 5:0 PE, 4:0 PF
- Pins configured as digital inputs are Schmitt-triggered.
- Programmable control of GPIO's interrupts.
- Programmable control of GPIO's pad configuration.

19.2 Process of operating GPIO

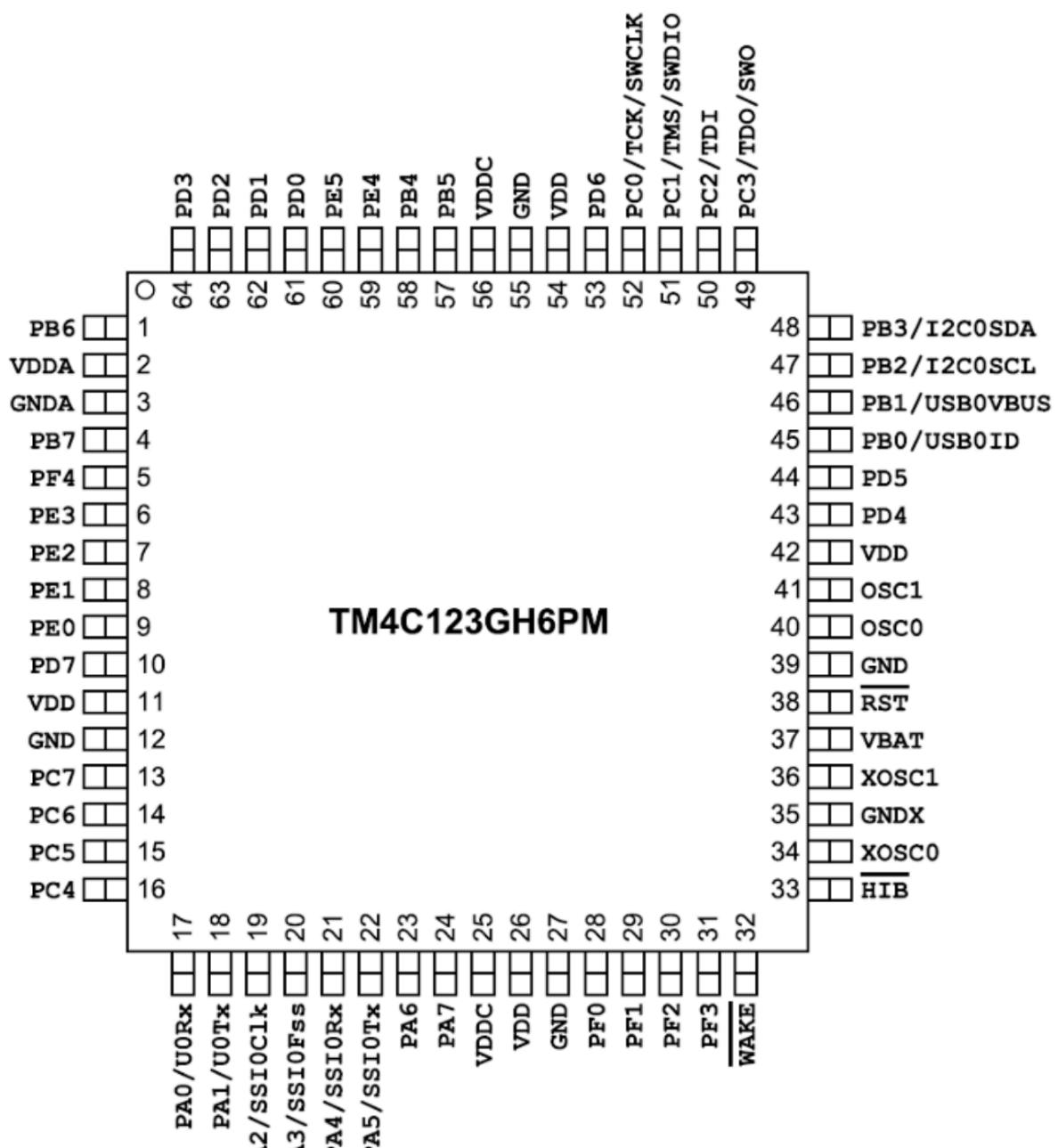
Control registers are used to:

- Configure input
- Configure internal processes

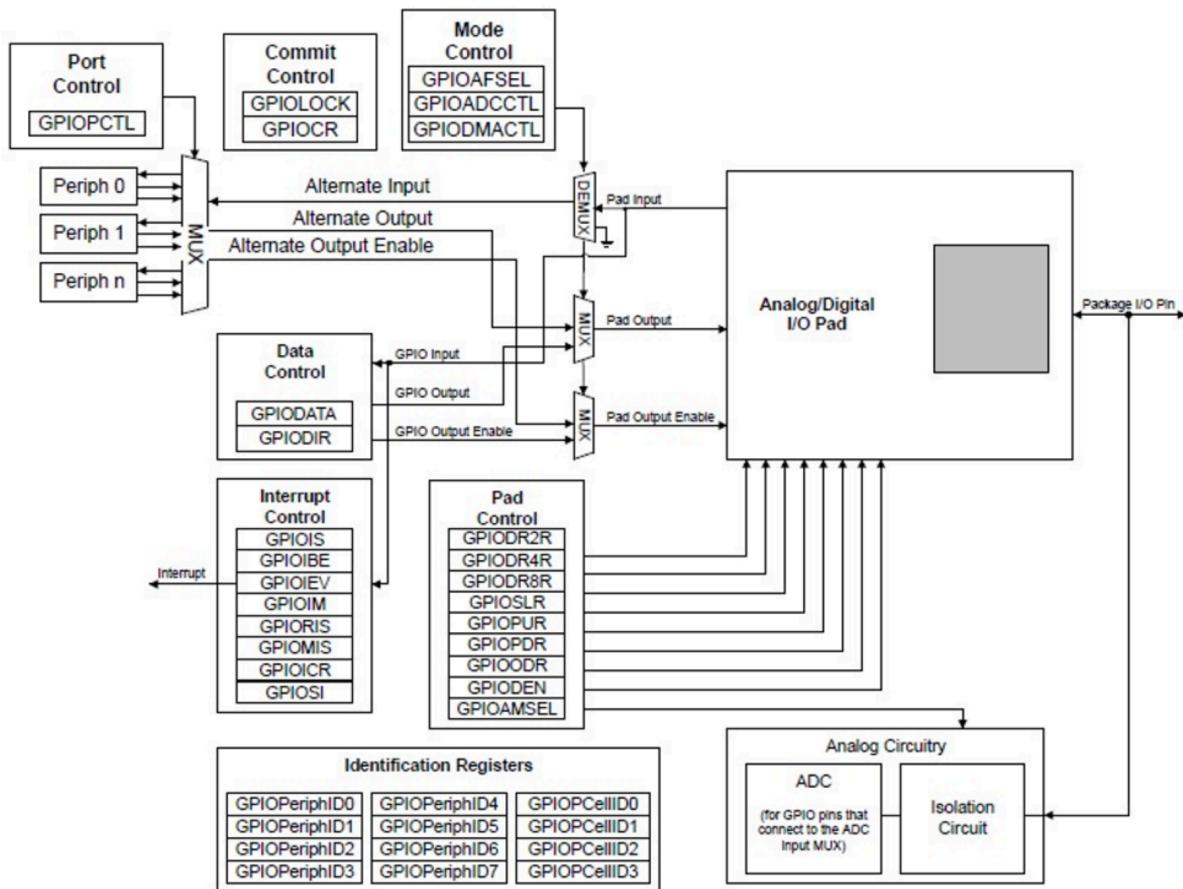
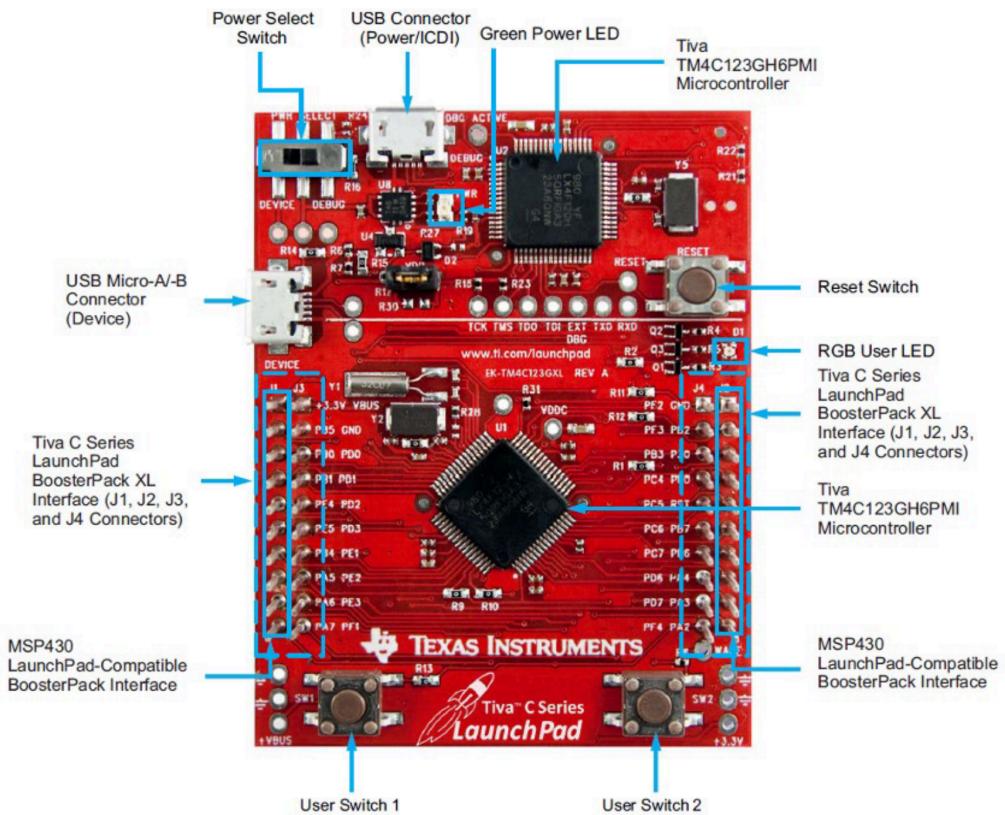
Status registers are used to:

- Configure output
- Run GPIO

19.3 Pin diagram



19.4 Analogue and digital I/O pads



19.5 GPIO pins and alternate functions

IO	Pin	Analog Function	Digital Function (GPIO PCTL PMC × Bit Field Encoding)											
			1	2	3	4	5	6	7	8	9	14	15	
PA0	17	-	U0Rx	-	-	-	CAN1Rx	-	-	-	-	-	-	
PA1	18	-	U0Tx	-	-	-	CAN1Tx	-	-	-	-	-	-	
PA2	19	-	-	SSI0Clk	-	-	-	-	-	-	-	-	-	
PA3	20	-	-	SSI0Fss	-	-	-	-	-	-	-	-	-	
PA4	21	-	-	SSI0Rx	-	-	-	-	-	-	-	-	-	
PA5	22	-	-	SSI0Tx	-	-	-	-	-	-	-	-	-	
PA6	23	-	-	-	I2C1SCL	-	M1PWM2	-	-	-	-	-	-	
PA7	24	-	-	-	I2C1SDA	-	M1PWM3	-	-	-	-	-	-	
PB0	45	USB0ID	U1Rx	-	-	-	-	-	T2CCP0	-	-	-	-	
PB1	46	USB0VBUS	U1Tx	-	-	-	-	-	T2CCP1	-	-	-	-	
PB2	47	-	-	-	I2C0SCL	-	-	-	T3CCP0	-	-	-	-	
PB3	48	-	-	-	I2C0SDA	-	-	-	T3CCP1	-	-	-	-	
PB4	58	AIN10	-	SSI2Clk	-	M0PWM2	-	-	T1CCP0	CAN0Rx	-	-	-	
PB5	57	AIN11	-	SSI2CFss	-	M0PWM3	-	-	T1CCP1	CAN0Tx	-	-	-	
PB6	1	-	-	SSI2Rx	-	M0PWM0	-	-	T0CCP0	-	-	-	-	
PB7	4	-	-	SSI2Tx	-	M0PWM1	-	-	T0CCP1	-	-	-	-	
PC0	52	-	TCK SWCLK	-	-	-	-	-	T4CCP0	-	-	-	-	
PC1	51	-	TMS SWDIO	-	-	-	-	-	T4CCP1	-	-	-	-	
PC2	50	-	TDI	-	-	-	-	-	T5CCP0	-	-	-	-	
PC3	49	-	TDO SW0	-	-	-	-	-	T5CCP1	-	-	-	-	
PC4	16	C1-	U4Rx	U1Rx	-	M0PWM6	-	IDX1	WT0CCP0	U1RTS	-	-	-	
PC5	15	C1+	U4Tx	U1Tx	-	M0PWM7	-	PhA1	WT0CCP1	U1CTS	-	-	-	
PC6	14	C0+	U3Rx	-	-	-	-	PhB1	WT1CCP0	USB0EPEN	-	-	-	
PC7	13	C0-	U3Tx	-	-	-	-	-	WT1CCP1	USB0PFLT	-	-	-	
PD0	61	AIN7	SSI3Clk	SSI1Clk	I2C3SCL	M0PWM6	M1PWM0	-	WT2CCP0	-	-	-	-	
PD1	62	AIN6	SSI3Fss	SSI1Fss	I2C3SDA	M0PWM7	M1PWM1	-	WT2CCP1	-	-	-	-	
PD2	63	AIN5	SSI3Rx	SSI1Rx	-	M0FAULT0	-	-	WT3CCP0	USB0EPEN	-	-	-	
PD3	64	AIN4	SSI3Tx	SSI1Tx	-	-	-	IDX0	WT3CCP1	USB0PFLT	-	-	-	
PD4	43	USB0DM	U6Rx	-	-	-	-	-	WT4CCP0	-	-	-	-	
PD5	44	USB0DP	U6Tx	-	-	-	-	-	WT4CCP1	-	-	-	-	
PD6	53	-	U2Rx	-	-	M0FAULT0	-	PhA0	WT5CCP0	-	-	-	-	
PD7	10	-	U2Tx	-	-	-	-	PhB0	WT5CCP1	NMI	-	-	-	
PE0	9	AIN3	U7Rx	-	-	-	-	-	-	-	-	-	-	
PE1	8	AIN2	U7Tx	-	-	-	-	-	-	-	-	-	-	
PE2	7	AIN1	-	-	-	-	-	-	-	-	-	-	-	
PE3	6	AIN0	-	-	-	-	-	-	-	-	-	-	-	

IO	Pin	Analog Function	Digital Function (GPIO PCTL PMC × Bit Field Encoding)											
			1	2	3	4	5	6	7	8	9	14	15	
PE4	59	AIN9	U5Rx	-	I2CSCL	M0PWM4	M1PWM2	-	-	CAN0Rx	-	-	-	
PE5	60	AIN8	U5Tx	-	I2CSDA	M0PWM5	M1PWM3	-	-	CAN0Tx	-	-	-	
PF0	28	-	U1RTS	SSI1Rx	CAN0RX	-	M1PWM4	PhA0	T0CCP0	NMI	C0o	-	-	
PF1	29	-	U1CTS	SSI1Tx	-	-	M1PWM5	PhB0	T0CCP1	-	C1o	TRD1	-	
PF2	30	-	-	SSI1Clk	-	M0FAULT0	M1PWM6	-	T1CCP0	-	-	TRD0	-	
PF3	31	-	-	SSI1Fss	CAN0Tx	-	M1PWM7	-	T1CCP1	-	-	TRDCLK	-	
PF4	5	-	-	-	-	-	M1FAULT0	IDX0	T2CCP0	USB0EPEN	-	-	-	

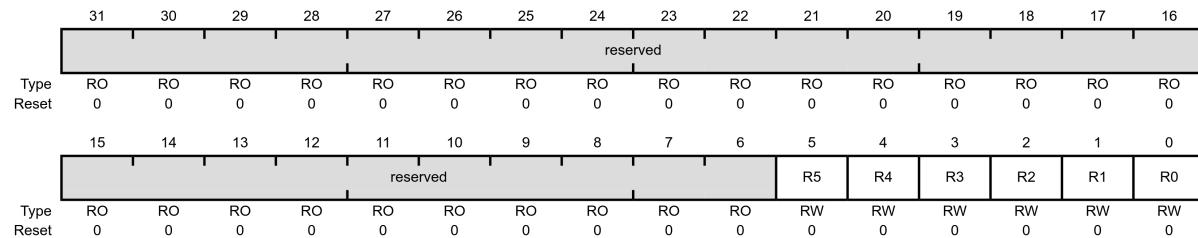
19.6 GPIO register map

Offset	Name	Type	Reset	Description
0x000	GPIODATA	RW	0x0000.0000	GPIO Data
0x400	GPIODIR	RW	0x0000.0000	GPIO Direction
0x404	GPIOIS	RW	0x0000.0000	GPIO Interrupt Sense
0x408	GPIOIBE	RW	0x0000.0000	GPIO Interrupt Both Edges
0x40C	GPIOIEV	RW	0x0000.0000	GPIO Interrupt Event
0x410	GPIOIM	RW	0x0000.0000	GPIO Interrupt Mask
0x414	GPIOIRIS	RO	0x0000.0000	GPIO Raw Interrupt Status
0x418	GPIOIMIS	RO	0x0000.0000	GPIO Masked Interrupt Status
0x41C	GPIOICR	W1C	0x0000.0000	GPIO Interrupt Clear
0x420	GPIOAFSEL	RW	-	GPIO Alternate Function Select
0x500	GPIODR2R	RW	0x0000.00FF	GPIO 2-mA Drive Select
0x504	GPIODR4R	RW	0x0000.0000	GPIO 4-mA Drive Select
0x508	GPIODR8R	RW	0x0000.0000	GPIO 8-mA Drive Select
0x50C	GPIOODR	RW	0x0000.0000	GPIO Open Drain Select
0x510	GPIOPUR	RW	-	GPIO Pull-Up Select
0x514	GPIOPDR	RW	0x0000.0000	GPIO Pull-Down Select
0x518	GPIOSLR	RW	0x0000.0000	GPIO Slew Rate Control Select
0x51C	GPIODEN	RW	-	GPIO Digital Enable
0x520	GPIOLOCK	RW	0x0000.0001	GPIO Lock
0x524	GPIOCR	-	-	GPIO Commit
0x528	GPIOAMSEL	RW	0x0000.0000	GPIO Analogue Mode Select
0x52C	GPIOPCTL	RW	-	GPIO Port Control
0x530	GPIOADCCTL	RW	0x0000.0000	GPIO ADC Control
0x534	GPIODMACTL	RW	0x0000.0000	GPIO DMA Control
0xFD0	GPIOPeriphID4	RO	0x0000.0000	GPIO Peripheral Identification 4
0xFD4	GPIOPeriphID5	RO	0x0000.0000	GPIO Peripheral Identification 5
0xFD8	GPIOPeriphID6	RO	0x0000.0000	GPIO Peripheral Identification 6
0xFDC	GPIOPeriphID7	RO	0x0000.0000	GPIO Peripheral Identification 7
0xFE0	GPIOPeriphID0	RO	0x0000.0061	GPIO Peripheral Identification 0
0xFE4	GPIOPeriphID1	RO	0x0000.0000	GPIO Peripheral Identification 1
0xFE8	GPIOPeriphID2	RO	0x0000.0018	GPIO Peripheral Identification 2
0xFEC	GPIOPeriphID3	RO	0x0000.0001	GPIO Peripheral Identification 3
0xFF0	GPIOPCellID0	RO	0x0000.000D	GPIO PrimeCell Identification 0
0xFF4	GPIOPCellID1	RO	0x0000.00F0	GPIO PrimeCell Identification 1
0xFF8	GPIOPCellID2	RO	0x0000.0005	GPIO PrimeCell Identification 2
0xFFC	GPIOPCellID3	RO	0x0000.00B1	GPIO PrimeCell Identification 3

19.7 Registers

19.7.1 General-Purpose Input/Output Run Mode Clock Gating Control (RCGCGPIO)

- Base: 0x400F.E000
- Offset: 0x608
- Type RW, reset 0x0000.0000

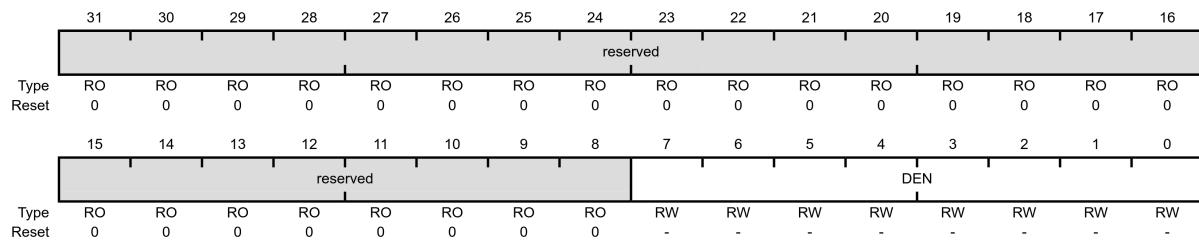


Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	R5	RW	0	GPIO Port F Run Mode Clock Gating Control Value Description 0 GPIO Port F is disabled. 1 Enable and provide a clock to GPIO Port F in Run mode.
4	R4	RW	0	GPIO Port E Run Mode Clock Gating Control Value Description 0 GPIO Port E is disabled. 1 Enable and provide a clock to GPIO Port E in Run mode.
3	R3	RW	0	GPIO Port D Run Mode Clock Gating Control Value Description 0 GPIO Port D is disabled. 1 Enable and provide a clock to GPIO Port D in Run mode.
2	R2	RW	0	GPIO Port C Run Mode Clock Gating Control Value Description 0 GPIO Port C is disabled. 1 Enable and provide a clock to GPIO Port C in Run mode.

Bit/Field	Name	Type	Reset	Description						
1	R1	RW	0	<p>GPIO Port B Run Mode Clock Gating Control</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>GPIO Port B is disabled.</td> </tr> <tr> <td>1</td> <td>Enable and provide a clock to GPIO Port B in Run mode.</td> </tr> </tbody> </table>	Value	Description	0	GPIO Port B is disabled.	1	Enable and provide a clock to GPIO Port B in Run mode.
Value	Description									
0	GPIO Port B is disabled.									
1	Enable and provide a clock to GPIO Port B in Run mode.									
0	R0	RW	0	<p>GPIO Port A Run Mode Clock Gating Control</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>GPIO Port A is disabled.</td> </tr> <tr> <td>1</td> <td>Enable and provide a clock to GPIO Port A in Run mode.</td> </tr> </tbody> </table>	Value	Description	0	GPIO Port A is disabled.	1	Enable and provide a clock to GPIO Port A in Run mode.
Value	Description									
0	GPIO Port A is disabled.									
1	Enable and provide a clock to GPIO Port A in Run mode.									

19.7.2 GPIO Digital Enable (GPIODEN)

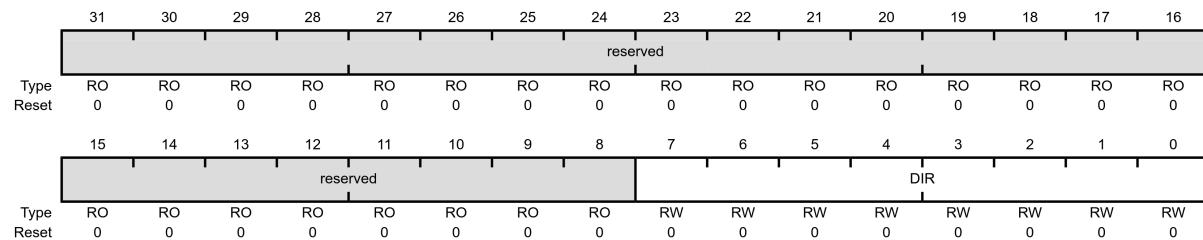
- GPIO Port A (APB) base: 0x4000.4000
- GPIO Port A (AHB) base: 0x4005.8000
- GPIO Port B (APB) base: 0x4000.5000
- GPIO Port B (AHB) base: 0x4005.9000
- GPIO Port C (APB) base: 0x4000.6000
- GPIO Port C (AHB) base: 0x4005.A000
- GPIO Port D (APB) base: 0x4000.7000
- GPIO Port D (AHB) base: 0x4005.B000
- GPIO Port E (APB) base: 0x4002.4000
- GPIO Port E (AHB) base: 0x4005.C000
- GPIO Port F (APB) base: 0x4002.5000
- GPIO Port F (AHB) base: 0x4005.D000
- Offset 0x51C
- Type RW, reset -



Bit/Field	Name	Type	Reset	Description								
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.								
7:0	DEN	RW	-	<p>Digital Enable</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The digital functions for the corresponding pin are disabled.</td> </tr> <tr> <td>1</td> <td>The digital functions for the corresponding pin are enabled.</td> </tr> <tr> <td>0x0000.0000</td> <td>The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in page 147.</td> </tr> </tbody> </table>	Value	Description	0	The digital functions for the corresponding pin are disabled.	1	The digital functions for the corresponding pin are enabled.	0x0000.0000	The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in page 147 .
Value	Description											
0	The digital functions for the corresponding pin are disabled.											
1	The digital functions for the corresponding pin are enabled.											
0x0000.0000	The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in page 147 .											

19.7.3 GPIO Direction (GPIODIR)

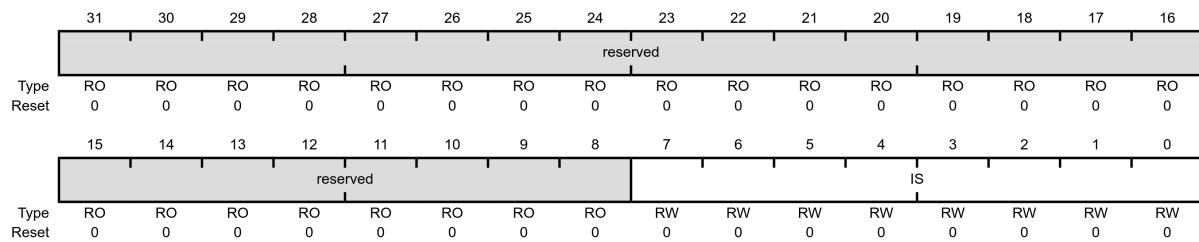
- GPIO Port A (APB) base: 0x4000.4000
- GPIO Port A (AHB) base: 0x4005.8000
- GPIO Port B (APB) base: 0x4000.5000
- GPIO Port B (AHB) base: 0x4005.9000
- GPIO Port C (APB) base: 0x4000.6000
- GPIO Port C (AHB) base: 0x4005.A000
- GPIO Port D (APB) base: 0x4000.7000
- GPIO Port D (AHB) base: 0x4005.B000
- GPIO Port E (APB) base: 0x4002.4000
- GPIO Port E (AHB) base: 0x4005.C000
- GPIO Port F (APB) base: 0x4002.5000
- GPIO Port F (AHB) base: 0x4005.D000
- Offset 0x400
- Type RW, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description						
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
7:0	DIR	RW	0x00	<p>GPIO Data Direction</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Corresponding pin is an input.</td> </tr> <tr> <td>1</td> <td>Corresponding pin is an output.</td> </tr> </tbody> </table>	Value	Description	0	Corresponding pin is an input.	1	Corresponding pin is an output.
Value	Description									
0	Corresponding pin is an input.									
1	Corresponding pin is an output.									

19.7.4 GPIO Interrupt Sense (GPIOIS)

- GPIO Port A (APB) base: 0x4000.4000
- GPIO Port A (AHB) base: 0x4005.8000
- GPIO Port B (APB) base: 0x4000.5000
- GPIO Port B (AHB) base: 0x4005.9000
- GPIO Port C (APB) base: 0x4000.6000
- GPIO Port C (AHB) base: 0x4005.A000
- GPIO Port D (APB) base: 0x4000.7000
- GPIO Port D (AHB) base: 0x4005.B000
- GPIO Port E (APB) base: 0x4002.4000
- GPIO Port E (AHB) base: 0x4005.C000
- GPIO Port F (APB) base: 0x4002.5000
- GPIO Port F (AHB) base: 0x4005.D000
- Offset 0x404
- Type RW, reset 0x0000.0000

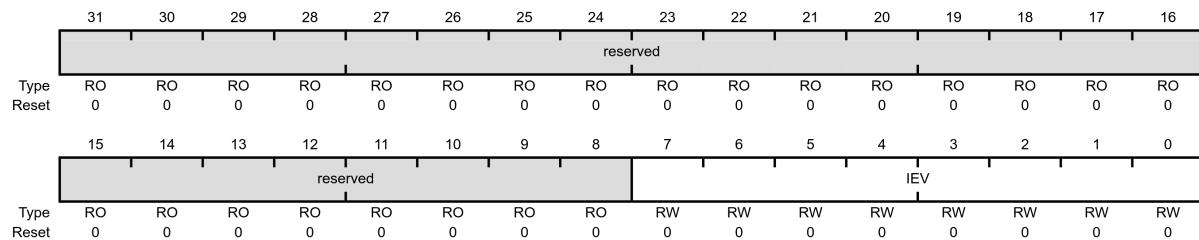


Bit/Field	Name	Type	Reset	Description						
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
7:0	IS	RW	0x00	<p>GPIO Interrupt Sense</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The edge on the corresponding pin is detected (edge-sensitive).</td> </tr> <tr> <td>1</td> <td>The level on the corresponding pin is detected (level-sensitive).</td> </tr> </tbody> </table>	Value	Description	0	The edge on the corresponding pin is detected (edge-sensitive).	1	The level on the corresponding pin is detected (level-sensitive).
Value	Description									
0	The edge on the corresponding pin is detected (edge-sensitive).									
1	The level on the corresponding pin is detected (level-sensitive).									

GPIOIS ([page 137](#)) and GPIOIEV ([page 138](#)) must be used together.

19.7.5 GPIO Interrupt Event (GPIOIEV)

- GPIO Port A (APB) base: 0x4000.4000
- GPIO Port A (AHB) base: 0x4005.8000
- GPIO Port B (APB) base: 0x4000.5000
- GPIO Port B (AHB) base: 0x4005.9000
- GPIO Port C (APB) base: 0x4000.6000
- GPIO Port C (AHB) base: 0x4005.A000
- GPIO Port D (APB) base: 0x4000.7000
- GPIO Port D (AHB) base: 0x4005.B000
- GPIO Port E (APB) base: 0x4002.4000
- GPIO Port E (AHB) base: 0x4005.C000
- GPIO Port F (APB) base: 0x4002.5000
- GPIO Port F (AHB) base: 0x4005.D000
- Offset 0x40C
- Type RW, reset 0x0000.0000

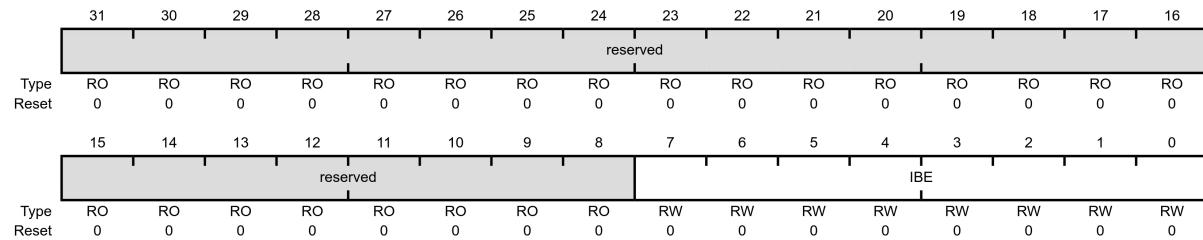


Bit/Field	Name	Type	Reset	Description						
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
7:0	IEV	RW	0x00	<p>GPIO Interrupt Event</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>A falling edge or a Low level on the corresponding pin triggers an interrupt.</td> </tr> <tr> <td>1</td> <td>A rising edge or a High level on the corresponding pin triggers an interrupt.</td> </tr> </tbody> </table>	Value	Description	0	A falling edge or a Low level on the corresponding pin triggers an interrupt.	1	A rising edge or a High level on the corresponding pin triggers an interrupt.
Value	Description									
0	A falling edge or a Low level on the corresponding pin triggers an interrupt.									
1	A rising edge or a High level on the corresponding pin triggers an interrupt.									

GPIOIS ([page 137](#)) and GPIOIEV ([page 138](#)) must be used together.

19.7.6 GPIO Interrupt Both Edges (GPIOIBE)

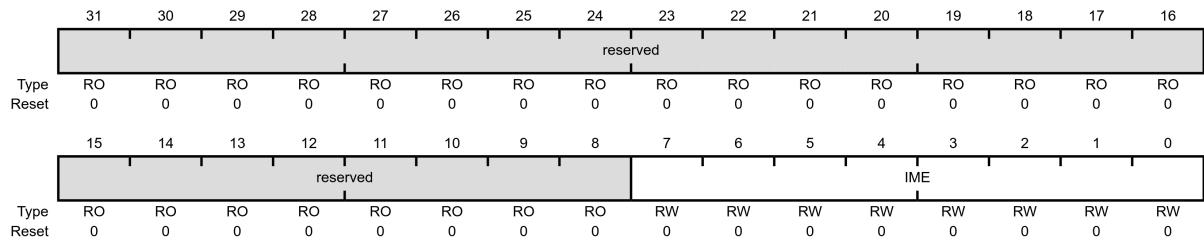
- GPIO Port A (APB) base: 0x4000.4000
- GPIO Port A (AHB) base: 0x4005.8000
- GPIO Port B (APB) base: 0x4000.5000
- GPIO Port B (AHB) base: 0x4005.9000
- GPIO Port C (APB) base: 0x4000.6000
- GPIO Port C (AHB) base: 0x4005.A000
- GPIO Port D (APB) base: 0x4000.7000
- GPIO Port D (AHB) base: 0x4005.B000
- GPIO Port E (APB) base: 0x4002.4000
- GPIO Port E (AHB) base: 0x4005.C000
- GPIO Port F (APB) base: 0x4002.5000
- GPIO Port F (AHB) base: 0x4005.D000
- Offset 0x408
- Type RW, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description						
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
7:0	IBE	RW	0x00	<p>GPIO Interrupt Both Edges</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Interrupt generation is controlled by the GPIO Interrupt Event (GPIOIEV) (page 138) register.</td> </tr> <tr> <td>1</td> <td>Both edges on the corresponding pin trigger an interrupt.</td> </tr> </tbody> </table>	Value	Description	0	Interrupt generation is controlled by the GPIO Interrupt Event (GPIOIEV) (page 138) register.	1	Both edges on the corresponding pin trigger an interrupt.
Value	Description									
0	Interrupt generation is controlled by the GPIO Interrupt Event (GPIOIEV) (page 138) register.									
1	Both edges on the corresponding pin trigger an interrupt.									

19.7.7 GPIO Interrupt Mask (GPIOIM)

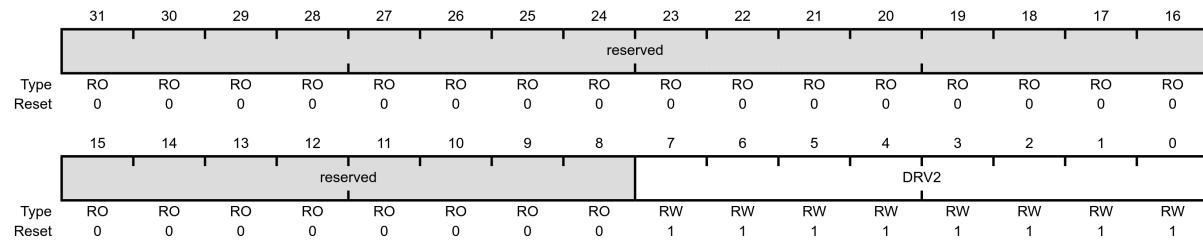
- GPIO Port A (APB) base: 0x4000.4000
 - GPIO Port A (AHB) base: 0x4005.8000
 - GPIO Port B (APB) base: 0x4000.5000
 - GPIO Port B (AHB) base: 0x4005.9000
 - GPIO Port C (APB) base: 0x4000.6000
 - GPIO Port C (AHB) base: 0x4005.A000
 - GPIO Port D (APB) base: 0x4000.7000
 - GPIO Port D (AHB) base: 0x4005.B000
 - GPIO Port E (APB) base: 0x4002.4000
 - GPIO Port E (AHB) base: 0x4005.C000
 - GPIO Port F (APB) base: 0x4002.5000
 - GPIO Port F (AHB) base: 0x4005.D000
 - Offset 0x410
 - Type RW, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description						
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
7:0	IME	RW	0x00	<p>GPIO Interrupt Mask Enable</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The interrupt from the corresponding pin is masked.</td> </tr> <tr> <td>1</td> <td>The interrupt from the corresponding pin is sent to the interrupt controller.</td> </tr> </tbody> </table>	Value	Description	0	The interrupt from the corresponding pin is masked.	1	The interrupt from the corresponding pin is sent to the interrupt controller.
Value	Description									
0	The interrupt from the corresponding pin is masked.									
1	The interrupt from the corresponding pin is sent to the interrupt controller.									

19.7.8 GPIO 2-mA Drive Select (GPIODR2R)

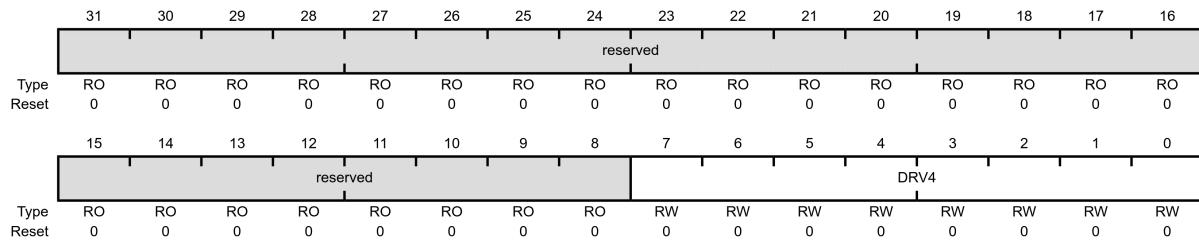
- GPIO Port A (APB) base: 0x4000.4000
- GPIO Port A (AHB) base: 0x4005.8000
- GPIO Port B (APB) base: 0x4000.5000
- GPIO Port B (AHB) base: 0x4005.9000
- GPIO Port C (APB) base: 0x4000.6000
- GPIO Port C (AHB) base: 0x4005.A000
- GPIO Port D (APB) base: 0x4000.7000
- GPIO Port D (AHB) base: 0x4005.B000
- GPIO Port E (APB) base: 0x4002.4000
- GPIO Port E (AHB) base: 0x4005.C000
- GPIO Port F (APB) base: 0x4002.5000
- GPIO Port F (AHB) base: 0x4005.D000
- Offset 0x500
- Type RW, reset 0x0000.00FF



Bit/Field	Name	Type	Reset	Description						
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
7:0	DRV2	RW	0xFF	<p>Output Pad 2-mA Drive Enable</p> <table> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>The drive for the corresponding GPIO pin is controlled by the GPIODR4R (page 142) or GPIODR8R (page 143) register.</td> </tr> <tr> <td>1</td> <td>The corresponding GPIO pin has 2-mA drive.</td> </tr> </table> <p>Setting a bit in either the GPIODR4R (page 142) register or the GPIODR8R (page 143) register clears the corresponding 2-mA enable bit. The change is effective on the second clock cycle after the write if accessing GPIO via the APB memory aperture. If using AHB access, the change is effective on the next clock cycle.</p>	Value	Description	0	The drive for the corresponding GPIO pin is controlled by the GPIODR4R (page 142) or GPIODR8R (page 143) register.	1	The corresponding GPIO pin has 2-mA drive.
Value	Description									
0	The drive for the corresponding GPIO pin is controlled by the GPIODR4R (page 142) or GPIODR8R (page 143) register.									
1	The corresponding GPIO pin has 2-mA drive.									

19.7.9 GPIO 4-mA Drive Select (GPIODR4R)

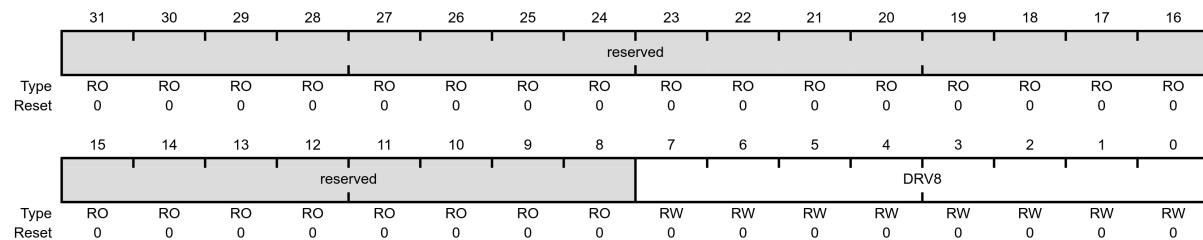
- GPIO Port A (APB) base: 0x4000.4000
- GPIO Port A (AHB) base: 0x4005.8000
- GPIO Port B (APB) base: 0x4000.5000
- GPIO Port B (AHB) base: 0x4005.9000
- GPIO Port C (APB) base: 0x4000.6000
- GPIO Port C (AHB) base: 0x4005.A000
- GPIO Port D (APB) base: 0x4000.7000
- GPIO Port D (AHB) base: 0x4005.B000
- GPIO Port E (APB) base: 0x4002.4000
- GPIO Port E (AHB) base: 0x4005.C000
- GPIO Port F (APB) base: 0x4002.5000
- GPIO Port F (AHB) base: 0x4005.D000
- Offset 0x504
- Type RW, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description						
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
7:0	DRV4	RW	0x00	<p>Output Pad 4-mA Drive Enable</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The drive for the corresponding GPIO pin is controlled by the GPIODR2R (page 141) or GPIODR8R (page 143) register.</td> </tr> <tr> <td>1</td> <td>The corresponding GPIO pin has 4-mA drive.</td> </tr> </tbody> </table> <p>Setting a bit in either the GPIODR2R (page 141) register or the GPIODR8R (page 143) register clears the corresponding 4-mA enable bit. The change is effective on the second clock cycle after the write if accessing GPIO via the APB memory aperture. If using AHB access, the change is effective on the next clock cycle.</p>	Value	Description	0	The drive for the corresponding GPIO pin is controlled by the GPIODR2R (page 141) or GPIODR8R (page 143) register.	1	The corresponding GPIO pin has 4-mA drive.
Value	Description									
0	The drive for the corresponding GPIO pin is controlled by the GPIODR2R (page 141) or GPIODR8R (page 143) register.									
1	The corresponding GPIO pin has 4-mA drive.									

19.7.10 GPIO 8-mA Drive Select (GPIODR8R)

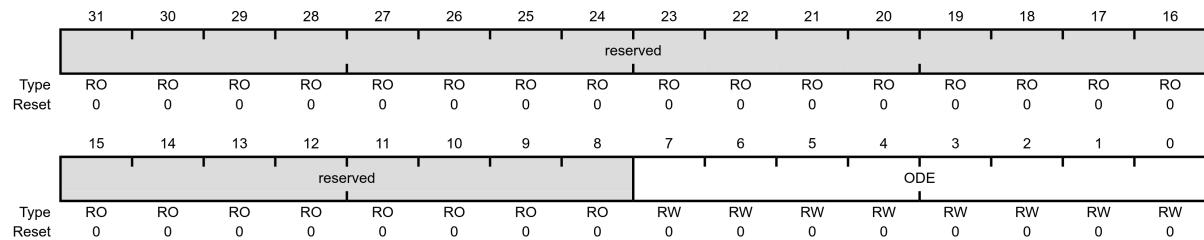
- GPIO Port A (APB) base: 0x4000.4000
- GPIO Port A (AHB) base: 0x4005.8000
- GPIO Port B (APB) base: 0x4000.5000
- GPIO Port B (AHB) base: 0x4005.9000
- GPIO Port C (APB) base: 0x4000.6000
- GPIO Port C (AHB) base: 0x4005.A000
- GPIO Port D (APB) base: 0x4000.7000
- GPIO Port D (AHB) base: 0x4005.B000
- GPIO Port E (APB) base: 0x4002.4000
- GPIO Port E (AHB) base: 0x4005.C000
- GPIO Port F (APB) base: 0x4002.5000
- GPIO Port F (AHB) base: 0x4005.D000
- Offset 0x508
- Type RW, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description						
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
7:0	DRV8	RW	0x00	<p>Output Pad 8-mA Drive Enable</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The drive for the corresponding GPIO pin is controlled by the GPIODR2R (page 141) or GPIODR4R (page 142) register.</td> </tr> <tr> <td>1</td> <td>The corresponding GPIO pin has 8-mA drive.</td> </tr> </tbody> </table> <p>Setting a bit in either the GPIODR2R (page 141) register or the GPIODR4R (page 142) register clears the corresponding 4-mA enable bit. The change is effective on the second clock cycle after the write if accessing GPIO via the APB memory aperture. If using AHB access, the change is effective on the next clock cycle.</p>	Value	Description	0	The drive for the corresponding GPIO pin is controlled by the GPIODR2R (page 141) or GPIODR4R (page 142) register.	1	The corresponding GPIO pin has 8-mA drive.
Value	Description									
0	The drive for the corresponding GPIO pin is controlled by the GPIODR2R (page 141) or GPIODR4R (page 142) register.									
1	The corresponding GPIO pin has 8-mA drive.									

19.7.11 GPIO Open Drain Select (GPIOODR)

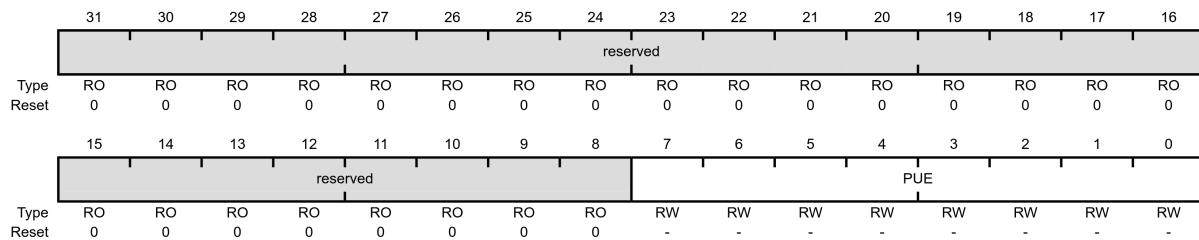
- GPIO Port A (APB) base: 0x4000.4000
- GPIO Port A (AHB) base: 0x4005.8000
- GPIO Port B (APB) base: 0x4000.5000
- GPIO Port B (AHB) base: 0x4005.9000
- GPIO Port C (APB) base: 0x4000.6000
- GPIO Port C (AHB) base: 0x4005.A000
- GPIO Port D (APB) base: 0x4000.7000
- GPIO Port D (AHB) base: 0x4005.B000
- GPIO Port E (APB) base: 0x4002.4000
- GPIO Port E (AHB) base: 0x4005.C000
- GPIO Port F (APB) base: 0x4002.5000
- GPIO Port F (AHB) base: 0x4005.D000
- Offset 0x50C
- Type RW, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description						
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
7:0	ODE	RW	0x00	<p>Output Pad Open Drain Enable</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The corresponding pin is not configured as open drain.</td> </tr> <tr> <td>1</td> <td>The corresponding pin is configured as open drain.</td> </tr> </tbody> </table>	Value	Description	0	The corresponding pin is not configured as open drain.	1	The corresponding pin is configured as open drain.
Value	Description									
0	The corresponding pin is not configured as open drain.									
1	The corresponding pin is configured as open drain.									

19.7.12 GPIO Pull-Up Select (GPIOPUR)

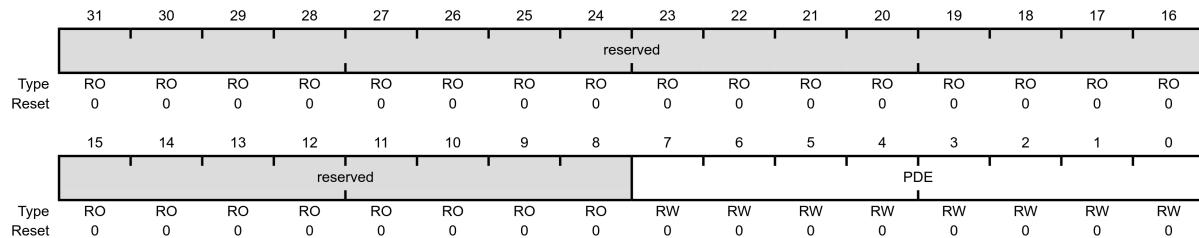
- GPIO Port A (APB) base: 0x4000.4000
- GPIO Port A (AHB) base: 0x4005.8000
- GPIO Port B (APB) base: 0x4000.5000
- GPIO Port B (AHB) base: 0x4005.9000
- GPIO Port C (APB) base: 0x4000.6000
- GPIO Port C (AHB) base: 0x4005.A000
- GPIO Port D (APB) base: 0x4000.7000
- GPIO Port D (AHB) base: 0x4005.B000
- GPIO Port E (APB) base: 0x4002.4000
- GPIO Port E (AHB) base: 0x4005.C000
- GPIO Port F (APB) base: 0x4002.5000
- GPIO Port F (AHB) base: 0x4005.D000
- Offset 0x510
- Type RW, reset -



Bit/Field	Name	Type	Reset	Description						
31:8	reserved	RO	0x0000.00	<p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p>						
7:0	PUE	RW	-	<p>Pad Weak Pull-Up Enable</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The corresponding pin's weak pull-up resistor is disabled.</td> </tr> <tr> <td>1</td> <td>The corresponding pin's weak pull-up resistor is enabled.</td> </tr> </tbody> </table> <p>Setting a bit in the GPIOPDR (page 146) register clears the corresponding bit in the GPIOPUR (page 145) register. The change is effective on the second clock cycle after the write if accessing GPIO via the APB memory aperture. If using AHB access, the change is effective on the next clock cycle.</p> <p>The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in page 147.</p>	Value	Description	0	The corresponding pin's weak pull-up resistor is disabled.	1	The corresponding pin's weak pull-up resistor is enabled.
Value	Description									
0	The corresponding pin's weak pull-up resistor is disabled.									
1	The corresponding pin's weak pull-up resistor is enabled.									

19.7.13 GPIO Pull-Down Select (GPIOOPDR)

- GPIO Port A (APB) base: 0x4000.4000
- GPIO Port A (AHB) base: 0x4005.8000
- GPIO Port B (APB) base: 0x4000.5000
- GPIO Port B (AHB) base: 0x4005.9000
- GPIO Port C (APB) base: 0x4000.6000
- GPIO Port C (AHB) base: 0x4005.A000
- GPIO Port D (APB) base: 0x4000.7000
- GPIO Port D (AHB) base: 0x4005.B000
- GPIO Port E (APB) base: 0x4002.4000
- GPIO Port E (AHB) base: 0x4005.C000
- GPIO Port F (APB) base: 0x4002.5000
- GPIO Port F (AHB) base: 0x4005.D000
- Offset 0x514
- Type RW, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description						
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
7:0	PDE	RW	0x00	<p>Pad Weak Pull-Down Enable</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The corresponding pin's weak pull-down resistor is disabled.</td> </tr> <tr> <td>1</td> <td>The corresponding pin's weak pull-down resistor is enabled.</td> </tr> </tbody> </table> <p>Setting a bit in the GPIOPUR (page 145) register clears the corresponding bit in the GPIOOPDR (page 146) register. The change is effective on the second clock cycle after the write if accessing GPIO via the APB memory aperture. If using AHB access, the change is effective on the next clock cycle.</p>	Value	Description	0	The corresponding pin's weak pull-down resistor is disabled.	1	The corresponding pin's weak pull-down resistor is enabled.
Value	Description									
0	The corresponding pin's weak pull-down resistor is disabled.									
1	The corresponding pin's weak pull-down resistor is enabled.									

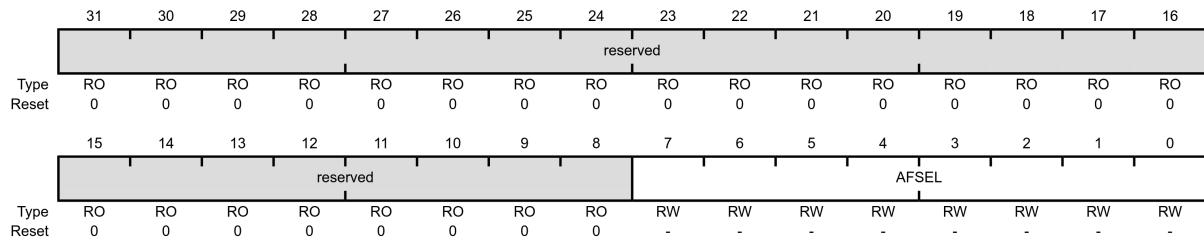
19.7.14 GPIO Pins with special considerations

GPIO Pins	Default Reset State	GPIOAFSEL	GPIODEN	GPIOPDR	GPIOPUR	GPIOPCTL	GPIOCR
PA[1:0]	UART0	0	0	0	0	0x1	1
PA[5:2]	SSI0	0	0	0	0	0x2	1
PB[3:2]	I ² C0	0	0	0	0	0x3	1
PC[3:0]	JTAG/SWD	1	1	0	1	0x1	0
PD[7]	GPIO ^a	0	0	0	0	0x0	0
PF[0]	GPIO ^a	0	0	0	0	0x0	0

a. This pin is configured as a GPIO by default but is locked and can only be reprogrammed by unlocking the pin in the **GPIOLOCK** (page 149) register and uncommitting it by setting the **GPIOCR** (page 150) register.

19.7.15 GPIO Alternate Function Select (GPIOAFSEL)

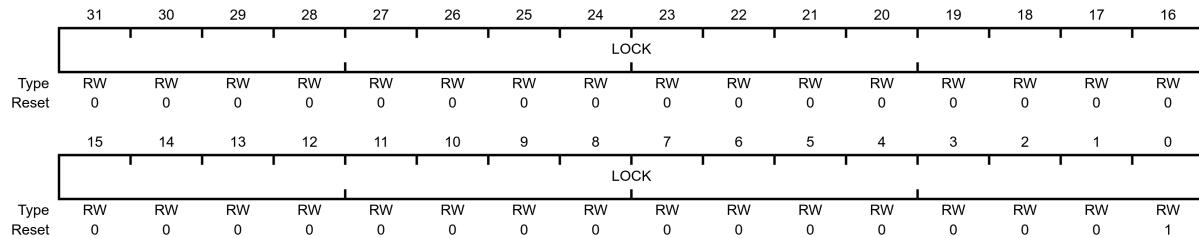
- GPIO Port A (APB) base: 0x4000.4000
- GPIO Port A (AHB) base: 0x4005.8000
- GPIO Port B (APB) base: 0x4000.5000
- GPIO Port B (AHB) base: 0x4005.9000
- GPIO Port C (APB) base: 0x4000.6000
- GPIO Port C (AHB) base: 0x4005.A000
- GPIO Port D (APB) base: 0x4000.7000
- GPIO Port D (AHB) base: 0x4005.B000
- GPIO Port E (APB) base: 0x4002.4000
- GPIO Port E (AHB) base: 0x4005.C000
- GPIO Port F (APB) base: 0x4002.5000
- GPIO Port F (AHB) base: 0x4005.D000
- Offset 0x420
- Type RW, reset -



Bit/Field	Name	Type	Reset	Description						
31:8	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
7:0	AFSEL	RW	-	<p>GPIO Alternate Function Select</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The associated pin functions as a GPIO and is controlled by the GPIO registers.</td> </tr> <tr> <td>1</td> <td>The associated pin functions as a peripheral signal and is controlled by the alternate hardware function.</td> </tr> </tbody> </table> <p>The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in page 147.</p>	Value	Description	0	The associated pin functions as a GPIO and is controlled by the GPIO registers.	1	The associated pin functions as a peripheral signal and is controlled by the alternate hardware function.
Value	Description									
0	The associated pin functions as a GPIO and is controlled by the GPIO registers.									
1	The associated pin functions as a peripheral signal and is controlled by the alternate hardware function.									

19.7.16 GPIO Lock (GPIOLOCK)

- GPIO Port A (APB) base: 0x4000.4000
- GPIO Port A (AHB) base: 0x4005.8000
- GPIO Port B (APB) base: 0x4000.5000
- GPIO Port B (AHB) base: 0x4005.9000
- GPIO Port C (APB) base: 0x4000.6000
- GPIO Port C (AHB) base: 0x4005.A000
- GPIO Port D (APB) base: 0x4000.7000
- GPIO Port D (AHB) base: 0x4005.B000
- GPIO Port E (APB) base: 0x4002.4000
- GPIO Port E (AHB) base: 0x4005.C000
- GPIO Port F (APB) base: 0x4002.5000
- GPIO Port F (AHB) base: 0x4005.D000
- Offset 0x520
- Type RW, reset 0x0000.0001

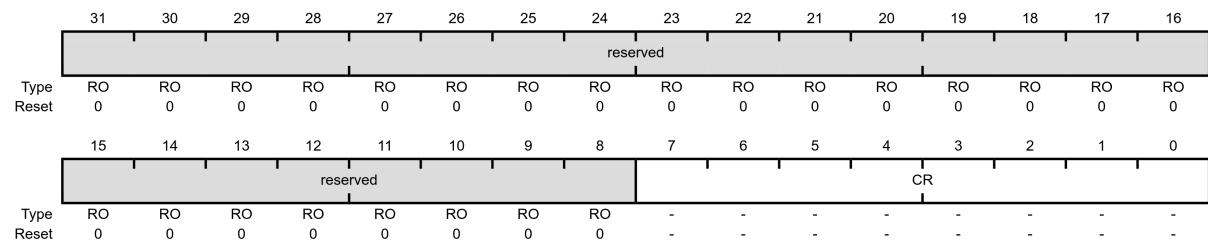


Bit/Field	Name	Type	Reset	Description						
31:0	LOCK	RW	0x0000.0001	<p>GPIO Lock</p> <p>A write of the value 0x4C4F.434B unlocks the GPIO Commit (GPIOCR) (page 150) register for write access. A write of any other value or a write to the GPIOCR (page 150) register reapplies the lock, preventing any register updates.</p> <p>A read of this register returns the following values:</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x1</td> <td>The GPIOCR (page 150) register is locked and may not be modified.</td> </tr> <tr> <td>0x0</td> <td>The GPIOCR (page 150) register is unlocked and may be modified.</td> </tr> </tbody> </table>	Value	Description	0x1	The GPIOCR (page 150) register is locked and may not be modified.	0x0	The GPIOCR (page 150) register is unlocked and may be modified.
Value	Description									
0x1	The GPIOCR (page 150) register is locked and may not be modified.									
0x0	The GPIOCR (page 150) register is unlocked and may be modified.									

19.7.17 GPIO Commit (GPIOCR)

The value of the **GPIOCR** (page 150) register determines which bits of the **GPIOAFSEL** (page 148), **GPIOPUR** (page 145), **GPIOPDR** (page 146), and **GPIODEN** (page 135) registers are committed when a write to these registers is performed.

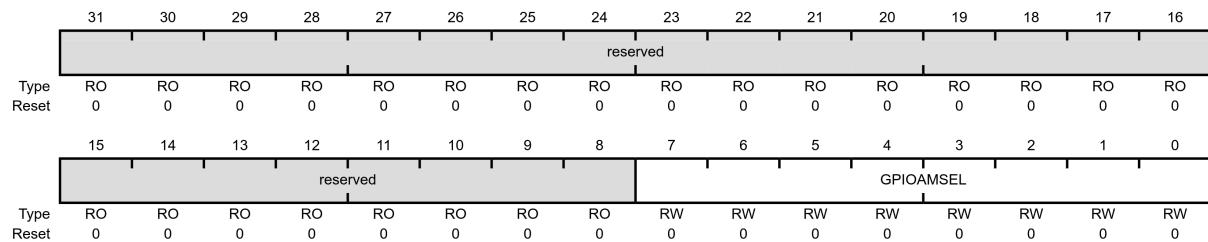
- GPIO Port A (APB) base: 0x4000.4000
- GPIO Port A (AHB) base: 0x4005.8000
- GPIO Port B (APB) base: 0x4000.5000
- GPIO Port B (AHB) base: 0x4005.9000
- GPIO Port C (APB) base: 0x4000.6000
- GPIO Port C (AHB) base: 0x4005.A000
- GPIO Port D (APB) base: 0x4000.7000
- GPIO Port D (AHB) base: 0x4005.B000
- GPIO Port E (APB) base: 0x4002.4000
- GPIO Port E (AHB) base: 0x4005.C000
- GPIO Port F (APB) base: 0x4002.5000
- GPIO Port F (AHB) base: 0x4005.D000
- Offset 0x524
- Type -, reset -



Bit/Field	Name	Type	Reset	Description						
7:0	CR	-	-	<p>GPIO Commit</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The corresponding GPIOAFSEL (page 148), GPIOPUR (page 145), GPIOPDR (page 145), or GPIODEN (page 145) bits cannot be written.</td> </tr> <tr> <td>1</td> <td>The corresponding GPIOAFSEL (page 145), GPIOPUR (page 145), GPIOPDR (page 145), or GPIODEN (page 145) bits can be written.</td> </tr> </tbody> </table> <p>Note:</p> <p>The default register type for the GPIOCR (page 150) register is RO for all GPIO pins with the exception of the NMI pin and the four JTAG/SWD pins. These six pins are the only GPIOs that are protected by the GPIOCR (page 150) register. Because of this, the register type for the corresponding GPIO Ports is RW.</p> <p>The default reset value for the GPIOCR (page 150) register is <code>0x0000.00FF</code> for all GPIO pins, with the exception of the NMI and JTAG/SWD pins . To ensure that the JTAG and NMI pins are not accidentally programmed as GPIO pins, these pins default to non-committable. Because of this, the default reset value of GPIOCR (page 150) changes for the corresponding ports.</p>	Value	Description	0	The corresponding GPIOAFSEL (page 148) , GPIOPUR (page 145) , GPIOPDR (page 145) , or GPIODEN (page 145) bits cannot be written.	1	The corresponding GPIOAFSEL (page 145) , GPIOPUR (page 145) , GPIOPDR (page 145) , or GPIODEN (page 145) bits can be written.
Value	Description									
0	The corresponding GPIOAFSEL (page 148) , GPIOPUR (page 145) , GPIOPDR (page 145) , or GPIODEN (page 145) bits cannot be written.									
1	The corresponding GPIOAFSEL (page 145) , GPIOPUR (page 145) , GPIOPDR (page 145) , or GPIODEN (page 145) bits can be written.									

19.7.18 GPIO Analog Mode Select (GPIOAMSEL)

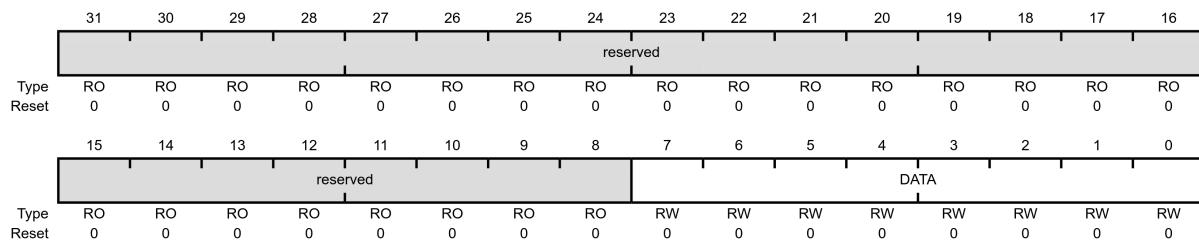
- GPIO Port A (APB) base: 0x4000.4000
- GPIO Port A (AHB) base: 0x4005.8000
- GPIO Port B (APB) base: 0x4000.5000
- GPIO Port B (AHB) base: 0x4005.9000
- GPIO Port C (APB) base: 0x4000.6000
- GPIO Port C (AHB) base: 0x4005.A000
- GPIO Port D (APB) base: 0x4000.7000
- GPIO Port D (AHB) base: 0x4005.B000
- GPIO Port E (APB) base: 0x4002.4000
- GPIO Port E (AHB) base: 0x4005.C000
- GPIO Port F (APB) base: 0x4002.5000
- GPIO Port F (AHB) base: 0x4005.D000
- Offset 0x528
- Type RW, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description						
31:8	reserved	RO	0x0000.0000	<p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p>						
7:0	GPIOAMSEL	RW	0x00	<p>GPIO Analog Mode Select</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The analog function of the pin is disabled, the isolation is enabled, and the pin is capable of digital functions as specified by the other GPIO configuration registers.</td> </tr> <tr> <td>1</td> <td>The analog function of the pin is enabled, the isolation is disabled, and the pin is capable of analog functions.</td> </tr> </tbody> </table> <p>Note: This register and bits are only valid for GPIO signals that share analog function through a unified I/O pad.</p> <p>The reset state of this register is 0 for all signals.</p>	Value	Description	0	The analog function of the pin is disabled, the isolation is enabled, and the pin is capable of digital functions as specified by the other GPIO configuration registers.	1	The analog function of the pin is enabled, the isolation is disabled, and the pin is capable of analog functions.
Value	Description									
0	The analog function of the pin is disabled, the isolation is enabled, and the pin is capable of digital functions as specified by the other GPIO configuration registers.									
1	The analog function of the pin is enabled, the isolation is disabled, and the pin is capable of analog functions.									

19.7.19 GPIO Data (GPIODATA)

- GPIO Port A (APB) base: 0x4000.4000
- GPIO Port A (AHB) base: 0x4005.8000
- GPIO Port B (APB) base: 0x4000.5000
- GPIO Port B (AHB) base: 0x4005.9000
- GPIO Port C (APB) base: 0x4000.6000
- GPIO Port C (AHB) base: 0x4005.A000
- GPIO Port D (APB) base: 0x4000.7000
- GPIO Port D (AHB) base: 0x4005.B000
- GPIO Port E (APB) base: 0x4002.4000
- GPIO Port E (AHB) base: 0x4005.C000
- GPIO Port F (APB) base: 0x4002.5000
- GPIO Port F (AHB) base: 0x4005.D000
- Offset 0x000
- Type RW, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DATA	RW	0x00	<p>GPIO Data</p> <p>This register is virtually mapped to 256 locations in the address space. To facilitate the reading and writing of data to these registers by independent drivers, the data read from and written to the registers are masked by the eight address lines [9:2]. Reads from this register return its current state. Writes to this register only affect bits that are not masked by ADDR[9:2] and are configured as outputs.</p>

Figure 10-3. GPIODATA Write Example

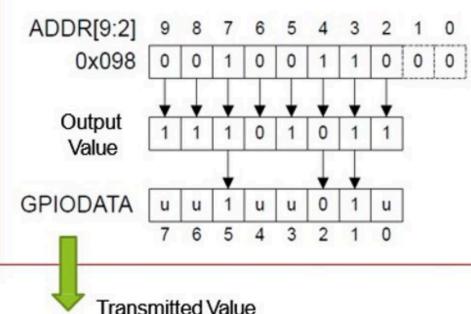
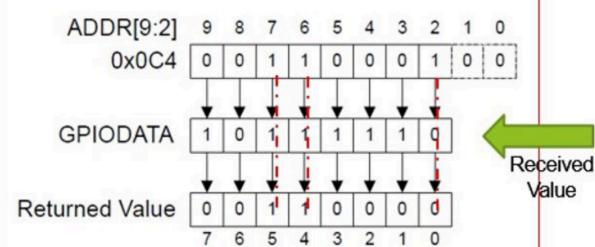


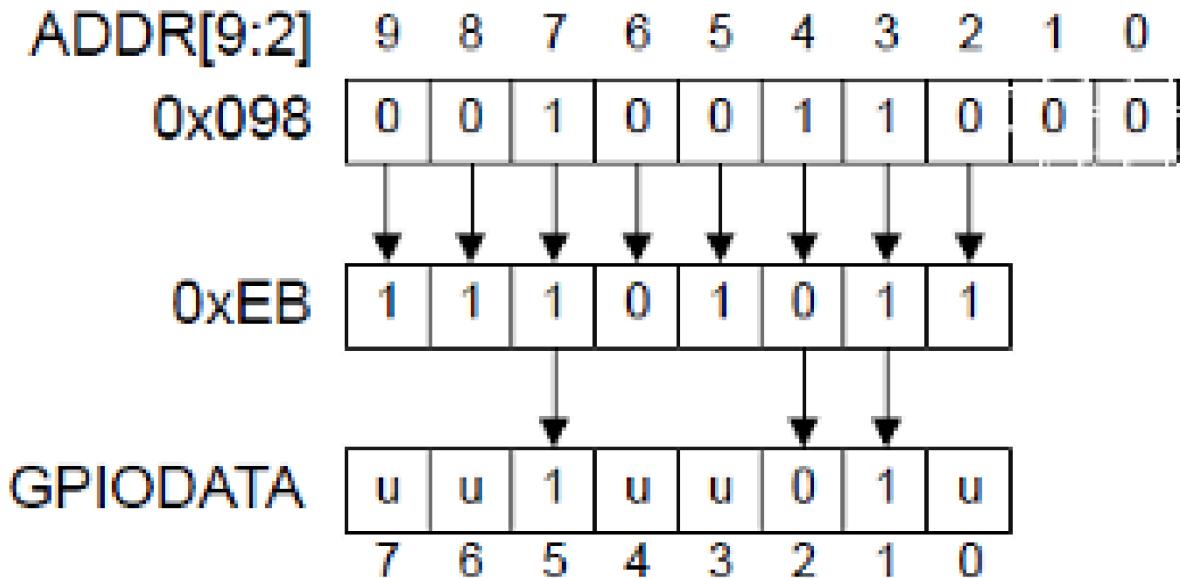
Figure 10-4. GPIODATA Read Example



19.7.19.1 GPIO write example

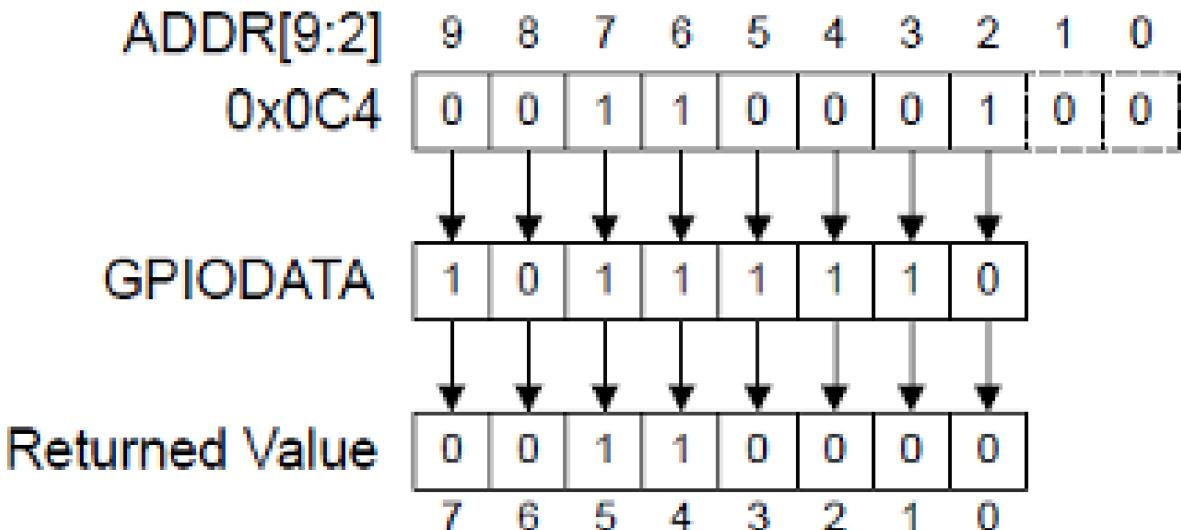
During a write, if the address bit associated with that data bit is set, the value of the **GPIODATA** (page 153) register is altered. If the address bit is cleared, the data bit is left unchanged.

For example, writing a value of 0xEB to the address **GPIODATA** (page 153) + 0x098 has the results shown below, where u indicates that data which are not changed by the write operation. The example below shows **GPIODATA** (page 153) bits 5, 2, and 1 being written.



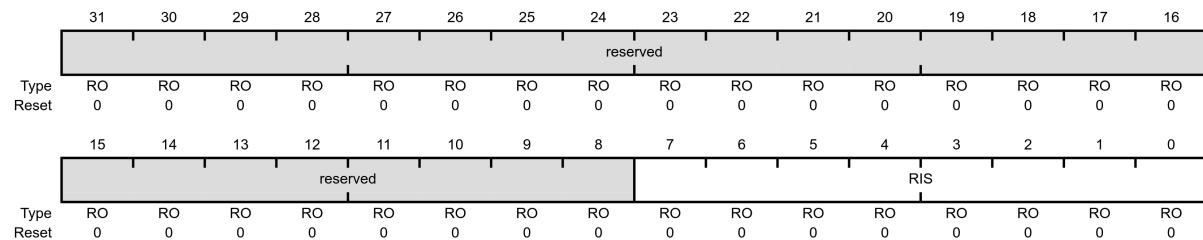
19.7.19.2 GPIO read example

During a read, if the address bit associated with the data bit is set, the value is read. If the address bit associated with the data bit is cleared, the data bit is read as a zero, regardless of its actual value. For example, reading address **GPIODATA** (page 153) + 0x0C4 yields the value shown below. The example below shows how to read **GPIODATA** (page 153) bits 5, 4, and 0.



19.7.20 GPIO Raw Interrupt Status (GPIORIS)

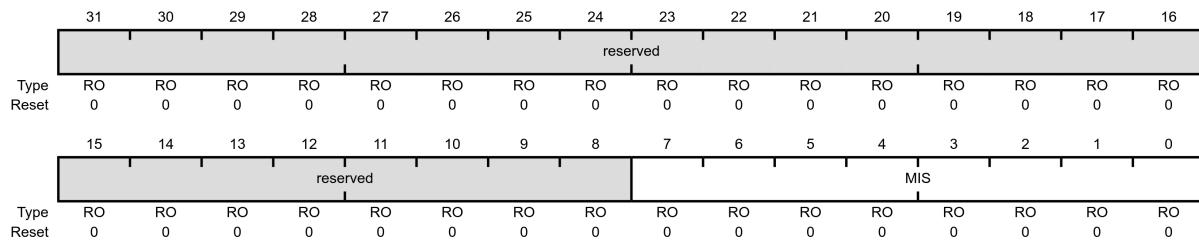
- GPIO Port A (APB) base: 0x4000.4000
- GPIO Port A (AHB) base: 0x4005.8000
- GPIO Port B (APB) base: 0x4000.5000
- GPIO Port B (AHB) base: 0x4005.9000
- GPIO Port C (APB) base: 0x4000.6000
- GPIO Port C (AHB) base: 0x4005.A000
- GPIO Port D (APB) base: 0x4000.7000
- GPIO Port D (AHB) base: 0x4005.B000
- GPIO Port E (APB) base: 0x4002.4000
- GPIO Port E (AHB) base: 0x4005.C000
- GPIO Port F (APB) base: 0x4002.5000
- GPIO Port F (AHB) base: 0x4005.D000
- Offset 0x414
- Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description							
31:8	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.							
7:0	RIS	RO	0x00	<p>GPIO Interrupt Raw Status</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>An interrupt condition has not occurred on the corresponding pin.</td> </tr> <tr> <td>1</td> <td>An interrupt condition has occurred on the corresponding pin.</td> </tr> </tbody> </table> <p>For edge-detect interrupts, this bit is cleared by writing a 1 to the corresponding bit in the GPIOICR (page 157) register.</p> <p>For a GPIO level-detect interrupt, the bit is cleared when the level is de-asserted.</p>		Value	Description	0	An interrupt condition has not occurred on the corresponding pin.	1	An interrupt condition has occurred on the corresponding pin.
Value	Description										
0	An interrupt condition has not occurred on the corresponding pin.										
1	An interrupt condition has occurred on the corresponding pin.										

19.7.21 GPIO Masked Interrupt Status (GPIO MIS)

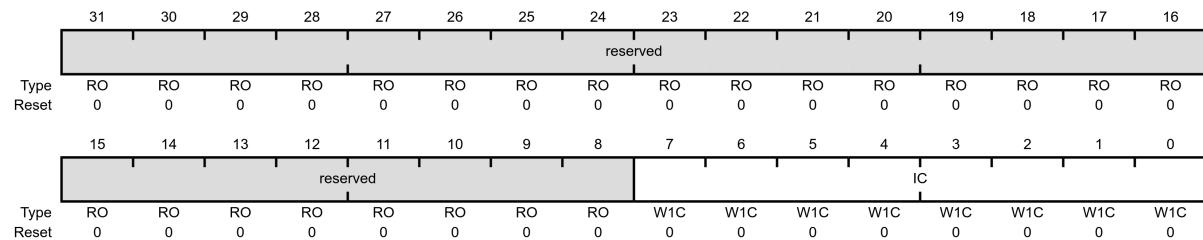
- GPIO Port A (APB) base: 0x4000.4000
- GPIO Port A (AHB) base: 0x4005.8000
- GPIO Port B (APB) base: 0x4000.5000
- GPIO Port B (AHB) base: 0x4005.9000
- GPIO Port C (APB) base: 0x4000.6000
- GPIO Port C (AHB) base: 0x4005.A000
- GPIO Port D (APB) base: 0x4000.7000
- GPIO Port D (AHB) base: 0x4005.B000
- GPIO Port E (APB) base: 0x4002.4000
- GPIO Port E (AHB) base: 0x4005.C000
- GPIO Port F (APB) base: 0x4002.5000
- GPIO Port F (AHB) base: 0x4005.D000
- Offset 0x418
- Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description						
31:8	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
7:0	MIS	RO	0x00	<p>GPIO Masked Interrupt Status</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>An interrupt condition on the corresponding pin is masked or has not occurred.</td> </tr> <tr> <td>1</td> <td>An interrupt condition on the corresponding pin has triggered an interrupt to the interrupt controller.</td> </tr> </tbody> </table> <p>For edge-detect interrupts, this bit is cleared by writing a 1 to the corresponding bit in the GPIOICR (page 157) register.</p> <p>For a GPIO level-detect interrupt, the bit is cleared when the level is de-asserted.</p>	Value	Description	0	An interrupt condition on the corresponding pin is masked or has not occurred.	1	An interrupt condition on the corresponding pin has triggered an interrupt to the interrupt controller.
Value	Description									
0	An interrupt condition on the corresponding pin is masked or has not occurred.									
1	An interrupt condition on the corresponding pin has triggered an interrupt to the interrupt controller.									

19.7.22 GPIO Interrupt Clear (GPIOICR)

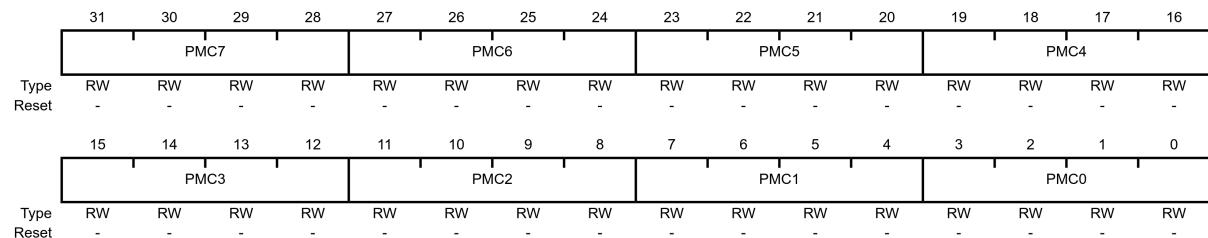
- GPIO Port A (APB) base: 0x4000.4000
- GPIO Port A (AHB) base: 0x4005.8000
- GPIO Port B (APB) base: 0x4000.5000
- GPIO Port B (AHB) base: 0x4005.9000
- GPIO Port C (APB) base: 0x4000.6000
- GPIO Port C (AHB) base: 0x4005.A000
- GPIO Port D (APB) base: 0x4000.7000
- GPIO Port D (AHB) base: 0x4005.B000
- GPIO Port E (APB) base: 0x4002.4000
- GPIO Port E (AHB) base: 0x4005.C000
- GPIO Port F (APB) base: 0x4002.5000
- GPIO Port F (AHB) base: 0x4005.D000
- Offset 0x41C
- Type W1C, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description						
31:8	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
7:0	IC	W1C	0x00	GPIO Interrupt Clear <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The corresponding interrupt is unaffected.</td> </tr> <tr> <td>1</td> <td>The corresponding interrupt is cleared.</td> </tr> </tbody> </table>	Value	Description	0	The corresponding interrupt is unaffected.	1	The corresponding interrupt is cleared.
Value	Description									
0	The corresponding interrupt is unaffected.									
1	The corresponding interrupt is cleared.									

19.7.23 GPIO Port Control (GPIOPCTL)

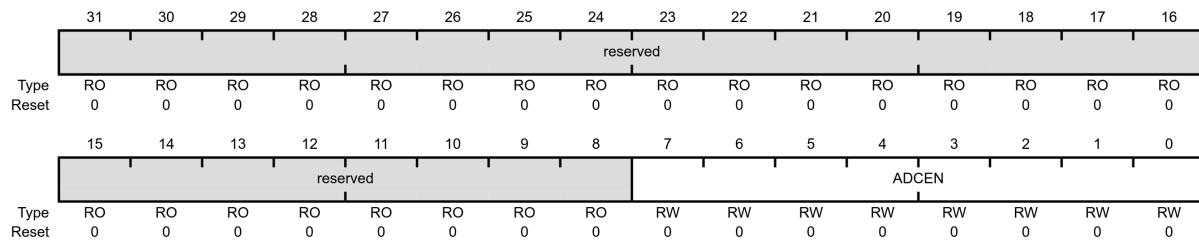
- GPIO Port A (APB) base: 0x4000.4000
- GPIO Port A (AHB) base: 0x4005.8000
- GPIO Port B (APB) base: 0x4000.5000
- GPIO Port B (AHB) base: 0x4005.9000
- GPIO Port C (APB) base: 0x4000.6000
- GPIO Port C (AHB) base: 0x4005.A000
- GPIO Port D (APB) base: 0x4000.7000
- GPIO Port D (AHB) base: 0x4005.B000
- GPIO Port E (APB) base: 0x4002.4000
- GPIO Port E (AHB) base: 0x4005.C000
- GPIO Port F (APB) base: 0x4002.5000
- GPIO Port F (AHB) base: 0x4005.D000
- Offset 0x52C
- Type RW, reset -



Bit/Field	Name	Type	Reset	Description
31:28	PMC7	RW	-	Port Mux Control 7 This field controls the configuration for GPIO pin 7.
27:24	PMC6	RW	-	Port Mux Control 6 This field controls the configuration for GPIO pin 6.
23:20	PMC5	RW	-	Port Mux Control 5 This field controls the configuration for GPIO pin 5.
19:16	PMC4	RW	-	Port Mux Control 4 This field controls the configuration for GPIO pin 4.
15:12	PMC3	RW	-	Port Mux Control 3 This field controls the configuration for GPIO pin 3.
11:8	PMC2	RW	-	Port Mux Control 2 This field controls the configuration for GPIO pin 2.
7:4	PMC1	RW	-	Port Mux Control 1 This field controls the configuration for GPIO pin 1.
3:0	PMC0	RW	-	Port Mux Control 0 This field controls the configuration for GPIO pin 0.

19.7.24 GPIO ADC Control (GPIOADCCTL)

- GPIO Port A (APB) base: 0x4000.4000
- GPIO Port A (AHB) base: 0x4005.8000
- GPIO Port B (APB) base: 0x4000.5000
- GPIO Port B (AHB) base: 0x4005.9000
- GPIO Port C (APB) base: 0x4000.6000
- GPIO Port C (AHB) base: 0x4005.A000
- GPIO Port D (APB) base: 0x4000.7000
- GPIO Port D (AHB) base: 0x4005.B000
- GPIO Port E (APB) base: 0x4002.4000
- GPIO Port E (AHB) base: 0x4005.C000
- GPIO Port F (APB) base: 0x4002.5000
- GPIO Port F (AHB) base: 0x4005.D000
- Offset 0x530
- Type RW, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description						
31:8	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
7:0	ADCEN	RW	0x00	<p>ADC Trigger Enable</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The corresponding pin is not used to trigger the ADC.</td> </tr> <tr> <td>1</td> <td>The corresponding pin is used to trigger the ADC.</td> </tr> </tbody> </table>	Value	Description	0	The corresponding pin is not used to trigger the ADC.	1	The corresponding pin is used to trigger the ADC.
Value	Description									
0	The corresponding pin is not used to trigger the ADC.									
1	The corresponding pin is used to trigger the ADC.									

19.8 Initialisation

1. Enable the clock for the port using the **RCGCGPIO** (page 133) register.
2. Enable or disable analogue functionality using the **GPIOAMSEL** (page 152) register.
3. Configure the pins as GPIO or non-GPIO using the **GPIOPCTL** (page 158) register.
4. Set the pin's direction, like input or output using the **GPIODIR** (page 136) register.
5. Enable (1) or disable (0) the alternate function using the **GPIOAFSEL** (page 148) register.
6. Enable (1) or disable (0) pull-up resistors if needed using the **GPIOPUR** (page 145) register.
7. Enable (1) or disable (0) the digital ports using the **GPIODEN** (page 135) register.

19.8.1 Example

```
GPIO_PORTA_DATA_R      EQU 0x400043FC
GPIO_PORTA_DIR_R       EQU 0x40004400
GPIO_PORTA_AFSEL        EQU 0x40004420
GPIO_PORTA_PUR_R        EQU 0x40004510
GPIO_PORTA_DEN_R        EQU 0x4000451C
GPIO_PORTA_AMSEL_R      EQU 0x40004528
GPIO_PORTA_PCTL_R       EQU 0x4000452C
PA_4567                 EQU 0x400043C0 ; Port A bits 4-7

SYSCTL_RCGCGPIO_R      EQU 0x400FE608

; Initialize Port A
; Enable digital I/O, ensure alternate functions are off.

; Activate clock for Port A
LDR R1, =SYSCTL_RCGCGPIO_R
LDR R0, [R1]
ORR R0, R0, #0x01        ; Turn on GPIO Port A clock
STR R0, [R1]
NOP                      ; Allow time for clock to finish
NOP
NOP
NOP

; Disable analogue functionality
; 1 for enable, 0 for disable
LDR R1, =GPIO_PORTA_AMSEL_R
LDR R0, [R1]
BIC R0, R0, #0xF0        ; disable analogue mode on Port A bits 4-7
STR R0, [R1]

; Configure as GPIO
; 0 for GPIO, and 1 for non-GPIO
LDR R1, =GPIO_PORTA_PCTL_R
LDR R0, [R1]
BIC R0, R0, #0x00FF0000  ; clear Port A bits 4 & 5
BIC R0, R0, #0xFF000000  ; clear Port A bits 6 & 7
STR R0, [R1]

; Set the direction (input or output)
; 1 for output, and 0 for input
LDR R1, =GPIO_PORTA_DIR_R
LDR R0, [R1]
BIC R0, R0, #0xF0        ; set Port A bits 4-7 as input
STR R0, [R1]
```

```

; Disable alternate function
; 1 for enable, and 0 for disable
    LDR R1, =GPIO_PORTA_AFSEL_R
    LDR R0, [R1]
    BIC R0, R0, #0xF0          ; Disable alternate function on Port A bits 4-7
    STR R0, [R1]

; Enable pull-up resistors on switch pins (optional)
; 1 for enable, and 0 for disable
    LDR R1, =GPIO_PORTA_PUR_R
    LDR R0, [R1]
    ORR R0, R0, #0xF0          ; Enable pull-up resistors on Port A bits 4-7
    STR R0, [R1]

; Enable digital port
; 1 for enable, and 0 for disable
    LDR R1, =GPIO_PORTA_DEN_R
    LDR R0, [R1]
    ORR R0, R0, #0xF0          ; Enable digital I/O on Port A bits 4-7
    STR R0, [R1]

```

19.9 Quick reference

RCGCGPIO ([page 133](#)) register (0x400F.E608):

- Write a 1 to the binary value below to enable the port.

Port: --FE DCBA

Binary: 0000 0000

Base addresses:

- GPIO Port A: 0x4000.4000
- GPIO Port B: 0x4000.5000
- GPIO Port C: 0x4000.6000
- GPIO Port D: 0x4000.7000
- GPIO Port E: 0x4002.4000
- GPIO Port F: 0x4002.5000

Offsets:

- **GPIODATA** ([page 153](#)): 0x0000.03FC
- **GPIODIR** ([page 136](#)): 0x0000.0400
- **GPIOAFSEL** ([page 148](#)): 0x0000.0420
- **GIOPPUR** ([page 145](#)): 0x0000.0510
- **GPIODEN** ([page 135](#)): 0x0000.051C
- **GPIOAMSEL** ([page 152](#)): 0x0000.00528
- **GIOPCTL** ([page 158](#)): 0x0000.052C

GPIODATA ([page 153](#)) bit layout for reading and writing:

Port: --76 5432 10--

Binary: 0000 0000 0000

20 TM4C123GH6PM Cortex M4 analogue to digital converter (ADC)

20.1 Functions

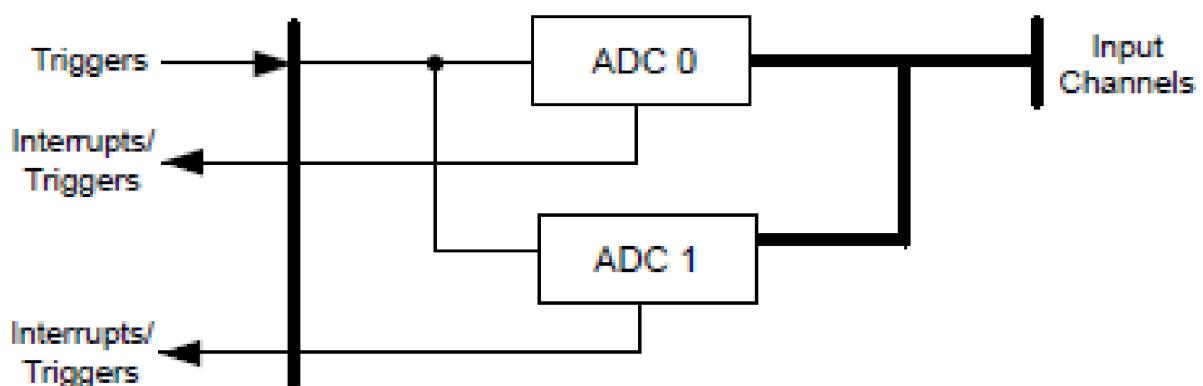
- The TM4C123GH6PM has two ADC modules sharing 12 input channels each with 12-bit resolution, plus an internal temperature sensor.
- Each ADC module has 4 programmable sample sequencers allowing the sampling of multiple analogue input sources.
- Each sample sequencer provides fully programmable input source, trigger events, interrupt generation, and sequencer priority. Conversion values can be diverted to a digital comparator module. Each ADC module provides 8 digital comparators.
- Each digital comparator evaluates the ADC conversion value against its two user-defined values to determine the operation range of the signal. The trigger source for ADC0 and ADC1 may be independent of the two ADC modules may operate from the same trigger source and operate on the same or different inputs.
- A phase shifter can delay the start of the sampling by a specified phase angle.

20.2 Main features

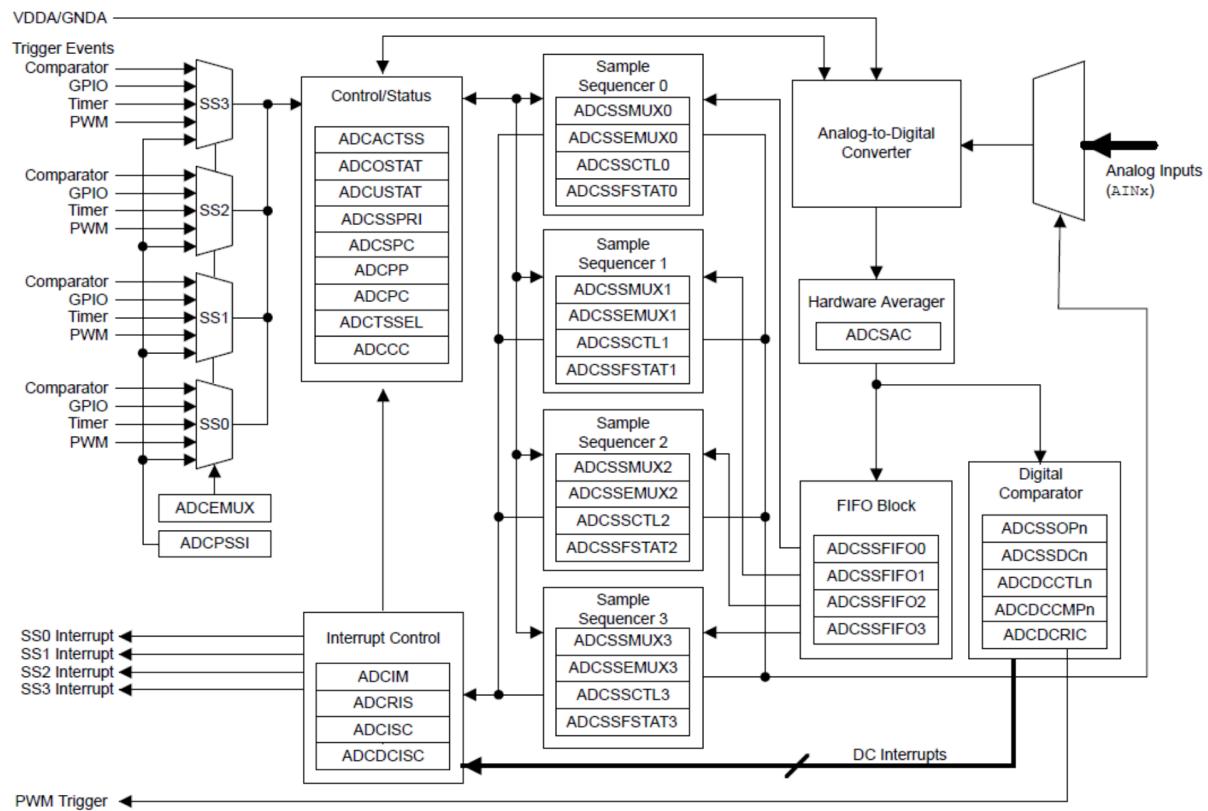
The TM4C123GH6PM provides two ADC modules each having:

- 12 shared analogue input channels
- 12-bit precision analogue to digital converter
- On-chip internal temperature sensor
- Maximum sample rate of up to one million samples per second
- Four programmable sample conversion sequencers from 1 to 8 entries long, with corresponding conversion result FIFOs.
- Flexible trigger control, like controllers, timers, GPIO, analogue comparators, PWM.
- Hardware averaging of up to 64 samples.
- 8 digital comparators
- ADC clock runs at 16 MHz

20.3 Implementation of 2 ADC blocks



20.4 ADC module block diagram



20.5 ADC signal pin names

- The AIN x signals are analogue functions for some GPIO signals.
- Pin or mux assignment is configured by clearing the DEN bit in **GPIODEN** (page 135) and setting the **GPIOAMSEL** (page 152) register.

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type	Description
AIN0	6	PE3	I	Analog	Analog-to-digital converter input 0.
AIN1	7	PE2	I	Analog	Analog-to-digital converter input 1.
AIN2	8	PE1	I	Analog	Analog-to-digital converter input 2.
AIN3	9	PE0	I	Analog	Analog-to-digital converter input 3.
AIN4	64	PD3	I	Analog	Analog-to-digital converter input 4.
AIN5	63	PD2	I	Analog	Analog-to-digital converter input 5.
AIN6	62	PD1	I	Analog	Analog-to-digital converter input 6.
AIN7	61	PD0	I	Analog	Analog-to-digital converter input 7.
AIN8	60	PE5	I	Analog	Analog-to-digital converter input 8.
AIN9	59	PE4	I	Analog	Analog-to-digital converter input 9.
AIN10	58	PB4	I	Analog	Analog-to-digital converter input 10.
AIN11	57	PB5	I	Analog	Analog-to-digital converter input 11.

20.6 ADC register map

Offset	Name	Type	Reset	Description
0x000	ADCACTSS	RW	0x0000.0000	ADC Active Sample Sequencer
0x004	ADCRIS	RO	0x0000.0000	ADC Raw Interrupt Status
0x008	ADCIM	RW	0x0000.0000	ADC Interrupt Mask
0x00C	ADCISC	RW1C	0x0000.0000	ADC Interrupt Status and Clear
0x010	ADCOSTAT	RW1C	0x0000.0000	ADC Overflow Status
0x014	ADCEMUX	RW	0x0000.0000	ADC Event Multiplexer Select
0x018	ADCUSTAT	RW1C	0x0000.0000	ADC Underflow Status
0x01C	ADCTSSEL	RW	0x0000.0000	ADC Trigger Source Select
0x020	ADCSSPRI	RW	0x0000.3210	ADC Sample Sequencer Priority
0x024	ADCSPC	RW	0x0000.0000	ADC Sample Phase Control
0x028	ADCPSSI	RW	-	ADC Processor Sample Sequence Initiate
0x030	ADCSAC	RW	0x0000.0000	ADC Sample Averaging Control
0x034	ADCDICISC	RW1C	0x0000.0000	ADC Digital Comparator Interrupt Status and Clear
0x038	ADCCTL	RW	0x0000.0000	ADC Control
0x040	ADCSSMUX0	RW	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 0
0x044	ADCSCTL0	RW	0x0000.0000	ADC Sample Sequence Control 0
0x048	ADCSSFIFO0	RO	-	ADC Sample Sequence Result FIFO 0
0x04C	ADCSSFSTAT0	RO	0x0000.0100	ADC Sample Sequence FIFO 0 Status
0x050	ADCSSOP0	RW	0x0000.0000	ADC Sample Sequence 0 Operation
0x054	ADCSSDC0	RW	0x0000.0000	ADC Sample Sequence 0 Digital Comparator Select
0x060	ADCSSMUX1	RW	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 1
0x064	ADCSCTL1	RW	0x0000.0000	ADC Sample Sequence Control 1
0x068	ADCSSFIFO1	RO	-	ADC Sample Sequence Result FIFO 1
0x06C	ADCSSFSTAT1	RO	0x0000.0100	ADC Sample Sequence FIFO 1 Status
0x070	ADCSSOP1	RW	0x0000.0000	ADC Sample Sequence 1 Operation
0x074	ADCSSDC1	RW	0x0000.0000	ADC Sample Sequence 1 Digital Comparator Select
0x080	ADCSSMUX2	RW	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 2
0x084	ADCSCTL2	RW	0x0000.0000	ADC Sample Sequence Control 2
0x088	ADCSSFIFO2	RO	-	ADC Sample Sequence Result FIFO 2
0x08C	ADCSSFSTAT2	RO	0x0000.0100	ADC Sample Sequence FIFO 2 Status
0x090	ADCSSOP2	RW	0x0000.0000	ADC Sample Sequence 2 Operation
0x094	ADCSSDC2	RW	0x0000.0000	ADC Sample Sequence 2 Digital Comparator Select
0x0A0	ADCSSMUX3	RW	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 3
0x0A4	ADCSCTL3	RW	0x0000.0000	ADC Sample Sequence Control 3
0x0A8	ADCSSFIFO3	RO	-	ADC Sample Sequence Result FIFO 3
0x0AC	ADCSSFSTAT3	RO	0x0000.0100	ADC Sample Sequence FIFO 3 Status
0x0B0	ADCSSOP3	RW	0x0000.0000	ADC Sample Sequence 3 Operation
0x0B4	ADCSSDC3	RW	0x0000.0000	ADC Sample Sequence 3 Digital Comparator Select

Offset	Name	Type	Reset	Description
0xD00	ADCDCRIC	WO	0x0000.0000	ADC Digital Comparator Reset Initial Conditions
0xE00	ADCDCTL0	RW	0x0000.0000	ADC Digital Comparator Control 0
0xE04	ADCDCTL1	RW	0x0000.0000	ADC Digital Comparator Control 1
0xE08	ADCDCTL2	RW	0x0000.0000	ADC Digital Comparator Control 2
0xE0C	ADCDCTL3	RW	0x0000.0000	ADC Digital Comparator Control 3
0xE10	ADCDCTL4	RW	0x0000.0000	ADC Digital Comparator Control 4
0xE14	ADCDCTL5	RW	0x0000.0000	ADC Digital Comparator Control 5
0xE18	ADCDCTL6	RW	0x0000.0000	ADC Digital Comparator Control 6
0xE1C	ADCDCTL7	RW	0x0000.0000	ADC Digital Comparator Control 7
0xE40	ADCDCMP0	RW	0x0000.0000	ADC Digital Comparator Range 0
0xE44	ADCDCMP1	RW	0x0000.0000	ADC Digital Comparator Range 1
0xE48	ADCDCMP2	RW	0x0000.0000	ADC Digital Comparator Range 2
0xE4C	ADCDCMP3	RW	0x0000.0000	ADC Digital Comparator Range 3
0xE50	ADCDCMP4	RW	0x0000.0000	ADC Digital Comparator Range 4
0xE54	ADCDCMP5	RW	0x0000.0000	ADC Digital Comparator Range 5
0xE58	ADCDCMP6	RW	0x0000.0000	ADC Digital Comparator Range 6
0xE5C	ADCDCMP7	RW	0x0000.0000	ADC Digital Comparator Range 7
0xF00	ADCPP	RO	0x0080.20C7	ADC Peripheral Properties
0xF04	ADCPC	RW	0x0000.0007	ADC Peripheral Configuration
0xF08	ADCCC	RW	0x0000.0000	ADC Clock Configuration

20.7 GPIOPCTL for ADC

IO	Pin	Analog Function	Digital Function (GPIOPCTL PMC × Bit Field Encoding)											
			1	2	3	4	5	6	7	8	9	14	15	
PB4	58	AIN10	-	SSI2Clk	-	M0PWM2	-	-	T1CCP0	CAN0Rx	-	-	-	
PB5	57	AIN11	-	SSI2Fss	-	M0PWM3	-	-	T1CCP1	CAN0Tx	-	-	-	
PD0	61	AIN7	SSI3Clk	SSI1Clk	I2C3SCL	M0PWM6	M1PWM0	-	WT2CCP0	-	-	-	-	
PD1	62	AIN6	SSI3Fss	SSI1Fss	I2C3SDA	M0PWM7	M1PWM1	-	WT2CCP1	-	-	-	-	
PD2	63	AIN5	SSI3Rx	SSI1Rx	-	M0FAULT0	-	-	WT3CCP0	USB0EPEN	-	-	-	
PD3	64	AIN4	SSI3Tx	SSI1Tx	-	-	-	IDX0	WT3CCP1	USB0PFLT	-	-	-	
PE0	9	AIN3	U7Rx	-	-	-	-	-	-	-	-	-	-	
PE1	8	AIN2	U7Tx	-	-	-	-	-	-	-	-	-	-	
PE2	7	AIN1	-	-	-	-	-	-	-	-	-	-	-	
PE3	6	AIN0	-	-	-	-	-	-	-	-	-	-	-	
PE4	59	AIN9	U5Rx	-	I2C2SCL	M0PWM4	M1PWM2	-	-	CAN0Rx	-	-	-	
PE5	60	AIN8	U5Tx	-	I2C2SDA	M0PWM5	M1PWM3	-	-	CAN0Tx	-	-	-	

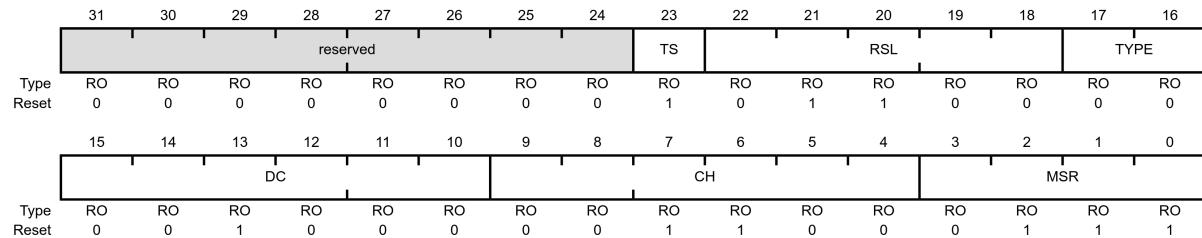
20.8 Functional description

- The TM4C123GH6PM ADC collects sample data by using a programmable sequence-based approach.
- Each sample sequence is a fully programmed series of consecutive (back-to-back) samples, allowing the ADC to collect data from multiple input sources without having to be re-configured or serviced by the processor.
- The programming of each sample in the sample sequence includes parameters such as the input source and mode (differential versus single-ended input), interrupt generation on sample completion, and the indicator for the last sample in the sequence.
- The sampling control and data capture is handled by the sample sequencers.
- For a given sample sequence, each sample is defined by bit fields in the ADC Sample Sequence Input Multiplexer Select (ADCSSMUXn) ([page 175](#)) and ADC Sample Sequence Control (ADCSSCTLn) ([page 183](#)) registers.

20.9 Registers

20.9.1 ADC Peripheral Properties (ADCPP)

- ADC0 base: 0x4003.8000
- ADC1 base: 0x4003.9000
- Offset 0xFC0
- Type RO, reset 0x00B0.20C7

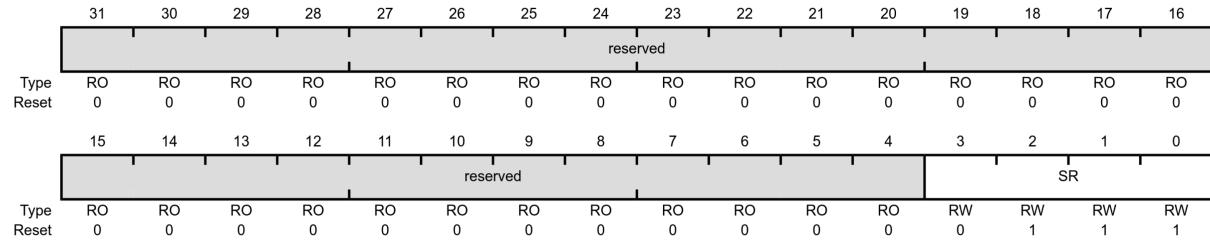


Bit/Field	Name	Type	Reset	Description						
31:24	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
23	TS	RO	0x1	<p>Temperature Sensor</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The ADC module does not have a temperature sensor.</td> </tr> <tr> <td>1</td> <td>The ADC module has a temperature sensor.</td> </tr> </tbody> </table> <p>This field provides the similar information as the legacy DC1 register TEMPSNS bit.</p>	Value	Description	0	The ADC module does not have a temperature sensor.	1	The ADC module has a temperature sensor.
Value	Description									
0	The ADC module does not have a temperature sensor.									
1	The ADC module has a temperature sensor.									
22:18	RSL	RO	0xC	<p>Resolution</p> <p>This field specifies the maximum number of binary bits used to represent the converted sample. The field is encoded as a binary value, in the range of 0 to 32 bits.</p>						
17:16	TYPE	RO	0x0	<p>ADC Architecture</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>SAR</td> </tr> <tr> <td>0x1 - 0x3</td> <td>Reserved</td> </tr> </tbody> </table>	Value	Description	0x0	SAR	0x1 - 0x3	Reserved
Value	Description									
0x0	SAR									
0x1 - 0x3	Reserved									
15:10	DC	RO	0x8	<p>Digital Comparator Count</p> <p>This field specifies the number of ADC digital comparators available to the converter. The field is encoded as a binary value, in the range of 0 to 63.</p> <p>This field provides similar information to the legacy DC9 register ADCnDCn bits.</p>						

Bit/Field	Name	Type	Reset	Description																				
9:4	CH	RO	0xC	<p>ADC Channel Count</p> <p>This field specifies the number of ADC input channels available to the converter. This field is encoded as a binary value, in the range of 0 to 63.</p> <p>This field provides similar information to the legacy DC3 and DC8 register ADCnAINn bits.</p>																				
3:0	MSR	RO	0x7	<p>Maximum ADC Sample Rate</p> <p>This field specifies the maximum number of ADC conversions per second. The MSR field is encoded as follows:</p> <table data-bbox="659 662 976 1111"> <thead> <tr> <th data-bbox="659 662 770 707">Value</th> <th data-bbox="770 662 976 707">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="659 707 770 752">0x0</td> <td data-bbox="770 707 976 752">Reserved</td> </tr> <tr> <td data-bbox="659 752 770 797">0x1</td> <td data-bbox="770 752 976 797">125 ksps</td> </tr> <tr> <td data-bbox="659 797 770 842">0x2</td> <td data-bbox="770 797 976 842">Reserved</td> </tr> <tr> <td data-bbox="659 842 770 887">0x3</td> <td data-bbox="770 842 976 887">250 ksps</td> </tr> <tr> <td data-bbox="659 887 770 932">0x4</td> <td data-bbox="770 887 976 932">Reserved</td> </tr> <tr> <td data-bbox="659 932 770 977">0x5</td> <td data-bbox="770 932 976 977">500 ksps</td> </tr> <tr> <td data-bbox="659 977 770 1021">0x6</td> <td data-bbox="770 977 976 1021">Reserved</td> </tr> <tr> <td data-bbox="659 1021 770 1066">0x7</td> <td data-bbox="770 1021 976 1066">1 Msps</td> </tr> <tr> <td data-bbox="659 1066 770 1111">0x8 - 0xF</td> <td data-bbox="770 1066 976 1111">Reserved</td> </tr> </tbody> </table>	Value	Description	0x0	Reserved	0x1	125 ksps	0x2	Reserved	0x3	250 ksps	0x4	Reserved	0x5	500 ksps	0x6	Reserved	0x7	1 Msps	0x8 - 0xF	Reserved
Value	Description																							
0x0	Reserved																							
0x1	125 ksps																							
0x2	Reserved																							
0x3	250 ksps																							
0x4	Reserved																							
0x5	500 ksps																							
0x6	Reserved																							
0x7	1 Msps																							
0x8 - 0xF	Reserved																							

20.9.2 ADC Peripheral Configuration (ADCPC)

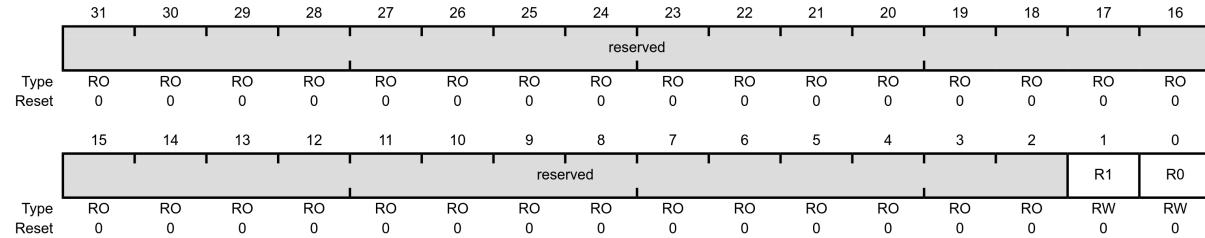
- ADC0 base: 0x4003.8000
- ADC1 base: 0x4003.9000
- Offset 0xFC4
- Type RW, reset 0x0000.0007



Bit/Field	Name	Type	Reset	Description																				
31:4	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.																				
3:0	SR	RW	0x7	<p>ADC Sample Rate</p> <p>This field specifies the number of ADC conversions per second and is used in Run, Sleep, and Deep-Sleep modes. The field encoding is based on the legacy RGCG0 register encoding. The programmed sample rate cannot exceed the maximum sample rate specified by the MSR field in the ADCPP register. The SR field is encoded as follows:</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr><td>0x0</td><td>Reserved</td></tr> <tr><td>0x1</td><td>125 ksps</td></tr> <tr><td>0x2</td><td>Reserved</td></tr> <tr><td>0x3</td><td>250 ksps</td></tr> <tr><td>0x4</td><td>Reserved</td></tr> <tr><td>0x5</td><td>500 ksps</td></tr> <tr><td>0x6</td><td>Reserved</td></tr> <tr><td>0x7</td><td>1 Msps</td></tr> <tr><td>0x8 - 0xF</td><td>Reserved</td></tr> </tbody> </table>	Value	Description	0x0	Reserved	0x1	125 ksps	0x2	Reserved	0x3	250 ksps	0x4	Reserved	0x5	500 ksps	0x6	Reserved	0x7	1 Msps	0x8 - 0xF	Reserved
Value	Description																							
0x0	Reserved																							
0x1	125 ksps																							
0x2	Reserved																							
0x3	250 ksps																							
0x4	Reserved																							
0x5	500 ksps																							
0x6	Reserved																							
0x7	1 Msps																							
0x8 - 0xF	Reserved																							

20.9.3 Analogue-to-Digital Converter Run Mode Clock Gating Control (RCGCADC)

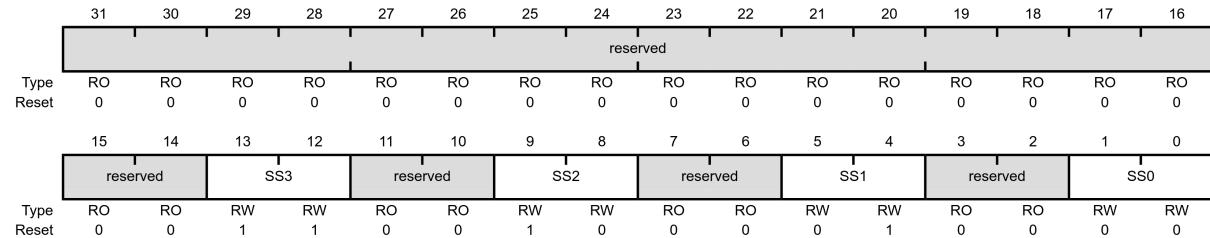
- Base 0x400F.E000
- Offset 0x638
- Type RW, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	R1	RW	0	ADC Module 1 Run Mode Clock Gating Control Value Description 0 ADC module 1 is disabled. 1 Enable and provide a clock to ADC module 1 in Run mode.
0	R0	RW	0	ADC Module 0 Run Mode Clock Gating Control Value Description 0 ADC module 0 is disabled. 1 Enable and provide a clock to ADC module 0 in Run mode.

20.9.4 ADC Sample Sequencer Priority (ADCSSPRI)

- ADC0 base: 0x4003.8000
- ADC1 base: 0x4003.9000
- Offset 0x020
- Type RW, reset 0x0000.3210



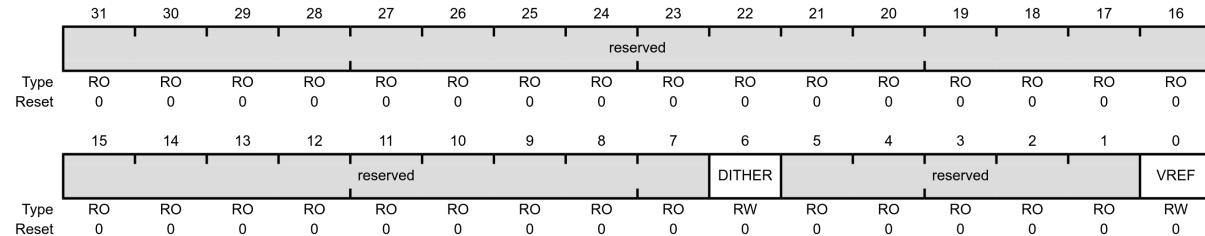
Bit/Field	Name	Type	Reset	Description
31:14	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13:12	SS3	RW	0x3	SS3 Priority This field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 3. A priority encoding of 0x0 is highest and 0x3 is lowest. The priorities assigned to the sequencers must be uniquely mapped. The ADC may not operate properly if two or more fields are equal.
11:10	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9:8	SS2	RW	0x2	SS2 Priority This field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 2. A priority encoding of 0x0 is highest and 0x3 is lowest. The priorities assigned to the sequencers must be uniquely mapped. The ADC may not operate properly if two or more fields are equal.
7:6	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
5:4	SS1	RW	0x1	<p>SS1 Priority</p> <p>This field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 1. A priority encoding of 0x0 is highest and 0x3 is lowest. The priorities assigned to the sequencers must be uniquely mapped. The ADC may not operate properly if two or more fields are equal.</p>
3:2	reserved	RO	0x0	<p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p>
1:0	SS0	RW	0x0	<p>SS0 Priority</p> <p>This field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 0. A priority encoding of 0x0 is highest and 0x3 is lowest. The priorities assigned to the sequencers must be uniquely mapped. The ADC may not operate properly if two or more fields are equal.</p>

20.9.5 ADC Control (ADCCTL)

Note that the values set in this register apply to **all** ADC modules. It is not possible to set one module to use internal references and another to use external references.

- ADC0 base: 0x4003.8000
- ADC1 base: 0x4003.9000
- Offset 0x038
- Type RW, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description						
31:7	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
6	DITHER	RW	0	<p>Dither Mode Enable</p> <table> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>Dither mode disabled</td> </tr> <tr> <td>1</td> <td>Dither mode enabled</td> </tr> </table>	Value	Description	0	Dither mode disabled	1	Dither mode enabled
Value	Description									
0	Dither mode disabled									
1	Dither mode enabled									
5:1	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
0	VREF	RW	0x0	<p>Voltage Reference Select</p> <table> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0x0</td> <td>VDDA and GND are the voltage references for all ADC modules.</td> </tr> <tr> <td>0x1</td> <td>Reserved</td> </tr> </table>	Value	Description	0x0	VDDA and GND are the voltage references for all ADC modules.	0x1	Reserved
Value	Description									
0x0	VDDA and GND are the voltage references for all ADC modules.									
0x1	Reserved									

20.9.6 ADC Sample Sequence Input Multiplexer Select 0 (ADCSSMUX0)

- ADC0 base: 0x4003.8000
- ADC1 base: 0x4003.9000
- Offset 0x040
- Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	MUX7				MUX6				MUX5				MUX4			
Type	RW	RW	RW	RW												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MUX3				MUX2				MUX1				MUX0			
Type	RW	RW	RW	RW												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:28	MUX7	RW	0x0	8th Sample Input Select The MUX7 field is used during the eighth sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion. The value set here indicates the corresponding pin, for example, a value of 0x1 indicates the input is AIN1.
27:24	MUX6	RW	0x0	7th Sample Input Select The MUX6 field is used during the seventh sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.
23:20	MUX5	RW	0x0	6th Sample Input Select The MUX5 field is used during the sixth sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.
19:16	MUX4	RW	0x0	5th Sample Input Select The MUX4 field is used during the fifth sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.
15:12	MUX3	RW	0x0	4th Sample Input Select The MUX3 field is used during the fourth sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.
11:8	MUX2	RW	0x0	3rd Sample Input Select The MUX2 field is used during the third sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.

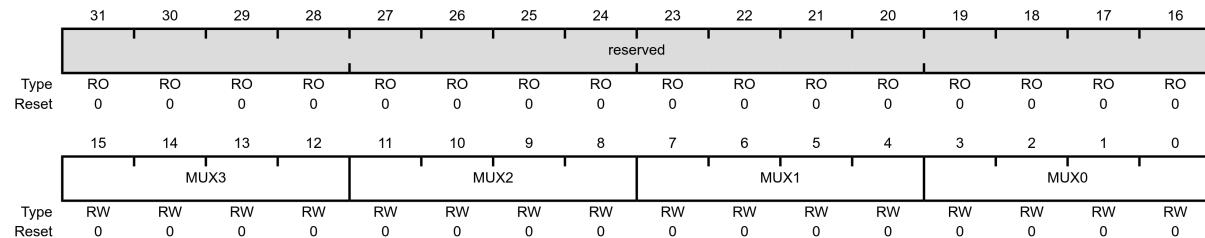
Bit/Field	Name	Type	Reset	Description
7:4	MUX1	RW	0x0	<p>2nd Sample Input Select</p> <p>The MUX1 field is used during the second sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.</p>
3:0	MUX0	RW	0x0	<p>1st Sample Input Select</p> <p>The MUX0 field is used during the first sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.</p>

20.9.7 ADC Sample Sequence Input Multiplexer Select n (ADCSSMUXn)

- ADC0 base: 0x4003.8000
- ADC1 base: 0x4003.9000
- Offset 0x060
- Type RW, reset 0x0000.0000

Registers:

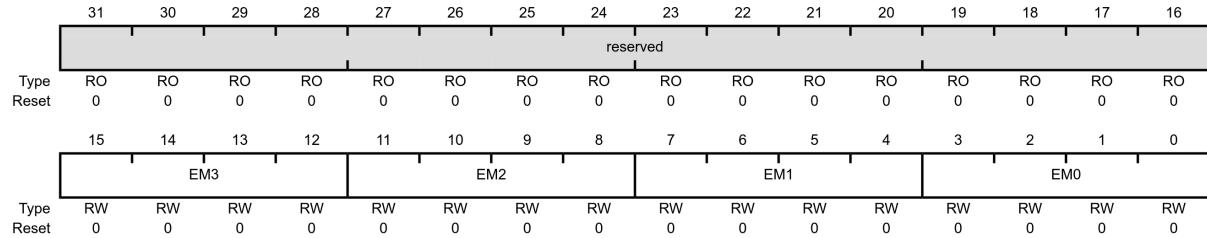
- Register 27: ADC Sample Sequence Input Multiplexer Select 1 (ADCSSMUX1), offset 0x060
- Register 28: ADC Sample Sequence Input Multiplexer Select 2 (ADCSSMUX2), offset 0x080



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:12	MUX3	RW	0x0	4th Sample Input Select
11:8	MUX2	RW	0x0	3rd Sample Input Select
7:4	MUX1	RW	0x0	2nd Sample Input Select
3:0	MUX0	RW	0x0	1st Sample Input Select

20.9.8 ADC Event Multiplexer Select (ADCEMUX)

- ADC0 base: 0x4003.8000
- ADC1 base: 0x4003.9000
- Offset 0x014
- Type RW, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:12	EM3	RW	0x0	SS3 Trigger Select This field selects the trigger source for Sample Sequencer 3.
11:8	EM2	RW	0x0	SS2 Trigger Select This field selects the trigger source for Sample Sequencer 2.
7:4	EM1	RW	0x0	SS1 Trigger Select This field selects the trigger source for Sample Sequencer 1.
3:0	EM0	RW	0x0	SS0 Trigger Select This field selects the trigger source for Sample Sequencer 0.

20.9.8.1 Valid configurations for the trigger select fields

Value	Event
0x0	<p>Processor (default)</p> <p>The trigger is initiated by setting the SS_n bit in the ADCPSSI register.</p>
0x1	<p>Analog Comparator 0</p> <p>This trigger is configured by the Analog Comparator Control 0 (ACCCTL0) register.</p>
0x2	<p>Analog Comparator 1</p> <p>This trigger is configured by the Analog Comparator Control 1 (ACCCTL1) register.</p>
0x3	Reserved
0x4	<p>External (GPIO Pins)</p> <p>This trigger is connected to the GPIO interrupt for the corresponding GPIO.</p> <p>Note: GPIOs that have AIN_x signals as alternate functions can be used to trigger the ADC. However, the pin cannot be used as both a GPIO and an analog input.</p>
0x5	<p>Timer</p> <p>In addition, the trigger must be enabled with the T_nR0TE bit in the GPTMCTL register.</p>
0x6	<p>PWM generator 0</p> <p>The PWM generator 0 trigger can be configured with the PWM0 Interrupt and Trigger Enable (PWM0INTEN) register.</p>
0x7	<p>PWM generator 1</p> <p>The PWM generator 1 trigger can be configured with the PWM1INTEN register.</p>
0x8	<p>PWM generator 2</p> <p>The PWM generator 2 trigger can be configured with the PWM2INTEN register.</p>
0x9	<p>PWM generator 3</p> <p>The PWM generator 3 trigger can be configured with the PWM3INTEN register.</p>
0xA - 0xE	Reserved
0xF	Always (continuously sample)

20.9.9 ADC Trigger Source Select (ADCTSSEL)

- The Cortex M4 has two PWM modules.
- Each PWM module has 4 signal generators.

Documentation:

- ADC0 base: 0x4003.8000
- ADC1 base: 0x4003.9000
- Offset 0x01C
- Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved		PS3				reserved				PS2			reserved		
Type	RO	RO	RW	RW	RO	RO	RO	RO	RO	RO	RW	RW	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved		PS1				reserved				PS0			reserved		
Type	RO	RO	RW	RW	RO	RO	RO	RO	RO	RO	RW	RW	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description								
31:30	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.								
29:28	PS3	RW	0x0	<p>Generator 3 PWM Module Trigger Select</p> <p>This field selects in which PWM module the Generator 3 trigger is located.</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Use Generator 3 (and its trigger) in PWM module 0</td> </tr> <tr> <td>0x1</td> <td>Use Generator 3 (and its trigger) in PWM module 1</td> </tr> <tr> <td>0x2 - 0x3</td> <td>Reserved</td> </tr> </tbody> </table>	Value	Description	0x0	Use Generator 3 (and its trigger) in PWM module 0	0x1	Use Generator 3 (and its trigger) in PWM module 1	0x2 - 0x3	Reserved
Value	Description											
0x0	Use Generator 3 (and its trigger) in PWM module 0											
0x1	Use Generator 3 (and its trigger) in PWM module 1											
0x2 - 0x3	Reserved											
27:22	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.								
21:20	PS2	RW	0x0	<p>Generator 2 PWM Module Trigger Select</p> <p>This field selects in which PWM module the Generator 2 trigger is located.</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Use Generator 2 (and its trigger) in PWM module 0</td> </tr> <tr> <td>0x1</td> <td>Use Generator 2 (and its trigger) in PWM module 1</td> </tr> <tr> <td>0x2 - 0x3</td> <td>Reserved</td> </tr> </tbody> </table>	Value	Description	0x0	Use Generator 2 (and its trigger) in PWM module 0	0x1	Use Generator 2 (and its trigger) in PWM module 1	0x2 - 0x3	Reserved
Value	Description											
0x0	Use Generator 2 (and its trigger) in PWM module 0											
0x1	Use Generator 2 (and its trigger) in PWM module 1											
0x2 - 0x3	Reserved											

Bit/Field	Name	Type	Reset	Description								
19:14	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.								
13:12	PS1	RW	0x0	<p>Generator 1 PWM Module Trigger Select</p> <p>This field selects in which PWM module the Generator 1 trigger is located.</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Use Generator 1 (and its trigger) in PWM module 0</td> </tr> <tr> <td>0x1</td> <td>Use Generator 1 (and its trigger) in PWM module 1</td> </tr> <tr> <td>0x2 - 0x3</td> <td>Reserved</td> </tr> </tbody> </table>	Value	Description	0x0	Use Generator 1 (and its trigger) in PWM module 0	0x1	Use Generator 1 (and its trigger) in PWM module 1	0x2 - 0x3	Reserved
Value	Description											
0x0	Use Generator 1 (and its trigger) in PWM module 0											
0x1	Use Generator 1 (and its trigger) in PWM module 1											
0x2 - 0x3	Reserved											
11:6	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.								
5:4	PS0	RW	0x0	<p>Generator 0 PWM Module Trigger Select</p> <p>This field selects in which PWM module the Generator 0 trigger is located.</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Use Generator 0 (and its trigger) in PWM module 0</td> </tr> <tr> <td>0x1</td> <td>Use Generator 0 (and its trigger) in PWM module 1</td> </tr> <tr> <td>0x2 - 0x3</td> <td>Reserved</td> </tr> </tbody> </table>	Value	Description	0x0	Use Generator 0 (and its trigger) in PWM module 0	0x1	Use Generator 0 (and its trigger) in PWM module 1	0x2 - 0x3	Reserved
Value	Description											
0x0	Use Generator 0 (and its trigger) in PWM module 0											
0x1	Use Generator 0 (and its trigger) in PWM module 1											
0x2 - 0x3	Reserved											
3:0	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.								

20.9.10 ADC Clock Configuration (ADCCC)

- ADC0 base: 0x4003.8000
- ADC1 base: 0x4003.9000
- Offset 0xFC8
- Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved																
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved																
Type	RO	RW	RW	RW	RW											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description								
31:4	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.								
3:0	CS	RW	0	<p>ADC Clock Source</p> <p>The following table specifies the clock source that generates the ADC clock input, see Figure 5-5 on page 222.</p> <table border="1"> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td> <p>Either the 16-MHz system clock (if the Phase Lock Loop [PLL] bypass is in effect) or the 16 MHz clock derived from $\text{PLL} \div 25$ (default).</p> <p>Note that when the PLL is bypassed, the system clock must be at least 16 MHz.</p> </td></tr> <tr> <td>0x1</td><td> <p>Precision Internal Oscillator (PIOOSC)</p> <p>The PIOOSC provides a 16-MHz clock source for the ADC. If the PIOOSC is used as the clock source, the ADC module can continue to operate in Deep-Sleep mode.</p> </td></tr> <tr> <td>0x2 - 0xF</td><td>Reserved</td></tr> </tbody> </table>	Value	Description	0x0	<p>Either the 16-MHz system clock (if the Phase Lock Loop [PLL] bypass is in effect) or the 16 MHz clock derived from $\text{PLL} \div 25$ (default).</p> <p>Note that when the PLL is bypassed, the system clock must be at least 16 MHz.</p>	0x1	<p>Precision Internal Oscillator (PIOOSC)</p> <p>The PIOOSC provides a 16-MHz clock source for the ADC. If the PIOOSC is used as the clock source, the ADC module can continue to operate in Deep-Sleep mode.</p>	0x2 - 0xF	Reserved
Value	Description											
0x0	<p>Either the 16-MHz system clock (if the Phase Lock Loop [PLL] bypass is in effect) or the 16 MHz clock derived from $\text{PLL} \div 25$ (default).</p> <p>Note that when the PLL is bypassed, the system clock must be at least 16 MHz.</p>											
0x1	<p>Precision Internal Oscillator (PIOOSC)</p> <p>The PIOOSC provides a 16-MHz clock source for the ADC. If the PIOOSC is used as the clock source, the ADC module can continue to operate in Deep-Sleep mode.</p>											
0x2 - 0xF	Reserved											

20.9.11 ADC Sample Sequence Control 0 (ADCSSCTL0)

- ADC0 base: 0x4003.8000
- ADC1 base: 0x4003.9000
- Offset 0x044
- Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Type	TS7	IE7	END7	D7	TS6	IE6	END6	D6	TS5	IE5	END5	D5	TS4	IE4	END4	D4
Reset	RW 0															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0
Reset	RW 0															

Bit/Field	Name	Type	Reset	Description
31	TS7	RW	0	<p>8th Sample Temp Sensor Select</p> <p>Value Description</p> <p>0 The input pin specified by the ADCSSMUXn (page 175) register is read during the eighth sample of the sample sequence.</p> <p>1 The temperature sensor is read during the eighth sample of the sample sequence.</p>
30	IE7	RW	0	<p>8th Sample Interrupt Enable</p> <p>Value Description</p> <p>0 The raw interrupt is not asserted to the interrupt controller.</p> <p>1 The raw interrupt signal (INR0) bit is asserted at the end of the eighth sample's conversion. If the MASK0 bit in the ADCIM (page 217) register is set, the interrupt is promoted to the interrupt controller.</p> <p>It is legal to have multiple samples within a sequence generate interrupts.</p>
29	END7	RW	0	<p>8th Sample is End of Sequence</p> <p>Value Description</p> <p>0 Another sample in the sequence is the final sample.</p> <p>1 The eighth sample is the last sample of the sequence.</p> <p>It is possible to end the sequence on any sample position. Software must set an ENDn bit somewhere within the sequence. Samples defined after the sample containing a set ENDn bit are not requested for conversion even though the fields may be non-zero.</p>

Bit/Field	Name	Type	Reset	Description						
28	D7	RW	0	<p>8th Sample Differential Input Select</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The analog inputs are not differentially sampled.</td></tr> <tr> <td>1</td><td>The analog input is differentially sampled. The corresponding ADCSSMUXn (page 175) nibble must be set to the pair number “i”, where the paired inputs are “2i and 2i+1”.</td></tr> </tbody> </table> <p>Because the temperature sensor does not have a differential option, this bit must not be set when the TS7 bit is set.</p>	Value	Description	0	The analog inputs are not differentially sampled.	1	The analog input is differentially sampled. The corresponding ADCSSMUXn (page 175) nibble must be set to the pair number “i”, where the paired inputs are “2i and 2i+1”.
Value	Description									
0	The analog inputs are not differentially sampled.									
1	The analog input is differentially sampled. The corresponding ADCSSMUXn (page 175) nibble must be set to the pair number “i”, where the paired inputs are “2i and 2i+1”.									
27	TS6	RW	0	<p>7th Sample Temp Sensor Select</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The input pin specified by the ADCSSMUXn (page 175) register is read during the seventh sample of the sample sequence.</td></tr> <tr> <td>1</td><td>The temperature sensor is read during the seventh sample of the sample sequence.</td></tr> </tbody> </table>	Value	Description	0	The input pin specified by the ADCSSMUXn (page 175) register is read during the seventh sample of the sample sequence.	1	The temperature sensor is read during the seventh sample of the sample sequence.
Value	Description									
0	The input pin specified by the ADCSSMUXn (page 175) register is read during the seventh sample of the sample sequence.									
1	The temperature sensor is read during the seventh sample of the sample sequence.									
26	IE6	RW	0	<p>7th Sample Interrupt Enable</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The raw interrupt is not asserted to the interrupt controller.</td></tr> <tr> <td>1</td><td>The raw interrupt signal (INR0) bit is asserted at the end of the seventh sample’s conversion. If the MASK0 bit in the ADCIM (page 217) register is set, the interrupt is promoted to the interrupt controller.</td></tr> </tbody> </table> <p>It is legal to have multiple samples within a sequence generate interrupts.</p>	Value	Description	0	The raw interrupt is not asserted to the interrupt controller.	1	The raw interrupt signal (INR0) bit is asserted at the end of the seventh sample’s conversion. If the MASK0 bit in the ADCIM (page 217) register is set, the interrupt is promoted to the interrupt controller.
Value	Description									
0	The raw interrupt is not asserted to the interrupt controller.									
1	The raw interrupt signal (INR0) bit is asserted at the end of the seventh sample’s conversion. If the MASK0 bit in the ADCIM (page 217) register is set, the interrupt is promoted to the interrupt controller.									
25	END6	RW	0	<p>7th Sample is End of Sequence</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Another sample in the sequence is the final sample.</td></tr> <tr> <td>1</td><td>The seventh sample is the last sample of the sequence.</td></tr> </tbody> </table> <p>It is possible to end the sequence on any sample position. Software must set an ENDn bit somewhere within the sequence. Samples defined after the sample containing a set ENDn bit are not requested for conversion even though the fields may be non-zero.</p>	Value	Description	0	Another sample in the sequence is the final sample.	1	The seventh sample is the last sample of the sequence.
Value	Description									
0	Another sample in the sequence is the final sample.									
1	The seventh sample is the last sample of the sequence.									

Bit/Field	Name	Type	Reset	Description						
24	D6	RW	0	<p>7th Sample Differential Input Select</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The analog inputs are not differentially sampled.</td></tr> <tr> <td>1</td><td>The analog input is differentially sampled. The corresponding ADCSSMUXn (page 175) nibble must be set to the pair number “i”, where the paired inputs are “2i and 2i+1”.</td></tr> </tbody> </table> <p>Because the temperature sensor does not have a differential option, this bit must not be set when the TS6 bit is set.</p>	Value	Description	0	The analog inputs are not differentially sampled.	1	The analog input is differentially sampled. The corresponding ADCSSMUXn (page 175) nibble must be set to the pair number “i”, where the paired inputs are “2i and 2i+1”.
Value	Description									
0	The analog inputs are not differentially sampled.									
1	The analog input is differentially sampled. The corresponding ADCSSMUXn (page 175) nibble must be set to the pair number “i”, where the paired inputs are “2i and 2i+1”.									
23	TS5	RW	0	<p>6th Sample Temp Sensor Select</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The input pin specified by the ADCSSMUXn (page 175) register is read during the sixth sample of the sample sequence.</td></tr> <tr> <td>1</td><td>The temperature sensor is read during the sixth sample of the sample sequence.</td></tr> </tbody> </table>	Value	Description	0	The input pin specified by the ADCSSMUXn (page 175) register is read during the sixth sample of the sample sequence.	1	The temperature sensor is read during the sixth sample of the sample sequence.
Value	Description									
0	The input pin specified by the ADCSSMUXn (page 175) register is read during the sixth sample of the sample sequence.									
1	The temperature sensor is read during the sixth sample of the sample sequence.									
22	IE5	RW	0	<p>6th Sample Interrupt Enable</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The raw interrupt is not asserted to the interrupt controller.</td></tr> <tr> <td>1</td><td>The raw interrupt signal (INR0) bit is asserted at the end of the sixth sample’s conversion. If the MASK0 bit in the ADCIM (page 217) register is set, the interrupt is promoted to the interrupt controller.</td></tr> </tbody> </table> <p>It is legal to have multiple samples within a sequence generate interrupts.</p>	Value	Description	0	The raw interrupt is not asserted to the interrupt controller.	1	The raw interrupt signal (INR0) bit is asserted at the end of the sixth sample’s conversion. If the MASK0 bit in the ADCIM (page 217) register is set, the interrupt is promoted to the interrupt controller.
Value	Description									
0	The raw interrupt is not asserted to the interrupt controller.									
1	The raw interrupt signal (INR0) bit is asserted at the end of the sixth sample’s conversion. If the MASK0 bit in the ADCIM (page 217) register is set, the interrupt is promoted to the interrupt controller.									
21	END5	RW	0	<p>6th Sample is End of Sequence</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Another sample in the sequence is the final sample.</td></tr> <tr> <td>1</td><td>The sixth sample is the last sample of the sequence.</td></tr> </tbody> </table> <p>It is possible to end the sequence on any sample position. Software must set an ENDn bit somewhere within the sequence. Samples defined after the sample containing a set ENDn bit are not requested for conversion even though the fields may be non-zero.</p>	Value	Description	0	Another sample in the sequence is the final sample.	1	The sixth sample is the last sample of the sequence.
Value	Description									
0	Another sample in the sequence is the final sample.									
1	The sixth sample is the last sample of the sequence.									

Bit/Field	Name	Type	Reset	Description						
20	D5	RW	0	<p>6th Sample Differential Input Select</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The analog inputs are not differentially sampled.</td></tr> <tr> <td>1</td><td>The analog input is differentially sampled. The corresponding ADCSSMUXn (page 175) nibble must be set to the pair number “i”, where the paired inputs are “2i and 2i+1”.</td></tr> </tbody> </table> <p>Because the temperature sensor does not have a differential option, this bit must not be set when the TS5 bit is set.</p>	Value	Description	0	The analog inputs are not differentially sampled.	1	The analog input is differentially sampled. The corresponding ADCSSMUXn (page 175) nibble must be set to the pair number “i”, where the paired inputs are “2i and 2i+1”.
Value	Description									
0	The analog inputs are not differentially sampled.									
1	The analog input is differentially sampled. The corresponding ADCSSMUXn (page 175) nibble must be set to the pair number “i”, where the paired inputs are “2i and 2i+1”.									
19	TS4	RW	0	<p>5th Sample Temp Sensor Select</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The input pin specified by the ADCSSMUXn (page 175) register is read during the fifth sample of the sample sequence.</td></tr> <tr> <td>1</td><td>The temperature sensor is read during the fifth sample of the sample sequence.</td></tr> </tbody> </table>	Value	Description	0	The input pin specified by the ADCSSMUXn (page 175) register is read during the fifth sample of the sample sequence.	1	The temperature sensor is read during the fifth sample of the sample sequence.
Value	Description									
0	The input pin specified by the ADCSSMUXn (page 175) register is read during the fifth sample of the sample sequence.									
1	The temperature sensor is read during the fifth sample of the sample sequence.									
18	IE4	RW	0	<p>5th Sample Interrupt Enable</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The raw interrupt is not asserted to the interrupt controller.</td></tr> <tr> <td>1</td><td>The raw interrupt signal (INR0) bit is asserted at the end of the fifth sample’s conversion. If the MASK0 bit in the ADCIM (page 217) register is set, the interrupt is promoted to the interrupt controller.</td></tr> </tbody> </table> <p>It is legal to have multiple samples within a sequence generate interrupts.</p>	Value	Description	0	The raw interrupt is not asserted to the interrupt controller.	1	The raw interrupt signal (INR0) bit is asserted at the end of the fifth sample’s conversion. If the MASK0 bit in the ADCIM (page 217) register is set, the interrupt is promoted to the interrupt controller.
Value	Description									
0	The raw interrupt is not asserted to the interrupt controller.									
1	The raw interrupt signal (INR0) bit is asserted at the end of the fifth sample’s conversion. If the MASK0 bit in the ADCIM (page 217) register is set, the interrupt is promoted to the interrupt controller.									
17	END4	RW	0	<p>5th Sample is End of Sequence</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Another sample in the sequence is the final sample.</td></tr> <tr> <td>1</td><td>The fifth sample is the last sample of the sequence.</td></tr> </tbody> </table> <p>It is possible to end the sequence on any sample position. Software must set an ENDn bit somewhere within the sequence. Samples defined after the sample containing a set ENDn bit are not requested for conversion even though the fields may be non-zero.</p>	Value	Description	0	Another sample in the sequence is the final sample.	1	The fifth sample is the last sample of the sequence.
Value	Description									
0	Another sample in the sequence is the final sample.									
1	The fifth sample is the last sample of the sequence.									

Bit/Field	Name	Type	Reset	Description						
16	D4	RW	0	<p>5th Sample Differential Input Select</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The analog inputs are not differentially sampled.</td> </tr> <tr> <td>1</td> <td>The analog input is differentially sampled. The corresponding ADCSSMUXn (page 175) nibble must be set to the pair number “i”, where the paired inputs are “2i and 2i+1”.</td> </tr> </tbody> </table> <p>Because the temperature sensor does not have a differential option, this bit must not be set when the TS4 bit is set.</p>	Value	Description	0	The analog inputs are not differentially sampled.	1	The analog input is differentially sampled. The corresponding ADCSSMUXn (page 175) nibble must be set to the pair number “i”, where the paired inputs are “2i and 2i+1”.
Value	Description									
0	The analog inputs are not differentially sampled.									
1	The analog input is differentially sampled. The corresponding ADCSSMUXn (page 175) nibble must be set to the pair number “i”, where the paired inputs are “2i and 2i+1”.									
15	TS3	RW	0	<p>4th Sample Temp Sensor Select</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The input pin specified by the ADCSSMUXn (page 175) register is read during the fourth sample of the sample sequence.</td> </tr> <tr> <td>1</td> <td>The temperature sensor is read during the fourth sample of the sample sequence.</td> </tr> </tbody> </table>	Value	Description	0	The input pin specified by the ADCSSMUXn (page 175) register is read during the fourth sample of the sample sequence.	1	The temperature sensor is read during the fourth sample of the sample sequence.
Value	Description									
0	The input pin specified by the ADCSSMUXn (page 175) register is read during the fourth sample of the sample sequence.									
1	The temperature sensor is read during the fourth sample of the sample sequence.									
14	IE3	RW	0	<p>4th Sample Interrupt Enable</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The raw interrupt is not asserted to the interrupt controller.</td> </tr> <tr> <td>1</td> <td>The raw interrupt signal (INR0) bit is asserted at the end of the fourth sample’s conversion. If the MASK0 bit in the ADCIM (page 217) register is set, the interrupt is promoted to the interrupt controller.</td> </tr> </tbody> </table> <p>It is legal to have multiple samples within a sequence generate interrupts.</p>	Value	Description	0	The raw interrupt is not asserted to the interrupt controller.	1	The raw interrupt signal (INR0) bit is asserted at the end of the fourth sample’s conversion. If the MASK0 bit in the ADCIM (page 217) register is set, the interrupt is promoted to the interrupt controller.
Value	Description									
0	The raw interrupt is not asserted to the interrupt controller.									
1	The raw interrupt signal (INR0) bit is asserted at the end of the fourth sample’s conversion. If the MASK0 bit in the ADCIM (page 217) register is set, the interrupt is promoted to the interrupt controller.									
13	END3	RW	0	<p>4th Sample is End of Sequence</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Another sample in the sequence is the final sample.</td> </tr> <tr> <td>1</td> <td>The fourth sample is the last sample of the sequence.</td> </tr> </tbody> </table> <p>It is possible to end the sequence on any sample position. Software must set an ENDn bit somewhere within the sequence. Samples defined after the sample containing a set ENDn bit are not requested for conversion even though the fields may be non-zero.</p>	Value	Description	0	Another sample in the sequence is the final sample.	1	The fourth sample is the last sample of the sequence.
Value	Description									
0	Another sample in the sequence is the final sample.									
1	The fourth sample is the last sample of the sequence.									

Bit/Field	Name	Type	Reset	Description						
12	D3	RW	0	<p>4th Sample Differential Input Select</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The analog inputs are not differentially sampled.</td></tr> <tr> <td>1</td><td>The analog input is differentially sampled. The corresponding ADCSSMUXn (page 175) nibble must be set to the pair number “i”, where the paired inputs are “2i and 2i+1”.</td></tr> </tbody> </table> <p>Because the temperature sensor does not have a differential option, this bit must not be set when the TS3 bit is set.</p>	Value	Description	0	The analog inputs are not differentially sampled.	1	The analog input is differentially sampled. The corresponding ADCSSMUXn (page 175) nibble must be set to the pair number “i”, where the paired inputs are “2i and 2i+1”.
Value	Description									
0	The analog inputs are not differentially sampled.									
1	The analog input is differentially sampled. The corresponding ADCSSMUXn (page 175) nibble must be set to the pair number “i”, where the paired inputs are “2i and 2i+1”.									
11	TS2	RW	0	<p>3rd Sample Temp Sensor Select</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The input pin specified by the ADCSSMUXn (page 175) register is read during the third sample of the sample sequence.</td></tr> <tr> <td>1</td><td>The temperature sensor is read during the third sample of the sample sequence.</td></tr> </tbody> </table>	Value	Description	0	The input pin specified by the ADCSSMUXn (page 175) register is read during the third sample of the sample sequence.	1	The temperature sensor is read during the third sample of the sample sequence.
Value	Description									
0	The input pin specified by the ADCSSMUXn (page 175) register is read during the third sample of the sample sequence.									
1	The temperature sensor is read during the third sample of the sample sequence.									
10	IE2	RW	0	<p>3rd Sample Interrupt Enable</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The raw interrupt is not asserted to the interrupt controller.</td></tr> <tr> <td>1</td><td>The raw interrupt signal (INR0) bit is asserted at the end of the third sample’s conversion. If the MASK0 bit in the ADCIM (page 217) register is set, the interrupt is promoted to the interrupt controller.</td></tr> </tbody> </table> <p>It is legal to have multiple samples within a sequence generate interrupts.</p>	Value	Description	0	The raw interrupt is not asserted to the interrupt controller.	1	The raw interrupt signal (INR0) bit is asserted at the end of the third sample’s conversion. If the MASK0 bit in the ADCIM (page 217) register is set, the interrupt is promoted to the interrupt controller.
Value	Description									
0	The raw interrupt is not asserted to the interrupt controller.									
1	The raw interrupt signal (INR0) bit is asserted at the end of the third sample’s conversion. If the MASK0 bit in the ADCIM (page 217) register is set, the interrupt is promoted to the interrupt controller.									
9	END2	RW	0	<p>3rd Sample is End of Sequence</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Another sample in the sequence is the final sample.</td></tr> <tr> <td>1</td><td>The third sample is the last sample of the sequence.</td></tr> </tbody> </table> <p>It is possible to end the sequence on any sample position. Software must set an ENDn bit somewhere within the sequence. Samples defined after the sample containing a set ENDn bit are not requested for conversion even though the fields may be non-zero.</p>	Value	Description	0	Another sample in the sequence is the final sample.	1	The third sample is the last sample of the sequence.
Value	Description									
0	Another sample in the sequence is the final sample.									
1	The third sample is the last sample of the sequence.									

Bit/Field	Name	Type	Reset	Description						
8	D2	RW	0	<p>3rd Sample Differential Input Select</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The analog inputs are not differentially sampled.</td></tr> <tr> <td>1</td><td>The analog input is differentially sampled. The corresponding ADCSSMUXn (page 175) nibble must be set to the pair number “i”, where the paired inputs are “2i and 2i+1”.</td></tr> </tbody> </table> <p>Because the temperature sensor does not have a differential option, this bit must not be set when the TS2 bit is set.</p>	Value	Description	0	The analog inputs are not differentially sampled.	1	The analog input is differentially sampled. The corresponding ADCSSMUXn (page 175) nibble must be set to the pair number “i”, where the paired inputs are “2i and 2i+1”.
Value	Description									
0	The analog inputs are not differentially sampled.									
1	The analog input is differentially sampled. The corresponding ADCSSMUXn (page 175) nibble must be set to the pair number “i”, where the paired inputs are “2i and 2i+1”.									
7	TS1	RW	0	<p>2nd Sample Temp Sensor Select</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The input pin specified by the ADCSSMUXn (page 175) register is read during the second sample of the sample sequence.</td></tr> <tr> <td>1</td><td>The temperature sensor is read during the second sample of the sample sequence.</td></tr> </tbody> </table>	Value	Description	0	The input pin specified by the ADCSSMUXn (page 175) register is read during the second sample of the sample sequence.	1	The temperature sensor is read during the second sample of the sample sequence.
Value	Description									
0	The input pin specified by the ADCSSMUXn (page 175) register is read during the second sample of the sample sequence.									
1	The temperature sensor is read during the second sample of the sample sequence.									
6	IE1	RW	0	<p>2nd Sample Interrupt Enable</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The raw interrupt is not asserted to the interrupt controller.</td></tr> <tr> <td>1</td><td>The raw interrupt signal (INR0) bit is asserted at the end of the second sample’s conversion. If the MASK0 bit in the ADCIM (page 217) register is set, the interrupt is promoted to the interrupt controller.</td></tr> </tbody> </table> <p>It is legal to have multiple samples within a sequence generate interrupts.</p>	Value	Description	0	The raw interrupt is not asserted to the interrupt controller.	1	The raw interrupt signal (INR0) bit is asserted at the end of the second sample’s conversion. If the MASK0 bit in the ADCIM (page 217) register is set, the interrupt is promoted to the interrupt controller.
Value	Description									
0	The raw interrupt is not asserted to the interrupt controller.									
1	The raw interrupt signal (INR0) bit is asserted at the end of the second sample’s conversion. If the MASK0 bit in the ADCIM (page 217) register is set, the interrupt is promoted to the interrupt controller.									
5	END1	RW	0	<p>2nd Sample is End of Sequence</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Another sample in the sequence is the final sample.</td></tr> <tr> <td>1</td><td>The second sample is the last sample of the sequence.</td></tr> </tbody> </table> <p>It is possible to end the sequence on any sample position. Software must set an ENDn bit somewhere within the sequence. Samples defined after the sample containing a set ENDn bit are not requested for conversion even though the fields may be non-zero.</p>	Value	Description	0	Another sample in the sequence is the final sample.	1	The second sample is the last sample of the sequence.
Value	Description									
0	Another sample in the sequence is the final sample.									
1	The second sample is the last sample of the sequence.									

Bit/Field	Name	Type	Reset	Description						
4	D1	RW	0	<p>2nd Sample Differential Input Select</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The analog inputs are not differentially sampled.</td></tr> <tr> <td>1</td><td>The analog input is differentially sampled. The corresponding ADCSSMUXn (page 175) nibble must be set to the pair number “i”, where the paired inputs are “2i and 2i+1”.</td></tr> </tbody> </table> <p>Because the temperature sensor does not have a differential option, this bit must not be set when the TS1 bit is set.</p>	Value	Description	0	The analog inputs are not differentially sampled.	1	The analog input is differentially sampled. The corresponding ADCSSMUXn (page 175) nibble must be set to the pair number “i”, where the paired inputs are “2i and 2i+1”.
Value	Description									
0	The analog inputs are not differentially sampled.									
1	The analog input is differentially sampled. The corresponding ADCSSMUXn (page 175) nibble must be set to the pair number “i”, where the paired inputs are “2i and 2i+1”.									
3	TS0	RW	0	<p>1st Sample Temp Sensor Select</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The input pin specified by the ADCSSMUXn (page 175) register is read during the first sample of the sample sequence.</td></tr> <tr> <td>1</td><td>The temperature sensor is read during the first sample of the sample sequence.</td></tr> </tbody> </table>	Value	Description	0	The input pin specified by the ADCSSMUXn (page 175) register is read during the first sample of the sample sequence.	1	The temperature sensor is read during the first sample of the sample sequence.
Value	Description									
0	The input pin specified by the ADCSSMUXn (page 175) register is read during the first sample of the sample sequence.									
1	The temperature sensor is read during the first sample of the sample sequence.									
2	IE0	RW	0	<p>1st Sample Interrupt Enable</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The raw interrupt is not asserted to the interrupt controller.</td></tr> <tr> <td>1</td><td>The raw interrupt signal (INR0) bit is asserted at the end of the first sample’s conversion. If the MASK0 bit in the ADCIM (page 217) register is set, the interrupt is promoted to the interrupt controller.</td></tr> </tbody> </table> <p>It is legal to have multiple samples within a sequence generate interrupts.</p>	Value	Description	0	The raw interrupt is not asserted to the interrupt controller.	1	The raw interrupt signal (INR0) bit is asserted at the end of the first sample’s conversion. If the MASK0 bit in the ADCIM (page 217) register is set, the interrupt is promoted to the interrupt controller.
Value	Description									
0	The raw interrupt is not asserted to the interrupt controller.									
1	The raw interrupt signal (INR0) bit is asserted at the end of the first sample’s conversion. If the MASK0 bit in the ADCIM (page 217) register is set, the interrupt is promoted to the interrupt controller.									
1	END0	RW	0	<p>1st Sample is End of Sequence</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Another sample in the sequence is the final sample.</td></tr> <tr> <td>1</td><td>The first sample is the last sample of the sequence.</td></tr> </tbody> </table> <p>It is possible to end the sequence on any sample position. Software must set an ENDn bit somewhere within the sequence. Samples defined after the sample containing a set ENDn bit are not requested for conversion even though the fields may be non-zero.</p>	Value	Description	0	Another sample in the sequence is the final sample.	1	The first sample is the last sample of the sequence.
Value	Description									
0	Another sample in the sequence is the final sample.									
1	The first sample is the last sample of the sequence.									

Bit/Field	Name	Type	Reset	Description						
0	D0	RW	0	<p>1st Sample Differential Input Select</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The analog inputs are not differentially sampled.</td> </tr> <tr> <td>1</td> <td>The analog input is differentially sampled. The corresponding ADCSSMUXn (page 175) nibble must be set to the pair number “i”, where the paired inputs are “2i and 2i+1”.</td> </tr> </tbody> </table> <p>Because the temperature sensor does not have a differential option, this bit must not be set when the TS0 bit is set.</p>	Value	Description	0	The analog inputs are not differentially sampled.	1	The analog input is differentially sampled. The corresponding ADCSSMUXn (page 175) nibble must be set to the pair number “i”, where the paired inputs are “2i and 2i+1”.
Value	Description									
0	The analog inputs are not differentially sampled.									
1	The analog input is differentially sampled. The corresponding ADCSSMUXn (page 175) nibble must be set to the pair number “i”, where the paired inputs are “2i and 2i+1”.									

20.9.12 ADC Sample Sequence Control n (ADCSSCTLn)

- ADC0 base: 0x4003.8000
- ADC1 base: 0x4003.9000
- Offset 0x064
- Type RW, reset 0x0000.0000

Registers:

- Register 29: ADC Sample Sequence Control 1 (ADCSSCTL1), offset 0x064
- Register 30: ADC Sample Sequence Control 2 (ADCSSCTL2), offset 0x084

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0
Reset	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit/Field	Name	Type	Reset	Description						
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
15	TS3	RW	0	<p>4th Sample Temp Sensor Select</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The input pin specified by the ADCSSMUXn (page 177) register is read during the fourth sample of the sample sequence.</td> </tr> <tr> <td>1</td> <td>The temperature sensor is read during the fourth sample of the sample sequence.</td> </tr> </tbody> </table>	Value	Description	0	The input pin specified by the ADCSSMUXn (page 177) register is read during the fourth sample of the sample sequence.	1	The temperature sensor is read during the fourth sample of the sample sequence.
Value	Description									
0	The input pin specified by the ADCSSMUXn (page 177) register is read during the fourth sample of the sample sequence.									
1	The temperature sensor is read during the fourth sample of the sample sequence.									
14	IE3	RW	0	<p>4th Sample Interrupt Enable</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The raw interrupt is not asserted to the interrupt controller.</td> </tr> <tr> <td>1</td> <td>The raw interrupt signal (INR0 bit) is asserted at the end of the fourth sample's conversion. If the MASK0 bit in the ADCIM (page 217) register is set, the interrupt is promoted to the interrupt controller.</td> </tr> </tbody> </table> <p>It is legal to have multiple samples within a sequence generate interrupts.</p>	Value	Description	0	The raw interrupt is not asserted to the interrupt controller.	1	The raw interrupt signal (INR0 bit) is asserted at the end of the fourth sample's conversion. If the MASK0 bit in the ADCIM (page 217) register is set, the interrupt is promoted to the interrupt controller.
Value	Description									
0	The raw interrupt is not asserted to the interrupt controller.									
1	The raw interrupt signal (INR0 bit) is asserted at the end of the fourth sample's conversion. If the MASK0 bit in the ADCIM (page 217) register is set, the interrupt is promoted to the interrupt controller.									

Bit/Field	Name	Type	Reset	Description						
13	END3	RW	0	<p>4th Sample is End of Sequence</p> <table> <thead> <tr> <th data-bbox="679 316 743 339">Value</th> <th data-bbox="679 316 933 339">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="679 361 700 384">0</td> <td data-bbox="679 361 1303 428">Another sample in the sequence is the final sample.</td> </tr> <tr> <td data-bbox="679 451 700 473">1</td> <td data-bbox="679 451 1298 518">The fourth sample is the last sample of the sequence.</td> </tr> </tbody> </table> <p>It is possible to end the sequence on any sample position. Software must set an ENDn bit somewhere within the sequence.</p> <p>Samples defined after the sample containing a set ENDn bit are not requested for conversion even though the fields may be non-zero.</p>	Value	Description	0	Another sample in the sequence is the final sample.	1	The fourth sample is the last sample of the sequence.
Value	Description									
0	Another sample in the sequence is the final sample.									
1	The fourth sample is the last sample of the sequence.									
12	D3	RW	0	<p>4th Sample Differential Input Select</p> <table> <thead> <tr> <th data-bbox="679 869 743 891">Value</th> <th data-bbox="679 869 933 891">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="679 914 700 936">0</td> <td data-bbox="679 914 1367 936">The analog inputs are not differentially sampled.</td> </tr> <tr> <td data-bbox="679 959 700 981">1</td> <td data-bbox="679 959 1362 1102">The analog input is differentially sampled. The corresponding ADCSSMUXn (page 177) nibble must be set to the pair number 1, where the paired inputs are 3 and 2.</td> </tr> </tbody> </table> <p>Because the temperature sensor does not have a differential option, this bit must not be set when the TS3 bit is set.</p>	Value	Description	0	The analog inputs are not differentially sampled.	1	The analog input is differentially sampled. The corresponding ADCSSMUXn (page 177) nibble must be set to the pair number 1, where the paired inputs are 3 and 2.
Value	Description									
0	The analog inputs are not differentially sampled.									
1	The analog input is differentially sampled. The corresponding ADCSSMUXn (page 177) nibble must be set to the pair number 1, where the paired inputs are 3 and 2.									
11	TS2	RW	0	<p>3rd Sample Temp Sensor Select</p> <table> <thead> <tr> <th data-bbox="679 1340 743 1363">Value</th> <th data-bbox="679 1340 933 1363">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="679 1385 700 1408">0</td> <td data-bbox="679 1385 1335 1484">The input pin specified by the ADCSSMUXn (page 177) register is read during the third sample of the sample sequence.</td> </tr> <tr> <td data-bbox="679 1507 700 1529">1</td> <td data-bbox="679 1507 1356 1551">The temperature sensor is read during the third sample of the sample sequence.</td> </tr> </tbody> </table>	Value	Description	0	The input pin specified by the ADCSSMUXn (page 177) register is read during the third sample of the sample sequence.	1	The temperature sensor is read during the third sample of the sample sequence.
Value	Description									
0	The input pin specified by the ADCSSMUXn (page 177) register is read during the third sample of the sample sequence.									
1	The temperature sensor is read during the third sample of the sample sequence.									

Bit/Field	Name	Type	Reset	Description						
10	IE2	RW	0	<p>3rd Sample Interrupt Enable</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The raw interrupt is not asserted to the interrupt controller.</td> </tr> <tr> <td>1</td> <td>The raw interrupt signal (INR0 bit) is asserted at the end of the third sample's conversion. If the MASK0 bit in the ADCIM (page 217) register is set, the interrupt is promoted to the interrupt controller.</td> </tr> </tbody> </table> <p>It is legal to have multiple samples within a sequence generate interrupts.</p>	Value	Description	0	The raw interrupt is not asserted to the interrupt controller.	1	The raw interrupt signal (INR0 bit) is asserted at the end of the third sample's conversion. If the MASK0 bit in the ADCIM (page 217) register is set, the interrupt is promoted to the interrupt controller.
Value	Description									
0	The raw interrupt is not asserted to the interrupt controller.									
1	The raw interrupt signal (INR0 bit) is asserted at the end of the third sample's conversion. If the MASK0 bit in the ADCIM (page 217) register is set, the interrupt is promoted to the interrupt controller.									
9	END2	RW	0	<p>3rd Sample is End of Sequence</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Another sample in the sequence is the final sample.</td> </tr> <tr> <td>1</td> <td>The third sample is the last sample of the sequence.</td> </tr> </tbody> </table> <p>It is possible to end the sequence on any sample position. Software must set an ENDn bit somewhere within the sequence.</p> <p>Samples defined after the sample containing a set ENDn bit are not requested for conversion even though the fields may be non-zero.</p>	Value	Description	0	Another sample in the sequence is the final sample.	1	The third sample is the last sample of the sequence.
Value	Description									
0	Another sample in the sequence is the final sample.									
1	The third sample is the last sample of the sequence.									
8	D2	RW	0	<p>3rd Sample Differential Input Select</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The analog inputs are not differentially sampled.</td> </tr> <tr> <td>1</td> <td>The analog input is differentially sampled. The corresponding ADCSSMUXn (page 177) nibble must be set to the pair number “i”, where the paired inputs are “2i and 2i+1”.</td> </tr> </tbody> </table> <p>Because the temperature sensor does not have a differential option, this bit must not be set when the TS2 bit is set.</p>	Value	Description	0	The analog inputs are not differentially sampled.	1	The analog input is differentially sampled. The corresponding ADCSSMUXn (page 177) nibble must be set to the pair number “i”, where the paired inputs are “2i and 2i+1”.
Value	Description									
0	The analog inputs are not differentially sampled.									
1	The analog input is differentially sampled. The corresponding ADCSSMUXn (page 177) nibble must be set to the pair number “i”, where the paired inputs are “2i and 2i+1”.									

Bit/Field	Name	Type	Reset	Description						
7	TS1	RW	0	<p>2nd Sample Temp Sensor Select</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The input pin specified by the ADCSSMUXn (page 177) register is read during the second sample of the sample sequence.</td> </tr> <tr> <td>1</td> <td>The temperature sensor is read during the second sample of the sample sequence.</td> </tr> </tbody> </table>	Value	Description	0	The input pin specified by the ADCSSMUXn (page 177) register is read during the second sample of the sample sequence.	1	The temperature sensor is read during the second sample of the sample sequence.
Value	Description									
0	The input pin specified by the ADCSSMUXn (page 177) register is read during the second sample of the sample sequence.									
1	The temperature sensor is read during the second sample of the sample sequence.									
6	IE1	RW	0	<p>2nd Sample Interrupt Enable</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The raw interrupt is not asserted to the interrupt controller.</td> </tr> <tr> <td>1</td> <td>The raw interrupt signal (INR0 bit) is asserted at the end of the second sample's conversion. If the MASK0 bit in the ADCIM (page 217) register is set, the interrupt is promoted to the interrupt controller.</td> </tr> </tbody> </table> <p>It is legal to have multiple samples within a sequence generate interrupts.</p>	Value	Description	0	The raw interrupt is not asserted to the interrupt controller.	1	The raw interrupt signal (INR0 bit) is asserted at the end of the second sample's conversion. If the MASK0 bit in the ADCIM (page 217) register is set, the interrupt is promoted to the interrupt controller.
Value	Description									
0	The raw interrupt is not asserted to the interrupt controller.									
1	The raw interrupt signal (INR0 bit) is asserted at the end of the second sample's conversion. If the MASK0 bit in the ADCIM (page 217) register is set, the interrupt is promoted to the interrupt controller.									
5	END1	RW	0	<p>2nd Sample is End of Sequence</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Another sample in the sequence is the final sample.</td> </tr> <tr> <td>1</td> <td>The second sample is the last sample of the sequence.</td> </tr> </tbody> </table> <p>It is possible to end the sequence on any sample position. Software must set an ENDn bit somewhere within the sequence.</p> <p>Samples defined after the sample containing a set ENDn bit are not requested for conversion even though the fields may be non-zero.</p>	Value	Description	0	Another sample in the sequence is the final sample.	1	The second sample is the last sample of the sequence.
Value	Description									
0	Another sample in the sequence is the final sample.									
1	The second sample is the last sample of the sequence.									

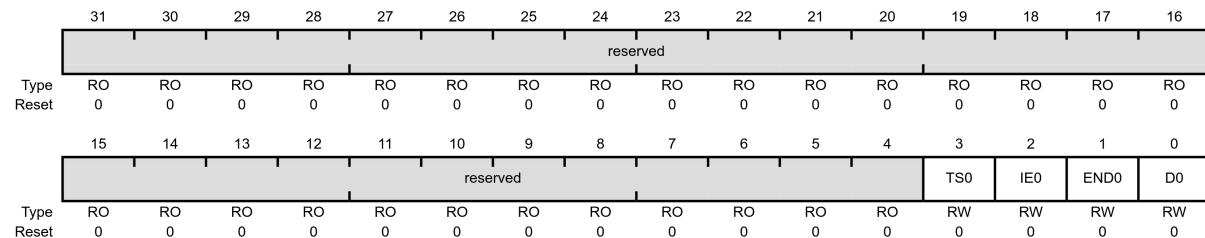
Bit/Field	Name	Type	Reset	Description						
4	D1	RW	0	<p>2nd Sample Differential Input Select</p> <table> <thead> <tr> <th data-bbox="679 309 768 348">Value</th> <th data-bbox="768 309 933 348">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="679 348 768 386">0</td> <td data-bbox="768 348 1362 386">The analog inputs are not differentially sampled.</td> </tr> <tr> <td data-bbox="679 386 768 424">1</td> <td data-bbox="768 386 1362 541">The analog input is differentially sampled. The corresponding ADCSSMUXn (page 177) nibble must be set to the pair number “i”, where the paired inputs are “2i and 2i+1”.</td> </tr> </tbody> </table> <p>Because the temperature sensor does not have a differential option, this bit must not be set when the TS1 bit is set.</p>	Value	Description	0	The analog inputs are not differentially sampled.	1	The analog input is differentially sampled. The corresponding ADCSSMUXn (page 177) nibble must be set to the pair number “i”, where the paired inputs are “2i and 2i+1”.
Value	Description									
0	The analog inputs are not differentially sampled.									
1	The analog input is differentially sampled. The corresponding ADCSSMUXn (page 177) nibble must be set to the pair number “i”, where the paired inputs are “2i and 2i+1”.									
3	TS0	RW	0	<p>1st Sample Temp Sensor Select</p> <table> <thead> <tr> <th data-bbox="679 774 768 813">Value</th> <th data-bbox="768 774 933 813">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="679 813 768 923">0</td> <td data-bbox="768 813 1367 923">The input pin specified by the ADCSSMUXn (page 177) register is read during the first sample of the sample sequence.</td> </tr> <tr> <td data-bbox="679 923 768 1010">1</td> <td data-bbox="768 923 1343 1010">The temperature sensor is read during the first sample of the sample sequence.</td> </tr> </tbody> </table>	Value	Description	0	The input pin specified by the ADCSSMUXn (page 177) register is read during the first sample of the sample sequence.	1	The temperature sensor is read during the first sample of the sample sequence.
Value	Description									
0	The input pin specified by the ADCSSMUXn (page 177) register is read during the first sample of the sample sequence.									
1	The temperature sensor is read during the first sample of the sample sequence.									
2	IE0	RW	0	<p>1st Sample Interrupt Enable</p> <table> <thead> <tr> <th data-bbox="679 1118 768 1156">Value</th> <th data-bbox="768 1118 933 1156">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="679 1156 768 1221">0</td> <td data-bbox="768 1156 1367 1221">The raw interrupt is not asserted to the interrupt controller.</td> </tr> <tr> <td data-bbox="679 1221 768 1421">1</td> <td data-bbox="768 1221 1362 1421">The raw interrupt signal (INR0 bit) is asserted at the end of the first sample’s conversion. If the MASK0 bit in the ADCIM (page 217) register is set, the interrupt is promoted to the interrupt controller.</td> </tr> </tbody> </table> <p>It is legal to have multiple samples within a sequence generate interrupts.</p>	Value	Description	0	The raw interrupt is not asserted to the interrupt controller.	1	The raw interrupt signal (INR0 bit) is asserted at the end of the first sample’s conversion. If the MASK0 bit in the ADCIM (page 217) register is set, the interrupt is promoted to the interrupt controller.
Value	Description									
0	The raw interrupt is not asserted to the interrupt controller.									
1	The raw interrupt signal (INR0 bit) is asserted at the end of the first sample’s conversion. If the MASK0 bit in the ADCIM (page 217) register is set, the interrupt is promoted to the interrupt controller.									

Bit/Field	Name	Type	Reset	Description						
1	END0	RW	0	<p>1st Sample is End of Sequence</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Another sample in the sequence is the final sample.</td> </tr> <tr> <td>1</td> <td>The first sample is the last sample of the sequence.</td> </tr> </tbody> </table> <p>It is possible to end the sequence on any sample position. Software must set an ENDn bit somewhere within the sequence.</p> <p>Samples defined after the sample containing a set ENDn bit are not requested for conversion even though the fields may be non-zero.</p>	Value	Description	0	Another sample in the sequence is the final sample.	1	The first sample is the last sample of the sequence.
Value	Description									
0	Another sample in the sequence is the final sample.									
1	The first sample is the last sample of the sequence.									
0	D0	RW	0	<p>1st Sample Differential Input Select</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The analog inputs are not differentially sampled.</td> </tr> <tr> <td>1</td> <td>The analog input is differentially sampled. The corresponding ADCSSMUXn (page 177) nibble must be set to the pair number 1, where the paired inputs are $2i$ and $2i+1$.</td> </tr> </tbody> </table> <p>Because the temperature sensor does not have a differential option, this bit must not be set when the TS0 bit is set.</p>	Value	Description	0	The analog inputs are not differentially sampled.	1	The analog input is differentially sampled. The corresponding ADCSSMUXn (page 177) nibble must be set to the pair number 1, where the paired inputs are $2i$ and $2i+1$.
Value	Description									
0	The analog inputs are not differentially sampled.									
1	The analog input is differentially sampled. The corresponding ADCSSMUXn (page 177) nibble must be set to the pair number 1, where the paired inputs are $2i$ and $2i+1$.									

20.9.13 ADC Sample Sequence Control 3 (ADCSSCTL3)

- ADC0 base: 0x4003.8000
- ADC1 base: 0x4003.9000
- Offset 0xA4
- Type RW, reset 0x0000.0000

Note: When configuring a sample sequence in this register, the END0 bit must be set.

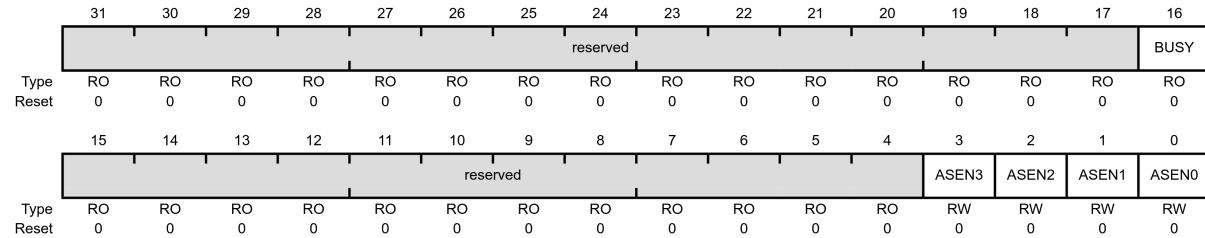


Bit/Field	Name	Type	Reset	Description						
31:4	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
3	TS0	RW	0	<p>1st Sample Temp Sensor Select</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The input pin specified by the ADCSSMUXn (page 177) register is read during the first sample of the sample sequence.</td> </tr> <tr> <td>1</td> <td>The temperature sensor is read during the first sample of the sample sequence.</td> </tr> </tbody> </table>	Value	Description	0	The input pin specified by the ADCSSMUXn (page 177) register is read during the first sample of the sample sequence.	1	The temperature sensor is read during the first sample of the sample sequence.
Value	Description									
0	The input pin specified by the ADCSSMUXn (page 177) register is read during the first sample of the sample sequence.									
1	The temperature sensor is read during the first sample of the sample sequence.									
2	IE0	RW	0	<p>Sample Interrupt Enable</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The raw interrupt is not asserted to the interrupt controller.</td> </tr> <tr> <td>1</td> <td>The raw interrupt signal (INR0 bit) is asserted at the end of this sample's conversion. If the MASK0 bit in the ADCIM (page 217) register is set, the interrupt is promoted to the interrupt controller.</td> </tr> </tbody> </table> <p>It is legal to have multiple samples within a sequence generate interrupts.</p>	Value	Description	0	The raw interrupt is not asserted to the interrupt controller.	1	The raw interrupt signal (INR0 bit) is asserted at the end of this sample's conversion. If the MASK0 bit in the ADCIM (page 217) register is set, the interrupt is promoted to the interrupt controller.
Value	Description									
0	The raw interrupt is not asserted to the interrupt controller.									
1	The raw interrupt signal (INR0 bit) is asserted at the end of this sample's conversion. If the MASK0 bit in the ADCIM (page 217) register is set, the interrupt is promoted to the interrupt controller.									

Bit/Field	Name	Type	Reset	Description						
1	END0	RW	0	<p>End of Sequence</p> <p>This bit must be set before initiating a single sample sequence.</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Sampling and conversion continues.</td> </tr> <tr> <td>1</td> <td>This is the end of sequence.</td> </tr> </tbody> </table>	Value	Description	0	Sampling and conversion continues.	1	This is the end of sequence.
Value	Description									
0	Sampling and conversion continues.									
1	This is the end of sequence.									
0	D0	RW	0	<p>Sample Differential Input Select</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The analog inputs are not differentially sampled.</td> </tr> <tr> <td>1</td> <td>The analog input is differentially sampled. The corresponding ADCSSMUXn (page 177) nibble must be set to the pair number “i”, where the paired inputs are “2i and 2i+1”.</td> </tr> </tbody> </table> <p>Because the temperature sensor does not have a differential option, this bit must not be set when the TS0 bit is set.</p>	Value	Description	0	The analog inputs are not differentially sampled.	1	The analog input is differentially sampled. The corresponding ADCSSMUXn (page 177) nibble must be set to the pair number “i”, where the paired inputs are “2i and 2i+1”.
Value	Description									
0	The analog inputs are not differentially sampled.									
1	The analog input is differentially sampled. The corresponding ADCSSMUXn (page 177) nibble must be set to the pair number “i”, where the paired inputs are “2i and 2i+1”.									

20.9.14 ADC Active Sample Sequencer (ADCACTSS)

- ADC0 base: 0x4003.8000
- ADC1 base: 0x4003.9000
- Offset 0x000
- Type RW, reset 0x0000.0000

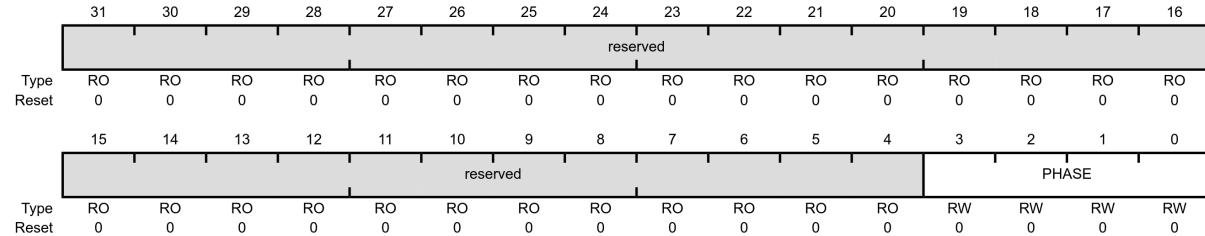


Bit/Field	Name	Type	Reset	Description
31:17	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	BUSY	RO	0	ADC Busy Value Description 0 ADC is idle 1 ADC is busy
15:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	ASEN3	RW	0	ADC SS3 Enable Value Description 0 Sample Sequencer 3 is disabled. 1 Sample Sequencer 3 is enabled.
2	ASEN2	RW	0	ADC SS2 Enable Value Description 0 Sample Sequencer 2 is disabled. 1 Sample Sequencer 2 is enabled.
1	ASEN1	RW	0	ADC SS1 Enable Value Description 0 Sample Sequencer 1 is disabled. 1 Sample Sequencer 1 is enabled.

Bit/Field	Name	Type	Reset	Description						
0	ASEN0	RW	0	ADC SS0 Enable <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Sample Sequencer 0 is disabled.</td> </tr> <tr> <td>1</td> <td>Sample Sequencer 0 is enabled.</td> </tr> </tbody> </table>	Value	Description	0	Sample Sequencer 0 is disabled.	1	Sample Sequencer 0 is enabled.
Value	Description									
0	Sample Sequencer 0 is disabled.									
1	Sample Sequencer 0 is enabled.									

20.9.15 ADC Sample Phase Control (ADCSPC)

- ADC0 base: 0x4003.8000
- ADC1 base: 0x4003.9000
- Offset 0x024
- Type RW, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description																																		
31:4	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.																																		
3:0	PHASE	RW	0x0	<p>Phase Difference</p> <p>This field selects the sample phase difference from the standard sample time.</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>ADC sample lags by 0.0°</td> </tr> <tr> <td>0x1</td> <td>ADC sample lags by 22.5°</td> </tr> <tr> <td>0x2</td> <td>ADC sample lags by 45.0°</td> </tr> <tr> <td>0x3</td> <td>ADC sample lags by 67.5°</td> </tr> <tr> <td>0x4</td> <td>ADC sample lags by 90.0°</td> </tr> <tr> <td>0x5</td> <td>ADC sample lags by 112.5°</td> </tr> <tr> <td>0x6</td> <td>ADC sample lags by 135.0°</td> </tr> <tr> <td>0x7</td> <td>ADC sample lags by 157.5°</td> </tr> <tr> <td>0x8</td> <td>ADC sample lags by 180.0°</td> </tr> <tr> <td>0x9</td> <td>ADC sample lags by 202.5°</td> </tr> <tr> <td>0xA</td> <td>ADC sample lags by 225.0°</td> </tr> <tr> <td>0xB</td> <td>ADC sample lags by 247.5°</td> </tr> <tr> <td>0xC</td> <td>ADC sample lags by 270.0°</td> </tr> <tr> <td>0xD</td> <td>ADC sample lags by 292.5°</td> </tr> <tr> <td>0xE</td> <td>ADC sample lags by 315.0°</td> </tr> <tr> <td>0xF</td> <td>ADC sample lags by 337.5°</td> </tr> </tbody> </table>	Value	Description	0x0	ADC sample lags by 0.0°	0x1	ADC sample lags by 22.5°	0x2	ADC sample lags by 45.0°	0x3	ADC sample lags by 67.5°	0x4	ADC sample lags by 90.0°	0x5	ADC sample lags by 112.5°	0x6	ADC sample lags by 135.0°	0x7	ADC sample lags by 157.5°	0x8	ADC sample lags by 180.0°	0x9	ADC sample lags by 202.5°	0xA	ADC sample lags by 225.0°	0xB	ADC sample lags by 247.5°	0xC	ADC sample lags by 270.0°	0xD	ADC sample lags by 292.5°	0xE	ADC sample lags by 315.0°	0xF	ADC sample lags by 337.5°
Value	Description																																					
0x0	ADC sample lags by 0.0°																																					
0x1	ADC sample lags by 22.5°																																					
0x2	ADC sample lags by 45.0°																																					
0x3	ADC sample lags by 67.5°																																					
0x4	ADC sample lags by 90.0°																																					
0x5	ADC sample lags by 112.5°																																					
0x6	ADC sample lags by 135.0°																																					
0x7	ADC sample lags by 157.5°																																					
0x8	ADC sample lags by 180.0°																																					
0x9	ADC sample lags by 202.5°																																					
0xA	ADC sample lags by 225.0°																																					
0xB	ADC sample lags by 247.5°																																					
0xC	ADC sample lags by 270.0°																																					
0xD	ADC sample lags by 292.5°																																					
0xE	ADC sample lags by 315.0°																																					
0xF	ADC sample lags by 337.5°																																					

20.9.16 ADC Digital Comparator Range n (ADCDCCMPn)

Each comparator has 2 values:

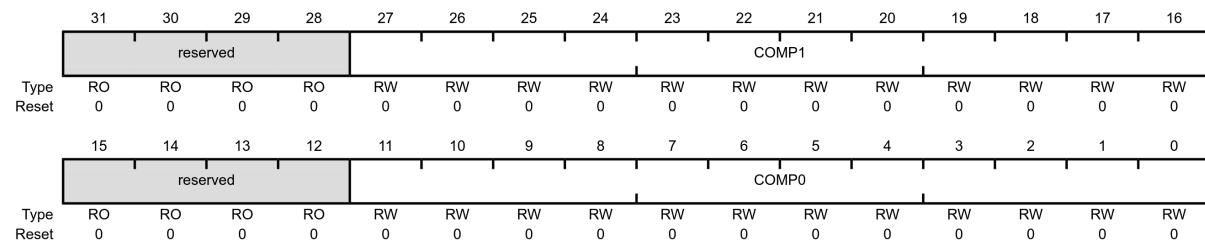
- COMP1: High value
- COMP0: Low value

Documentation:

- ADC0 base: 0x4003.8000
- ADC1 base: 0x4003.9000
- Offset 0xE40
- Type RW, reset 0x0000.0000

Registers:

- Register 48: ADC Digital Comparator Range 0 (ADCDCCMP0), offset 0xE40
- Register 49: ADC Digital Comparator Range 1 (ADCDCCMP1), offset 0xE44
- Register 50: ADC Digital Comparator Range 2 (ADCDCCMP2), offset 0xE48
- Register 51: ADC Digital Comparator Range 3 (ADCDCCMP3), offset 0xE4C
- Register 52: ADC Digital Comparator Range 4 (ADCDCCMP4), offset 0xE50
- Register 53: ADC Digital Comparator Range 5 (ADCDCCMP5), offset 0xE54
- Register 54: ADC Digital Comparator Range 6 (ADCDCCMP6), offset 0xE58
- Register 55: ADC Digital Comparator Range 7 (ADCDCCMP7), offset 0xE5C



Bit/Field	Name	Type	Reset	Description
31:28	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
27:16	COMP1	RW	0x000	Compare 1 The value in this field is compared against the ADC conversion data. The result of the comparison is used to determine if the data lies within the high-band region. Note that the value of COMP1 must be greater than or equal to the value of COMP0.
15:12	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11:0	COMP0	RW	0x000	Compare 0 The value in this field is compared against the ADC conversion data. The result of the comparison is used to determine if the data lies within the low-band region.

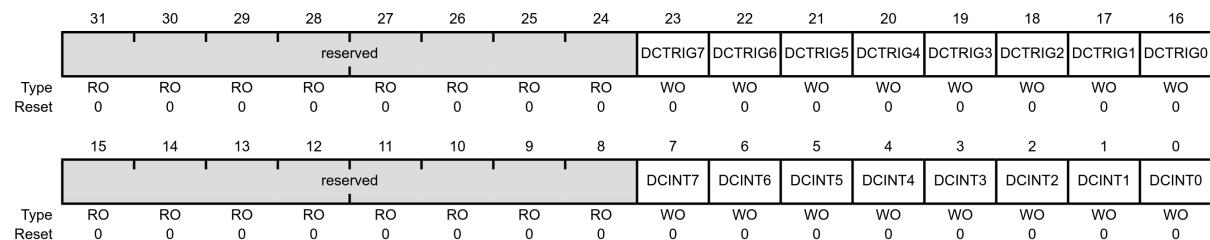
20.9.17 ADC Digital Comparator Reset Initial Conditions (ADCDCRIC)

Resetting initial conditions of digital comparators.

- Reset of triggers
- Reset of interrupts

Documentation:

- ADC0 base: 0x4003.8000
- ADC1 base: 0x4003.9000
- Offset 0xD00
- Type WO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description						
31:24	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
23	DCTRIG7	WO	0	<p>Digital Comparator Trigger 7</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No effect.</td> </tr> <tr> <td>1</td> <td>Resets the Digital Comparator 7 trigger unit to its initial conditions.</td> </tr> </tbody> </table> <p>When the trigger has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used. After setting this bit, software should wait until the bit clears before continuing.</p>	Value	Description	0	No effect.	1	Resets the Digital Comparator 7 trigger unit to its initial conditions.
Value	Description									
0	No effect.									
1	Resets the Digital Comparator 7 trigger unit to its initial conditions.									

Bit/Field	Name	Type	Reset	Description						
22	DCTRIG6	WO	0	<p>Digital Comparator Trigger 6</p> <table> <thead> <tr> <th data-bbox="684 309 774 343">Value</th> <th data-bbox="774 309 938 343">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="684 354 716 388">0</td> <td data-bbox="774 354 901 388">No effect.</td> </tr> <tr> <td data-bbox="684 399 716 433">1</td> <td data-bbox="774 399 1346 473">Resets the Digital Comparator 6 trigger unit to its initial conditions.</td> </tr> </tbody> </table> <p>When the trigger has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.</p>	Value	Description	0	No effect.	1	Resets the Digital Comparator 6 trigger unit to its initial conditions.
Value	Description									
0	No effect.									
1	Resets the Digital Comparator 6 trigger unit to its initial conditions.									
21	DCTRIG5	WO	0	<p>Digital Comparator Trigger 5</p> <table> <thead> <tr> <th data-bbox="684 804 774 837">Value</th> <th data-bbox="774 804 938 837">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="684 848 716 882">0</td> <td data-bbox="774 848 901 882">No effect.</td> </tr> <tr> <td data-bbox="684 893 716 927">1</td> <td data-bbox="774 893 1346 968">Resets the Digital Comparator 5 trigger unit to its initial conditions.</td> </tr> </tbody> </table> <p>When the trigger has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.</p>	Value	Description	0	No effect.	1	Resets the Digital Comparator 5 trigger unit to its initial conditions.
Value	Description									
0	No effect.									
1	Resets the Digital Comparator 5 trigger unit to its initial conditions.									
20	DCTRIG4	WO	0	<p>Digital Comparator Trigger 4</p> <table> <thead> <tr> <th data-bbox="684 1320 774 1354">Value</th> <th data-bbox="774 1320 938 1354">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="684 1365 716 1399">0</td> <td data-bbox="774 1365 901 1399">No effect.</td> </tr> <tr> <td data-bbox="684 1410 716 1444">1</td> <td data-bbox="774 1410 1346 1484">Resets the Digital Comparator 4 trigger unit to its initial conditions.</td> </tr> </tbody> </table> <p>When the trigger has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.</p>	Value	Description	0	No effect.	1	Resets the Digital Comparator 4 trigger unit to its initial conditions.
Value	Description									
0	No effect.									
1	Resets the Digital Comparator 4 trigger unit to its initial conditions.									

Bit/Field	Name	Type	Reset	Description						
19	DCTRIG3	WO	0	<p>Digital Comparator Trigger 3</p> <table> <thead> <tr> <th data-bbox="684 309 774 343">Value</th> <th data-bbox="774 309 938 343">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="684 354 716 388">0</td> <td data-bbox="774 354 901 388">No effect.</td> </tr> <tr> <td data-bbox="684 399 716 433">1</td> <td data-bbox="774 399 1346 473">Resets the Digital Comparator 3 trigger unit to its initial conditions.</td> </tr> </tbody> </table> <p>When the trigger has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.</p>	Value	Description	0	No effect.	1	Resets the Digital Comparator 3 trigger unit to its initial conditions.
Value	Description									
0	No effect.									
1	Resets the Digital Comparator 3 trigger unit to its initial conditions.									
18	DCTRIG2	WO	0	<p>Digital Comparator Trigger 2</p> <table> <thead> <tr> <th data-bbox="684 804 774 837">Value</th> <th data-bbox="774 804 938 837">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="684 848 716 882">0</td> <td data-bbox="774 848 901 882">No effect.</td> </tr> <tr> <td data-bbox="684 893 716 927">1</td> <td data-bbox="774 893 1346 968">Resets the Digital Comparator 2 trigger unit to its initial conditions.</td> </tr> </tbody> </table> <p>When the trigger has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.</p>	Value	Description	0	No effect.	1	Resets the Digital Comparator 2 trigger unit to its initial conditions.
Value	Description									
0	No effect.									
1	Resets the Digital Comparator 2 trigger unit to its initial conditions.									
17	DCTRIG1	WO	0	<p>Digital Comparator Trigger 1</p> <table> <thead> <tr> <th data-bbox="684 1320 774 1354">Value</th> <th data-bbox="774 1320 938 1354">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="684 1365 716 1399">0</td> <td data-bbox="774 1365 901 1399">No effect.</td> </tr> <tr> <td data-bbox="684 1410 716 1444">1</td> <td data-bbox="774 1410 1346 1484">Resets the Digital Comparator 1 trigger unit to its initial conditions.</td> </tr> </tbody> </table> <p>When the trigger has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.</p>	Value	Description	0	No effect.	1	Resets the Digital Comparator 1 trigger unit to its initial conditions.
Value	Description									
0	No effect.									
1	Resets the Digital Comparator 1 trigger unit to its initial conditions.									

Bit/Field	Name	Type	Reset	Description						
16	DCTRIG0	WO	0	<p>Digital Comparator Trigger 0</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No effect.</td> </tr> <tr> <td>1</td> <td>Resets the Digital Comparator 0 trigger unit to its initial conditions.</td> </tr> </tbody> </table> <p>When the trigger has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.</p>	Value	Description	0	No effect.	1	Resets the Digital Comparator 0 trigger unit to its initial conditions.
Value	Description									
0	No effect.									
1	Resets the Digital Comparator 0 trigger unit to its initial conditions.									
15:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
7	DCINT7	WO	0	<p>Digital Comparator Interrupt 7</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No effect.</td> </tr> <tr> <td>1</td> <td>Resets the Digital Comparator 7 interrupt unit to its initial conditions.</td> </tr> </tbody> </table> <p>When the interrupt has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.</p>	Value	Description	0	No effect.	1	Resets the Digital Comparator 7 interrupt unit to its initial conditions.
Value	Description									
0	No effect.									
1	Resets the Digital Comparator 7 interrupt unit to its initial conditions.									
6	DCINT6	WO	0	<p>Digital Comparator Interrupt 6</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No effect.</td> </tr> <tr> <td>1</td> <td>Resets the Digital Comparator 6 interrupt unit to its initial conditions.</td> </tr> </tbody> </table> <p>When the interrupt has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.</p>	Value	Description	0	No effect.	1	Resets the Digital Comparator 6 interrupt unit to its initial conditions.
Value	Description									
0	No effect.									
1	Resets the Digital Comparator 6 interrupt unit to its initial conditions.									

Bit/Field	Name	Type	Reset	Description						
5	DCINT5	WO	0	<p>Digital Comparator Interrupt 5</p> <table> <thead> <tr> <th data-bbox="684 309 768 343">Value</th> <th data-bbox="768 309 938 343">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="684 354 716 388">0</td> <td data-bbox="768 354 901 388">No effect.</td> </tr> <tr> <td data-bbox="684 399 716 433">1</td> <td data-bbox="768 399 1372 473">Resets the Digital Comparator 5 interrupt unit to its initial conditions.</td> </tr> </tbody> </table> <p>When the interrupt has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.</p>	Value	Description	0	No effect.	1	Resets the Digital Comparator 5 interrupt unit to its initial conditions.
Value	Description									
0	No effect.									
1	Resets the Digital Comparator 5 interrupt unit to its initial conditions.									
4	DCINT4	WO	0	<p>Digital Comparator Interrupt 4</p> <table> <thead> <tr> <th data-bbox="684 810 768 844">Value</th> <th data-bbox="768 810 938 844">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="684 855 716 889">0</td> <td data-bbox="768 855 901 889">No effect.</td> </tr> <tr> <td data-bbox="684 900 716 934">1</td> <td data-bbox="768 900 1372 974">Resets the Digital Comparator 4 interrupt unit to its initial conditions.</td> </tr> </tbody> </table> <p>When the interrupt has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.</p>	Value	Description	0	No effect.	1	Resets the Digital Comparator 4 interrupt unit to its initial conditions.
Value	Description									
0	No effect.									
1	Resets the Digital Comparator 4 interrupt unit to its initial conditions.									
3	DCINT3	WO	0	<p>Digital Comparator Interrupt 3</p> <table> <thead> <tr> <th data-bbox="684 1311 768 1345">Value</th> <th data-bbox="768 1311 938 1345">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="684 1356 716 1390">0</td> <td data-bbox="768 1356 901 1390">No effect.</td> </tr> <tr> <td data-bbox="684 1401 716 1435">1</td> <td data-bbox="768 1401 1372 1475">Resets the Digital Comparator 3 interrupt unit to its initial conditions.</td> </tr> </tbody> </table> <p>When the interrupt has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.</p>	Value	Description	0	No effect.	1	Resets the Digital Comparator 3 interrupt unit to its initial conditions.
Value	Description									
0	No effect.									
1	Resets the Digital Comparator 3 interrupt unit to its initial conditions.									

Bit/Field	Name	Type	Reset	Description						
2	DCINT2	WO	0	<p>Digital Comparator Interrupt 2</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No effect.</td> </tr> <tr> <td>1</td> <td>Resets the Digital Comparator 2 interrupt unit to its initial conditions.</td> </tr> </tbody> </table> <p>When the interrupt has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.</p>	Value	Description	0	No effect.	1	Resets the Digital Comparator 2 interrupt unit to its initial conditions.
Value	Description									
0	No effect.									
1	Resets the Digital Comparator 2 interrupt unit to its initial conditions.									
1	DCINT1	WO	0	<p>Digital Comparator Interrupt 1</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No effect.</td> </tr> <tr> <td>1</td> <td>Resets the Digital Comparator 1 interrupt unit to its initial conditions.</td> </tr> </tbody> </table> <p>When the interrupt has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.</p>	Value	Description	0	No effect.	1	Resets the Digital Comparator 1 interrupt unit to its initial conditions.
Value	Description									
0	No effect.									
1	Resets the Digital Comparator 1 interrupt unit to its initial conditions.									
0	DCINT0	WO	0	<p>Digital Comparator Interrupt 0</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No effect.</td> </tr> <tr> <td>1</td> <td>Resets the Digital Comparator 0 interrupt unit to its initial conditions.</td> </tr> </tbody> </table> <p>When the interrupt has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.</p>	Value	Description	0	No effect.	1	Resets the Digital Comparator 0 interrupt unit to its initial conditions.
Value	Description									
0	No effect.									
1	Resets the Digital Comparator 0 interrupt unit to its initial conditions.									

20.9.18 ADC Sample Sequence 0 Digital Comparator Select (ADCSSDC0)

This register determines which digital comparator receives the sample from the given conversion on Sample Sequence 0, if the corresponding SnDCOP bit in the **ADCSSOP0** (page 212) register is set.

- Sample sequencer can take up to eight samples.
- Each sample's digital value can be sent to a digital comparator.
- This register assigns a digital comparator to a sample.

Documentation:

- ADC0 base: 0x4003.8000
- ADC1 base: 0x4003.9000
- Offset 0x054
- Type RW, reset 0x0000.0000

Bit/Field	Name	Type	Reset	Description																				
31:28	S7DCSEL	RW	0x0	<p>Sample 7 Digital Comparator Select</p> <p>When the S7DCOP bit in the ADCSSOP0 (page 212) register is set, this field indicates which digital comparator unit (and its associated set of control registers) receives the eighth sample from Sample Sequencer 0.</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Digital Comparator Unit 0 (ADCDCMP0 and ADCDCCTL0)</td> </tr> <tr> <td>0x1</td> <td>Digital Comparator Unit 1 (ADCDCMP1 and ADCDCCTL1)</td> </tr> <tr> <td>0x2</td> <td>Digital Comparator Unit 2 (ADCDCMP2 and ADCDCCTL2)</td> </tr> <tr> <td>0x3</td> <td>Digital Comparator Unit 3 (ADCDCMP3 and ADCDCCTL3)</td> </tr> <tr> <td>0x4</td> <td>Digital Comparator Unit 4 (ADCDCMP4 and ADCDCCTL4)</td> </tr> <tr> <td>0x5</td> <td>Digital Comparator Unit 5 (ADCDCMP5 and ADCDCCTL5)</td> </tr> <tr> <td>0x6</td> <td>Digital Comparator Unit 6 (ADCDCMP6 and ADCDCCTL6)</td> </tr> <tr> <td>0x7</td> <td>Digital Comparator Unit 7 (ADCDCMP7 and ADCDCCTL7)</td> </tr> <tr> <td>0x8 - 0xF</td> <td>Reserved</td> </tr> </tbody> </table>	Value	Description	0x0	Digital Comparator Unit 0 (ADCDCMP0 and ADCDCCTL0)	0x1	Digital Comparator Unit 1 (ADCDCMP1 and ADCDCCTL1)	0x2	Digital Comparator Unit 2 (ADCDCMP2 and ADCDCCTL2)	0x3	Digital Comparator Unit 3 (ADCDCMP3 and ADCDCCTL3)	0x4	Digital Comparator Unit 4 (ADCDCMP4 and ADCDCCTL4)	0x5	Digital Comparator Unit 5 (ADCDCMP5 and ADCDCCTL5)	0x6	Digital Comparator Unit 6 (ADCDCMP6 and ADCDCCTL6)	0x7	Digital Comparator Unit 7 (ADCDCMP7 and ADCDCCTL7)	0x8 - 0xF	Reserved
Value	Description																							
0x0	Digital Comparator Unit 0 (ADCDCMP0 and ADCDCCTL0)																							
0x1	Digital Comparator Unit 1 (ADCDCMP1 and ADCDCCTL1)																							
0x2	Digital Comparator Unit 2 (ADCDCMP2 and ADCDCCTL2)																							
0x3	Digital Comparator Unit 3 (ADCDCMP3 and ADCDCCTL3)																							
0x4	Digital Comparator Unit 4 (ADCDCMP4 and ADCDCCTL4)																							
0x5	Digital Comparator Unit 5 (ADCDCMP5 and ADCDCCTL5)																							
0x6	Digital Comparator Unit 6 (ADCDCMP6 and ADCDCCTL6)																							
0x7	Digital Comparator Unit 7 (ADCDCMP7 and ADCDCCTL7)																							
0x8 - 0xF	Reserved																							
27:24	S6DCSEL	RW	0x0	This field has the same encodings as S7DCSEL but is used during the seventh sample.																				
23:20	S5DCSEL	RW	0x0	This field has the same encodings as S7DCSEL but is used during the sixth sample.																				
19:16	S4DCSEL	RW	0x0	This field has the same encodings as S7DCSEL but is used during the fifth sample.																				

Bit/Field	Name	Type	Reset	Description
15:12	S3DCSEL	RW	0x0	This field has the same encodings as S7DCSEL but is used during the fourth sample.
11:8	S2DCSEL	RW	0x0	This field has the same encodings as S7DCSEL but is used during the third sample.
7:4	S1DCSEL	RW	0x0	This field has the same encodings as S7DCSEL but is used during the second sample.
3:0	S0DCSEL	RW	0x0	This field has the same encodings as S7DCSEL but is used during the first sample.

20.9.19 ADC Sample Sequence 0 Operation (ADCSSOP0)

This register determines whether the sample from the given conversion on Sample Sequence 0 is saved in the Sample Sequence FIFO0 or sent to the digital comparator unit.

Documentation:

- ADC0 base: 0x4003.8000
- ADC1 base: 0x4003.9000
- Offset 0x050
- Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved			S7DCOP	reserved			S6DCOP	reserved			S5DCOP	reserved			S4DCOP
Type	RO	RO	RO	RW												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved			S3DCOP	reserved			S2DCOP	reserved			S1DCOP	reserved			S0DCOP
Type	RO	RO	RO	RW												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description						
31:29	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
28	S7DCOP	RW	0	<p>Sample 7 Digital Comparator Operation</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The eighth sample is saved in Sample Sequence FIFO0.</td> </tr> <tr> <td>1</td> <td>The eighth sample is sent to the digital comparator unit specified by the S7DCSEL bit in the ADCSSDC0 (page 210) register, and the value is not written to the FIFO.</td> </tr> </tbody> </table>	Value	Description	0	The eighth sample is saved in Sample Sequence FIFO0.	1	The eighth sample is sent to the digital comparator unit specified by the S7DCSEL bit in the ADCSSDC0 (page 210) register, and the value is not written to the FIFO.
Value	Description									
0	The eighth sample is saved in Sample Sequence FIFO0.									
1	The eighth sample is sent to the digital comparator unit specified by the S7DCSEL bit in the ADCSSDC0 (page 210) register, and the value is not written to the FIFO.									
27:25	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
24	S6DCOP	RW	0	<p>Sample 6 Digital Comparator Operation</p> <p>Same definition as S7DCOP but used during the seventh sample.</p>						
23:21	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
20	S5DCOP	RW	0	<p>Sample 5 Digital Comparator Operation</p> <p>Same definition as S7DCOP but used during the sixth sample.</p>						

Bit/Field	Name	Type	Reset	Description
19:17	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	S4DCOP	RW	0	Sample 4 Digital Comparator Operation Same definition as S7DCOP but used during the fifth sample.
15:13	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	S3DCOP	RW	0	Sample 3 Digital Comparator Operation Same definition as S7DCOP but used during the fourth sample.
11:9	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	S2DCOP	RW	0	Sample 2 Digital Comparator Operation Same definition as S7DCOP but used during the third sample.
7:5	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	S1DCOP	RW	0	Sample 1 Digital Comparator Operation Same definition as S7DCOP but used during the second sample.
3:1	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	S0DCOP	RW	0	Sample 0 Digital Comparator Operation Same definition as S7DCOP but used during the first sample.

20.9.20 ADC Processor Sample Sequence Initiate (ADCPSSI)

- ADC0 base: 0x4003.8000
- ADC1 base: 0x4003.9000
- Offset 0x028
- Type RW, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	GSYNC	reserved			SYNCWAIT	reserved										
Type	RW	RO	RO	RO	RW	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved										SS3	SS2	SS1	SS0		
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	WO	WO	WO	WO	WO	WO
Reset	0	0	0	0	0	0	0	0	0	0	-	-	-	-	-	-

Bit/Field	Name	Type	Reset	Description						
31	GSYNC	RW	0	<p>Global Synchronize</p> <table> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>This bit is cleared once sampling has been initiated.</td> </tr> <tr> <td>1</td> <td>This bit initiates sampling in multiple ADC modules at the same time. Any ADC module that has been initialized by setting an SS_n bit and the SYNCWAIT bit starts sampling once this bit is written.</td> </tr> </table>	Value	Description	0	This bit is cleared once sampling has been initiated.	1	This bit initiates sampling in multiple ADC modules at the same time. Any ADC module that has been initialized by setting an SS _n bit and the SYNCWAIT bit starts sampling once this bit is written.
Value	Description									
0	This bit is cleared once sampling has been initiated.									
1	This bit initiates sampling in multiple ADC modules at the same time. Any ADC module that has been initialized by setting an SS _n bit and the SYNCWAIT bit starts sampling once this bit is written.									
30:28	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
27	SYNCWAIT	RW	0	<p>Synchronize Wait</p> <table> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>Sampling begins when a sample sequence has been initiated.</td> </tr> <tr> <td>1</td> <td>This bit allows the sample sequences to be initiated, but delays sampling until the GSYNC bit is set.</td> </tr> </table>	Value	Description	0	Sampling begins when a sample sequence has been initiated.	1	This bit allows the sample sequences to be initiated, but delays sampling until the GSYNC bit is set.
Value	Description									
0	Sampling begins when a sample sequence has been initiated.									
1	This bit allows the sample sequences to be initiated, but delays sampling until the GSYNC bit is set.									
26:4	reserved	RO	0x000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						

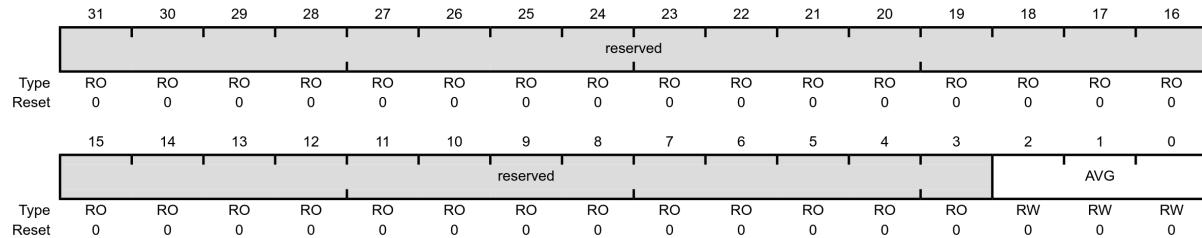
Bit/Field	Name	Type	Reset	Description						
3	SS3	WO	-	<p>SS3 Initiate</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No effect.</td> </tr> <tr> <td>1</td> <td>Begin sampling on Sample Sequencer 3, if the sequencer is enabled in the ADCACTSS register.</td> </tr> </tbody> </table> <p>Only a write by software is valid; a read of this register returns no meaningful data.</p>	Value	Description	0	No effect.	1	Begin sampling on Sample Sequencer 3, if the sequencer is enabled in the ADCACTSS register.
Value	Description									
0	No effect.									
1	Begin sampling on Sample Sequencer 3, if the sequencer is enabled in the ADCACTSS register.									
2	SS2	WO	-	<p>SS2 Initiate</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No effect.</td> </tr> <tr> <td>1</td> <td>Begin sampling on Sample Sequencer 2, if the sequencer is enabled in the ADCACTSS register.</td> </tr> </tbody> </table> <p>Only a write by software is valid; a read of this register returns no meaningful data.</p>	Value	Description	0	No effect.	1	Begin sampling on Sample Sequencer 2, if the sequencer is enabled in the ADCACTSS register.
Value	Description									
0	No effect.									
1	Begin sampling on Sample Sequencer 2, if the sequencer is enabled in the ADCACTSS register.									
1	SS1	WO	-	<p>SS1 Initiate</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No effect.</td> </tr> <tr> <td>1</td> <td>Begin sampling on Sample Sequencer 1, if the sequencer is enabled in the ADCACTSS register.</td> </tr> </tbody> </table> <p>Only a write by software is valid; a read of this register returns no meaningful data.</p>	Value	Description	0	No effect.	1	Begin sampling on Sample Sequencer 1, if the sequencer is enabled in the ADCACTSS register.
Value	Description									
0	No effect.									
1	Begin sampling on Sample Sequencer 1, if the sequencer is enabled in the ADCACTSS register.									
0	SS0	WO	-	<p>SS0 Initiate</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No effect.</td> </tr> <tr> <td>1</td> <td>Begin sampling on Sample Sequencer 0, if the sequencer is enabled in the ADCACTSS register.</td> </tr> </tbody> </table> <p>Only a write by software is valid; a read of this register returns no meaningful data.</p>	Value	Description	0	No effect.	1	Begin sampling on Sample Sequencer 0, if the sequencer is enabled in the ADCACTSS register.
Value	Description									
0	No effect.									
1	Begin sampling on Sample Sequencer 0, if the sequencer is enabled in the ADCACTSS register.									

20.9.21 ADC Sample Averaging Control (ADCSAC)

This register controls the amount of hardware averaging applied to conversion results.

Documentation:

- ADC0 base: 0x4003.8000
- ADC1 base: 0x4003.9000
- Offset 0x030
- Type RW, reset 0x0000.0000



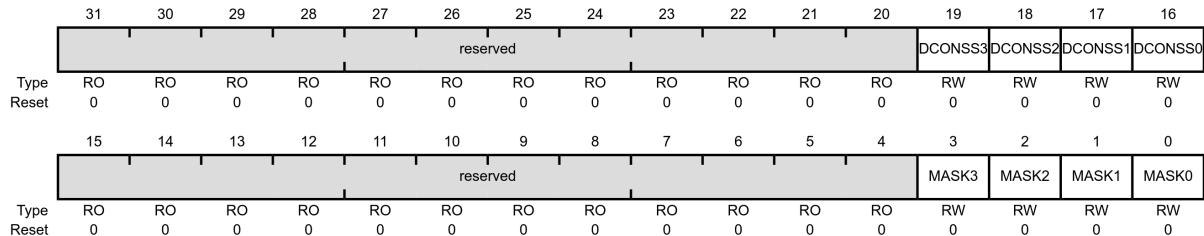
Bit/Field	Name	Type	Reset	Description																		
31:3	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.																		
2:0	AVG	RW	0x0	<p>Hardware Averaging Control</p> <p>Specifies the amount of hardware averaging that will be applied to ADC samples. The AVG field can be any value between 0 and 6. Entering a value of 7 creates unpredictable results.</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>No hardware oversampling</td> </tr> <tr> <td>0x1</td> <td>2x hardware oversampling</td> </tr> <tr> <td>0x2</td> <td>4x hardware oversampling</td> </tr> <tr> <td>0x3</td> <td>8x hardware oversampling</td> </tr> <tr> <td>0x4</td> <td>16x hardware oversampling</td> </tr> <tr> <td>0x5</td> <td>32x hardware oversampling</td> </tr> <tr> <td>0x6</td> <td>64x hardware oversampling</td> </tr> <tr> <td>0x7</td> <td>Reserved</td> </tr> </tbody> </table>	Value	Description	0x0	No hardware oversampling	0x1	2x hardware oversampling	0x2	4x hardware oversampling	0x3	8x hardware oversampling	0x4	16x hardware oversampling	0x5	32x hardware oversampling	0x6	64x hardware oversampling	0x7	Reserved
Value	Description																					
0x0	No hardware oversampling																					
0x1	2x hardware oversampling																					
0x2	4x hardware oversampling																					
0x3	8x hardware oversampling																					
0x4	16x hardware oversampling																					
0x5	32x hardware oversampling																					
0x6	64x hardware oversampling																					
0x7	Reserved																					

20.9.22 ADC Interrupt Mask (ADCIM)

This register controls whether the sample sequencer and digital comparator raw interrupt signals are sent to the interrupt controller. Each raw interrupt signal can be masked independently.

Documentation:

- ADC0 base: 0x4003.8000
- ADC1 base: 0x4003.9000
- Offset 0x008
- Type RW, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description							
31:20	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.							
19	DCONSS3	RW	0	Digital Comparator Interrupt on SS3 <table> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>The status of the digital comparators does not affect the SS3 interrupt status.</td> </tr> <tr> <td>1</td> <td>The raw interrupt signal from the digital comparators (INRDC bit in the ADCRIS (page 222) register) is sent to the interrupt controller on the SS3 interrupt line.</td> </tr> </table>		Value	Description	0	The status of the digital comparators does not affect the SS3 interrupt status.	1	The raw interrupt signal from the digital comparators (INRDC bit in the ADCRIS (page 222) register) is sent to the interrupt controller on the SS3 interrupt line.
Value	Description										
0	The status of the digital comparators does not affect the SS3 interrupt status.										
1	The raw interrupt signal from the digital comparators (INRDC bit in the ADCRIS (page 222) register) is sent to the interrupt controller on the SS3 interrupt line.										
18	DCONSS2	RW	0	Digital Comparator Interrupt on SS2 <table> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>The status of the digital comparators does not affect the SS2 interrupt status.</td> </tr> <tr> <td>1</td> <td>The raw interrupt signal from the digital comparators (INRDC bit in the ADCRIS (page 222) register) is sent to the interrupt controller on the SS2 interrupt line.</td> </tr> </table>		Value	Description	0	The status of the digital comparators does not affect the SS2 interrupt status.	1	The raw interrupt signal from the digital comparators (INRDC bit in the ADCRIS (page 222) register) is sent to the interrupt controller on the SS2 interrupt line.
Value	Description										
0	The status of the digital comparators does not affect the SS2 interrupt status.										
1	The raw interrupt signal from the digital comparators (INRDC bit in the ADCRIS (page 222) register) is sent to the interrupt controller on the SS2 interrupt line.										

Bit/Field	Name	Type	Reset	Description						
17	DCONSS1	RW	0	<p>Digital Comparator Interrupt on SS1</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The status of the digital comparators does not affect the SS1 interrupt status.</td></tr> <tr> <td>1</td><td>The raw interrupt signal from the digital comparators (INRDC bit in the ADCRIS (page 222) register) is sent to the interrupt controller on the SS1 interrupt line.</td></tr> </tbody> </table>	Value	Description	0	The status of the digital comparators does not affect the SS1 interrupt status.	1	The raw interrupt signal from the digital comparators (INRDC bit in the ADCRIS (page 222) register) is sent to the interrupt controller on the SS1 interrupt line.
Value	Description									
0	The status of the digital comparators does not affect the SS1 interrupt status.									
1	The raw interrupt signal from the digital comparators (INRDC bit in the ADCRIS (page 222) register) is sent to the interrupt controller on the SS1 interrupt line.									
16	DCONSS0	RW	0	<p>Digital Comparator Interrupt on SS0</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The status of the digital comparators does not affect the SS0 interrupt status.</td></tr> <tr> <td>1</td><td>The raw interrupt signal from the digital comparators (INRDC bit in the ADCRIS (page 222) register) is sent to the interrupt controller on the SS0 interrupt line.</td></tr> </tbody> </table>	Value	Description	0	The status of the digital comparators does not affect the SS0 interrupt status.	1	The raw interrupt signal from the digital comparators (INRDC bit in the ADCRIS (page 222) register) is sent to the interrupt controller on the SS0 interrupt line.
Value	Description									
0	The status of the digital comparators does not affect the SS0 interrupt status.									
1	The raw interrupt signal from the digital comparators (INRDC bit in the ADCRIS (page 222) register) is sent to the interrupt controller on the SS0 interrupt line.									
15:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
3	MASK3	RW	0	<p>SS3 Interrupt Mask</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The status of Sample Sequencer 3 does not affect the SS3 interrupt status.</td></tr> <tr> <td>1</td><td>The raw interrupt signal from Sample Sequencer 3 (ADCRIS (page 222) register INR3 bit) is sent to the interrupt controller.</td></tr> </tbody> </table>	Value	Description	0	The status of Sample Sequencer 3 does not affect the SS3 interrupt status.	1	The raw interrupt signal from Sample Sequencer 3 (ADCRIS (page 222) register INR3 bit) is sent to the interrupt controller.
Value	Description									
0	The status of Sample Sequencer 3 does not affect the SS3 interrupt status.									
1	The raw interrupt signal from Sample Sequencer 3 (ADCRIS (page 222) register INR3 bit) is sent to the interrupt controller.									
2	MASK2	RW	0	<p>SS2 Interrupt Mask</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The status of Sample Sequencer 2 does not affect the SS2 interrupt status.</td></tr> <tr> <td>1</td><td>The raw interrupt signal from Sample Sequencer 2 (ADCRIS (page 222) register INR2 bit) is sent to the interrupt controller.</td></tr> </tbody> </table>	Value	Description	0	The status of Sample Sequencer 2 does not affect the SS2 interrupt status.	1	The raw interrupt signal from Sample Sequencer 2 (ADCRIS (page 222) register INR2 bit) is sent to the interrupt controller.
Value	Description									
0	The status of Sample Sequencer 2 does not affect the SS2 interrupt status.									
1	The raw interrupt signal from Sample Sequencer 2 (ADCRIS (page 222) register INR2 bit) is sent to the interrupt controller.									

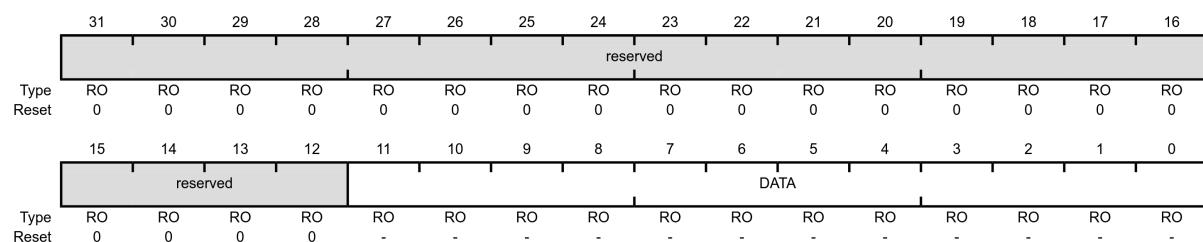
Bit/Field	Name	Type	Reset	Description						
1	MASK1	RW	0	<p>SS1 Interrupt Mask</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The status of Sample Sequencer 1 does not affect the SS1 interrupt status.</td> </tr> <tr> <td>1</td> <td>The raw interrupt signal from Sample Sequencer 1 (ADCRIS (page 222) register INR1 bit) is sent to the interrupt controller.</td> </tr> </tbody> </table>	Value	Description	0	The status of Sample Sequencer 1 does not affect the SS1 interrupt status.	1	The raw interrupt signal from Sample Sequencer 1 (ADCRIS (page 222) register INR1 bit) is sent to the interrupt controller.
Value	Description									
0	The status of Sample Sequencer 1 does not affect the SS1 interrupt status.									
1	The raw interrupt signal from Sample Sequencer 1 (ADCRIS (page 222) register INR1 bit) is sent to the interrupt controller.									
0	MASK0	RW	0	<p>SS0 Interrupt Mask</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The status of Sample Sequencer 0 does not affect the SS0 interrupt status.</td> </tr> <tr> <td>1</td> <td>The raw interrupt signal from Sample Sequencer 0 (ADCRIS (page 222) register INR0 bit) is sent to the interrupt controller.</td> </tr> </tbody> </table>	Value	Description	0	The status of Sample Sequencer 0 does not affect the SS0 interrupt status.	1	The raw interrupt signal from Sample Sequencer 0 (ADCRIS (page 222) register INR0 bit) is sent to the interrupt controller.
Value	Description									
0	The status of Sample Sequencer 0 does not affect the SS0 interrupt status.									
1	The raw interrupt signal from Sample Sequencer 0 (ADCRIS (page 222) register INR0 bit) is sent to the interrupt controller.									

20.9.23 ADC Sample Sequence Result FIFO n (ADCSSFIFOOn)

- ADC0 base: 0x4003.8000
- ADC1 base: 0x4003.9000
- Offset 0x048
- Type RO, reset -

Registers:

- Register 17: ADC Sample Sequence Result FIFO 0 (ADCSSFIFO0), offset 0x048
- Register 18: ADC Sample Sequence Result FIFO 1 (ADCSSFIFO1), offset 0x068
- Register 19: ADC Sample Sequence Result FIFO 2 (ADCSSFIFO2), offset 0x088
- Register 20: ADC Sample Sequence Result FIFO 3 (ADCSSFIFO3), offset 0x0A8



Bit/Field	Name	Type	Reset	Description
31:12	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11:0	DATA	RO	-	Conversion Result Data

20.9.24 ADC Sample Sequence FIFO n Status (ADCSSFSTATn)

FIFO stands for first in, first out queue.

- ADC0 base: 0x4003.8000
- ADC1 base: 0x4003.9000
- Offset 0x04C
- Type RO, reset 0x0000.0100

Registers:

- Register 21: ADC Sample Sequence FIFO 0 Status (ADCSSFSTAT0), offset 0x04C
- Register 22: ADC Sample Sequence FIFO 1 Status (ADCSSFSTAT1), offset 0x06C
- Register 23: ADC Sample Sequence FIFO 2 Status (ADCSSFSTAT2), offset 0x08C
- Register 24: ADC Sample Sequence FIFO 3 Status (ADCSSFSTAT3), offset 0x0AC

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved				FULL		reserved			EMPTY		HPTR			TPTR	
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description						
31:13	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
12	FULL	RO	0	<p>FIFO Full</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The FIFO is not currently full.</td> </tr> <tr> <td>1</td> <td>The FIFO is currently full.</td> </tr> </tbody> </table>	Value	Description	0	The FIFO is not currently full.	1	The FIFO is currently full.
Value	Description									
0	The FIFO is not currently full.									
1	The FIFO is currently full.									
11:9	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
8	EMPTY	RO	1	<p>FIFO Empty</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The FIFO is not currently empty.</td> </tr> <tr> <td>1</td> <td>The FIFO is currently empty.</td> </tr> </tbody> </table>	Value	Description	0	The FIFO is not currently empty.	1	The FIFO is currently empty.
Value	Description									
0	The FIFO is not currently empty.									
1	The FIFO is currently empty.									
7:4	HPTR	RO	0x0	<p>FIFO Head Pointer</p> <p>This field contains the current “head” pointer index for the FIFO, that is, the next entry to be written.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> • 0x0 - 0x7 for FIFO0 • 0x0 - 0x3 for FIFO1 and FIFO2 • 0x0 for FIFO3. 						

Bit/Field	Name	Type	Reset	Description
3:0	TPTR	RO	0x0	<p>FIFO Tail Pointer</p> <p>This field contains the current “tail” pointer index for the FIFO, that is, the next entry to be read.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> • 0x0 - 0x7 for FIFO0 • 0x0 - 0x3 for FIFO1 and FIFO2 • 0x0 for FIFO3

20.9.24.1 Illustration of the head and tail address

Tail Address →



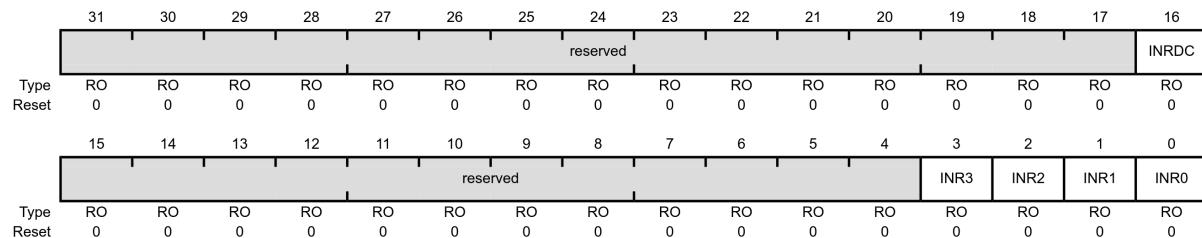
Head Address →

20.9.25 ADC Raw Interrupt Status (ADCRIS)

This register shows the status of the raw interrupt signal of each sample sequencer. These bits may be polled by software to look for interrupt conditions without sending the interrupts to the interrupt controller.

Documentation:

- ADC0 base: 0x4003.8000
- ADC1 base: 0x4003.9000
- Offset 0x004
- Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:17	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	INRDC	RO	0	Digital Comparator Raw Interrupt Status Value Description 0 All bits in the ADCDISC register are clear. 1 At least one bit in the ADCDISC register is set, meaning that a digital comparator interrupt has occurred.
15:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	INR3	RO	0	SS3 Raw Interrupt Status Value Description 0 An interrupt has not occurred. 1 A sample has completed conversion and the respective ADCSSCTL3 (page 198) IEn bit is set, enabling a raw interrupt. This bit is cleared by writing a 1 to the IN3 bit in the ADCISC register.

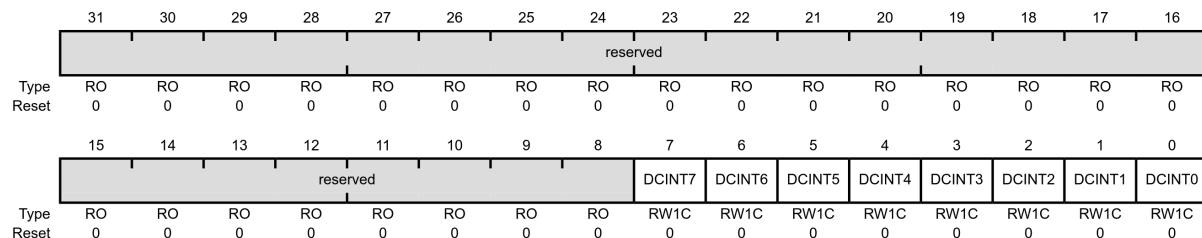
Bit/Field	Name	Type	Reset	Description						
2	INR2	RO	0	<p>SS2 Raw Interrupt Status</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>An interrupt has not occurred.</td> </tr> <tr> <td>1</td> <td>A sample has completed conversion and the respective ADCSSCTL2 IEn bit is set, enabling a raw interrupt.</td> </tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the IN2 bit in the ADCISC register.</p>	Value	Description	0	An interrupt has not occurred.	1	A sample has completed conversion and the respective ADCSSCTL2 IEn bit is set, enabling a raw interrupt.
Value	Description									
0	An interrupt has not occurred.									
1	A sample has completed conversion and the respective ADCSSCTL2 IEn bit is set, enabling a raw interrupt.									
1	INR1	RO	0	<p>SS1 Raw Interrupt Status</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>An interrupt has not occurred.</td> </tr> <tr> <td>1</td> <td>A sample has completed conversion and the respective ADCSSCTL1 IEn bit is set, enabling a raw interrupt.</td> </tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the IN1 bit in the ADCISC register.</p>	Value	Description	0	An interrupt has not occurred.	1	A sample has completed conversion and the respective ADCSSCTL1 IEn bit is set, enabling a raw interrupt.
Value	Description									
0	An interrupt has not occurred.									
1	A sample has completed conversion and the respective ADCSSCTL1 IEn bit is set, enabling a raw interrupt.									
0	INR0	RO	0	<p>SS0 Raw Interrupt Status</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>An interrupt has not occurred.</td> </tr> <tr> <td>1</td> <td>A sample has completed conversion and the respective ADCSSCTL0 IEn bit is set, enabling a raw interrupt.</td> </tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the IN0 bit in the ADCISC register.</p>	Value	Description	0	An interrupt has not occurred.	1	A sample has completed conversion and the respective ADCSSCTL0 IEn bit is set, enabling a raw interrupt.
Value	Description									
0	An interrupt has not occurred.									
1	A sample has completed conversion and the respective ADCSSCTL0 IEn bit is set, enabling a raw interrupt.									

20.9.26 ADC Digital Comparator Interrupt Status and Clear (ADCDCISC)

This register provides status and acknowledgement of digital comparator interrupts. One bit is provided for each comparator.

Documentation:

- ADC0 base: 0x4003.8000
- ADC1 base: 0x4003.9000
- Offset 0x034
- Type RW1C, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description						
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
7	DCINT7	RW1C	0	<p>Digital Comparator 7 Interrupt Status and Clear</p> <table> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>No interrupt.</td> </tr> <tr> <td>1</td> <td>Digital Comparator 7 has generated an interrupt.</td> </tr> </table> <p>This bit is cleared by writing a 1.</p>	Value	Description	0	No interrupt.	1	Digital Comparator 7 has generated an interrupt.
Value	Description									
0	No interrupt.									
1	Digital Comparator 7 has generated an interrupt.									
6	DCINT6	RW1C	0	<p>Digital Comparator 6 Interrupt Status and Clear</p> <table> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>No interrupt.</td> </tr> <tr> <td>1</td> <td>Digital Comparator 6 has generated an interrupt.</td> </tr> </table> <p>This bit is cleared by writing a 1.</p>	Value	Description	0	No interrupt.	1	Digital Comparator 6 has generated an interrupt.
Value	Description									
0	No interrupt.									
1	Digital Comparator 6 has generated an interrupt.									
5	DCINT5	RW1C	0	<p>Digital Comparator 5 Interrupt Status and Clear</p> <table> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>No interrupt.</td> </tr> <tr> <td>1</td> <td>Digital Comparator 5 has generated an interrupt.</td> </tr> </table> <p>This bit is cleared by writing a 1.</p>	Value	Description	0	No interrupt.	1	Digital Comparator 5 has generated an interrupt.
Value	Description									
0	No interrupt.									
1	Digital Comparator 5 has generated an interrupt.									

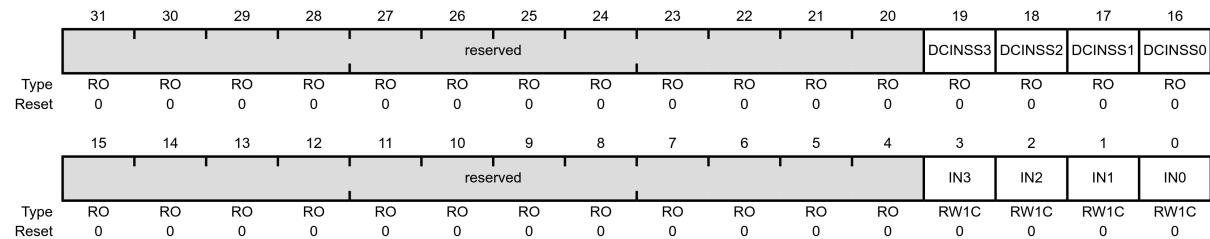
Bit/Field	Name	Type	Reset	Description						
4	DCINT4	RW1C	0	<p>Digital Comparator 4 Interrupt Status and Clear</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No interrupt.</td> </tr> <tr> <td>1</td> <td>Digital Comparator 4 has generated an interrupt.</td> </tr> </tbody> </table> <p>This bit is cleared by writing a 1.</p>	Value	Description	0	No interrupt.	1	Digital Comparator 4 has generated an interrupt.
Value	Description									
0	No interrupt.									
1	Digital Comparator 4 has generated an interrupt.									
3	DCINT3	RW1C	0	<p>Digital Comparator 3 Interrupt Status and Clear</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No interrupt.</td> </tr> <tr> <td>1</td> <td>Digital Comparator 3 has generated an interrupt.</td> </tr> </tbody> </table> <p>This bit is cleared by writing a 1.</p>	Value	Description	0	No interrupt.	1	Digital Comparator 3 has generated an interrupt.
Value	Description									
0	No interrupt.									
1	Digital Comparator 3 has generated an interrupt.									
2	DCINT2	RW1C	0	<p>Digital Comparator 2 Interrupt Status and Clear</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No interrupt.</td> </tr> <tr> <td>1</td> <td>Digital Comparator 2 has generated an interrupt.</td> </tr> </tbody> </table> <p>This bit is cleared by writing a 1.</p>	Value	Description	0	No interrupt.	1	Digital Comparator 2 has generated an interrupt.
Value	Description									
0	No interrupt.									
1	Digital Comparator 2 has generated an interrupt.									
1	DCINT1	RW1C	0	<p>Digital Comparator 1 Interrupt Status and Clear</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No interrupt.</td> </tr> <tr> <td>1</td> <td>Digital Comparator 1 has generated an interrupt.</td> </tr> </tbody> </table> <p>This bit is cleared by writing a 1.</p>	Value	Description	0	No interrupt.	1	Digital Comparator 1 has generated an interrupt.
Value	Description									
0	No interrupt.									
1	Digital Comparator 1 has generated an interrupt.									
0	DCINT0	RW1C	0	<p>Digital Comparator 0 Interrupt Status and Clear</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No interrupt.</td> </tr> <tr> <td>1</td> <td>Digital Comparator 0 has generated an interrupt.</td> </tr> </tbody> </table> <p>This bit is cleared by writing a 1.</p>	Value	Description	0	No interrupt.	1	Digital Comparator 0 has generated an interrupt.
Value	Description									
0	No interrupt.									
1	Digital Comparator 0 has generated an interrupt.									

20.9.27 ADC Interrupt Status and Clear (ADCISC)

This register provides the mechanism for clearing sample sequencer interrupt conditions and shows the status of interrupts generated by the sample sequencers and the digital comparators which have been sent to the interrupt controller.

Documentation:

- ADC0 base: 0x4003.8000
- ADC1 base: 0x4003.9000
- Offset 0x00C
- Type RW1C, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description						
31:20	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
19	DCINSS3	RO	0	<p>Digital Comparator Interrupt Status on SS3</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No interrupt has occurred or the interrupt is masked.</td> </tr> <tr> <td>1</td> <td>Both the INRDC bit in the ADCRIS (page 222) register and the DCONSS3 bit in the ADCIM (page 217) register are set, providing a level-based interrupt to the interrupt controller.</td> </tr> </tbody> </table> <p>This bit is cleared by writing a 1 to it. Clearing this bit also clears the INRDC bit in the ADCRIS (page 222) register.</p>	Value	Description	0	No interrupt has occurred or the interrupt is masked.	1	Both the INRDC bit in the ADCRIS (page 222) register and the DCONSS3 bit in the ADCIM (page 217) register are set, providing a level-based interrupt to the interrupt controller.
Value	Description									
0	No interrupt has occurred or the interrupt is masked.									
1	Both the INRDC bit in the ADCRIS (page 222) register and the DCONSS3 bit in the ADCIM (page 217) register are set, providing a level-based interrupt to the interrupt controller.									
18	DCINSS2	RO	0	<p>Digital Comparator Interrupt Status on SS2</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No interrupt has occurred or the interrupt is masked.</td> </tr> <tr> <td>1</td> <td>Both the INRDC bit in the ADCRIS (page 222) register and the DCONSS2 bit in the ADCIM (page 217) register are set, providing a level-based interrupt to the interrupt controller.</td> </tr> </tbody> </table> <p>This bit is cleared by writing a 1 to it. Clearing this bit also clears the INRDC bit in the ADCRIS (page 222) register.</p>	Value	Description	0	No interrupt has occurred or the interrupt is masked.	1	Both the INRDC bit in the ADCRIS (page 222) register and the DCONSS2 bit in the ADCIM (page 217) register are set, providing a level-based interrupt to the interrupt controller.
Value	Description									
0	No interrupt has occurred or the interrupt is masked.									
1	Both the INRDC bit in the ADCRIS (page 222) register and the DCONSS2 bit in the ADCIM (page 217) register are set, providing a level-based interrupt to the interrupt controller.									

Bit/Field	Name	Type	Reset	Description						
17	DCINSS1	RO	0	<p>Digital Comparator Interrupt Status on SS1</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>No interrupt has occurred or the interrupt is masked.</td></tr> <tr> <td>1</td><td>Both the INRDC bit in the ADCRIS (page 222) register and the DCONSS1 bit in the ADCIM (page 217) register are set, providing a level-based interrupt to the interrupt controller.</td></tr> </tbody> </table> <p>This bit is cleared by writing a 1 to it. Clearing this bit also clears the INRDC bit in the ADCRIS (page 222) register.</p>	Value	Description	0	No interrupt has occurred or the interrupt is masked.	1	Both the INRDC bit in the ADCRIS (page 222) register and the DCONSS1 bit in the ADCIM (page 217) register are set, providing a level-based interrupt to the interrupt controller.
Value	Description									
0	No interrupt has occurred or the interrupt is masked.									
1	Both the INRDC bit in the ADCRIS (page 222) register and the DCONSS1 bit in the ADCIM (page 217) register are set, providing a level-based interrupt to the interrupt controller.									
16	DCINSS0	RO	0	<p>Digital Comparator Interrupt Status on SS0</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>No interrupt has occurred or the interrupt is masked.</td></tr> <tr> <td>1</td><td>Both the INRDC bit in the ADCRIS (page 222) register and the DCONSS0 bit in the ADCIM (page 217) register are set, providing a level-based interrupt to the interrupt controller.</td></tr> </tbody> </table> <p>This bit is cleared by writing a 1 to it. Clearing this bit also clears the INRDC bit in the ADCRIS (page 222) register.</p>	Value	Description	0	No interrupt has occurred or the interrupt is masked.	1	Both the INRDC bit in the ADCRIS (page 222) register and the DCONSS0 bit in the ADCIM (page 217) register are set, providing a level-based interrupt to the interrupt controller.
Value	Description									
0	No interrupt has occurred or the interrupt is masked.									
1	Both the INRDC bit in the ADCRIS (page 222) register and the DCONSS0 bit in the ADCIM (page 217) register are set, providing a level-based interrupt to the interrupt controller.									
15:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
3	IN3	RW1C	0	<p>SS3 Interrupt Status and Clear</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>No interrupt has occurred or the interrupt is masked.</td></tr> <tr> <td>1</td><td>Both the INR3 bit in the ADCRIS (page 222) register and the MASK3 bit in the ADCIM (page 217) register are set, providing a level-based interrupt to the interrupt controller.</td></tr> </tbody> </table> <p>This bit is cleared by writing a 1. Clearing this bit also clears the INR3 bit in the ADCRIS (page 222) register.</p>	Value	Description	0	No interrupt has occurred or the interrupt is masked.	1	Both the INR3 bit in the ADCRIS (page 222) register and the MASK3 bit in the ADCIM (page 217) register are set, providing a level-based interrupt to the interrupt controller.
Value	Description									
0	No interrupt has occurred or the interrupt is masked.									
1	Both the INR3 bit in the ADCRIS (page 222) register and the MASK3 bit in the ADCIM (page 217) register are set, providing a level-based interrupt to the interrupt controller.									

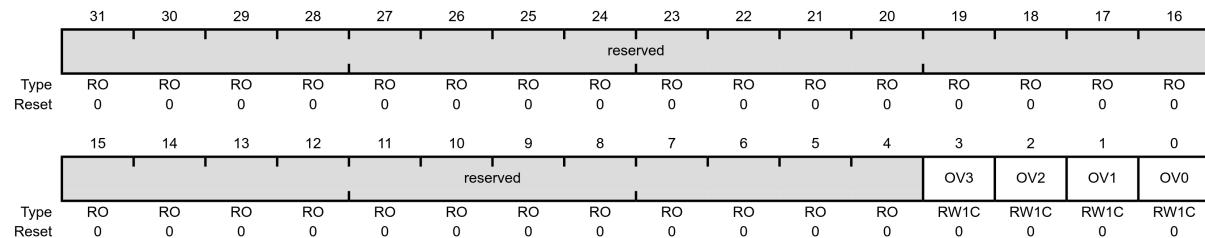
Bit/Field	Name	Type	Reset	Description						
2	IN2	RW1C	0	<p>SS2 Interrupt Status and Clear</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No interrupt has occurred or the interrupt is masked.</td> </tr> <tr> <td>1</td> <td>Both the INR2 bit in the ADCRIS (page 222) register and the MASK2 bit in the ADCIM (page 217) register are set, providing a level-based interrupt to the interrupt controller.</td> </tr> </tbody> </table> <p>This bit is cleared by writing a 1. Clearing this bit also clears the INR2 bit in the ADCRIS (page 222) register.</p>	Value	Description	0	No interrupt has occurred or the interrupt is masked.	1	Both the INR2 bit in the ADCRIS (page 222) register and the MASK2 bit in the ADCIM (page 217) register are set, providing a level-based interrupt to the interrupt controller.
Value	Description									
0	No interrupt has occurred or the interrupt is masked.									
1	Both the INR2 bit in the ADCRIS (page 222) register and the MASK2 bit in the ADCIM (page 217) register are set, providing a level-based interrupt to the interrupt controller.									
1	IN1	RW1C	0	<p>SS1 Interrupt Status and Clear</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No interrupt has occurred or the interrupt is masked.</td> </tr> <tr> <td>1</td> <td>Both the INR1 bit in the ADCRIS (page 222) register and the MASK1 bit in the ADCIM (page 217) register are set, providing a level-based interrupt to the interrupt controller.</td> </tr> </tbody> </table> <p>This bit is cleared by writing a 1. Clearing this bit also clears the INR1 bit in the ADCRIS (page 222) register.</p>	Value	Description	0	No interrupt has occurred or the interrupt is masked.	1	Both the INR1 bit in the ADCRIS (page 222) register and the MASK1 bit in the ADCIM (page 217) register are set, providing a level-based interrupt to the interrupt controller.
Value	Description									
0	No interrupt has occurred or the interrupt is masked.									
1	Both the INR1 bit in the ADCRIS (page 222) register and the MASK1 bit in the ADCIM (page 217) register are set, providing a level-based interrupt to the interrupt controller.									
0	IN0	RW1C	0	<p>SS0 Interrupt Status and Clear</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No interrupt has occurred or the interrupt is masked.</td> </tr> <tr> <td>1</td> <td>Both the INR0 bit in the ADCRIS (page 222) register and the MASK0 bit in the ADCIM (page 217) register are set, providing a level-based interrupt to the interrupt controller.</td> </tr> </tbody> </table> <p>This bit is cleared by writing a 1. Clearing this bit also clears the INR0 bit in the ADCRIS (page 222) register.</p>	Value	Description	0	No interrupt has occurred or the interrupt is masked.	1	Both the INR0 bit in the ADCRIS (page 222) register and the MASK0 bit in the ADCIM (page 217) register are set, providing a level-based interrupt to the interrupt controller.
Value	Description									
0	No interrupt has occurred or the interrupt is masked.									
1	Both the INR0 bit in the ADCRIS (page 222) register and the MASK0 bit in the ADCIM (page 217) register are set, providing a level-based interrupt to the interrupt controller.									

20.9.28 ADC Overflow Status (ADCOSTAT)

This register indicates overflow conditions in the sample sequencer FIFOs.

Documentation:

- ADC0 base: 0x4003.8000
- ADC1 base: 0x4003.9000
- Offset 0x010
- Type RW1C, reset 0x0000.0000

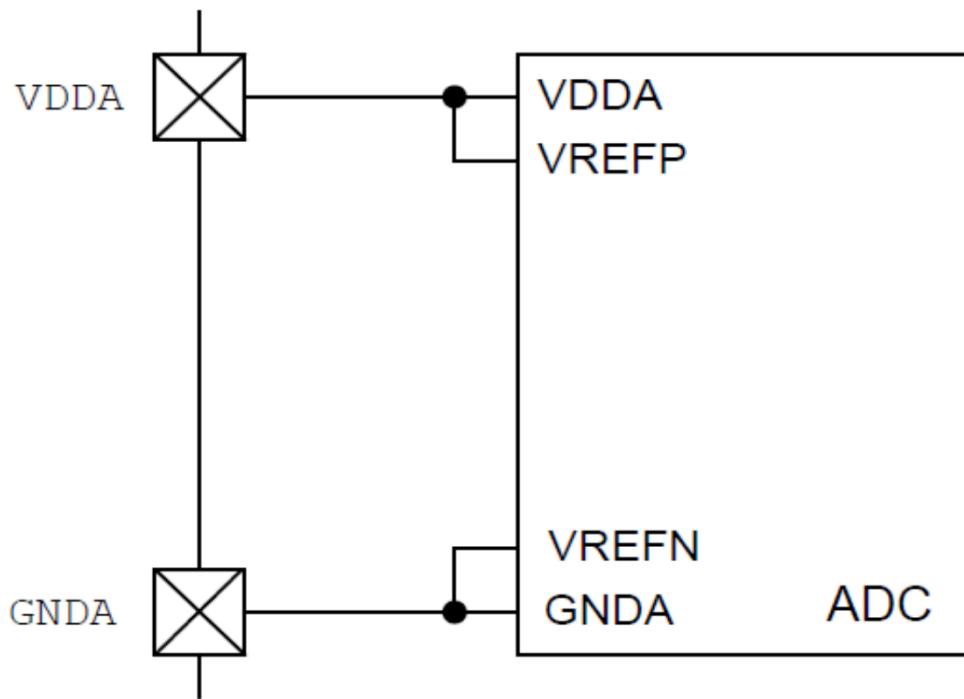


Bit/Field	Name	Type	Reset	Description						
31:4	reserved	RO	0x000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
3	OV3	RW1C	0	<p>SS3 FIFO Overflow</p> <table> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>The FIFO has not overflowed.</td> </tr> <tr> <td>1</td> <td>The FIFO for Sample Sequencer 3 has hit an overflow condition, meaning that the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped.</td> </tr> </table> <p>This bit is cleared by writing a 1.</p>	Value	Description	0	The FIFO has not overflowed.	1	The FIFO for Sample Sequencer 3 has hit an overflow condition, meaning that the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped.
Value	Description									
0	The FIFO has not overflowed.									
1	The FIFO for Sample Sequencer 3 has hit an overflow condition, meaning that the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped.									
2	OV2	RW1C	0	<p>SS2 FIFO Overflow</p> <table> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>The FIFO has not overflowed.</td> </tr> <tr> <td>1</td> <td>The FIFO for Sample Sequencer 2 has hit an overflow condition, meaning that the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped.</td> </tr> </table> <p>This bit is cleared by writing a 1.</p>	Value	Description	0	The FIFO has not overflowed.	1	The FIFO for Sample Sequencer 2 has hit an overflow condition, meaning that the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped.
Value	Description									
0	The FIFO has not overflowed.									
1	The FIFO for Sample Sequencer 2 has hit an overflow condition, meaning that the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped.									

Bit/Field	Name	Type	Reset	Description						
1	OV1	RW1C	0	<p>SS1 FIFO Overflow</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The FIFO has not overflowed.</td> </tr> <tr> <td>1</td> <td>The FIFO for Sample Sequencer 1 has hit an overflow condition, meaning that the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped.</td> </tr> </tbody> </table> <p>This bit is cleared by writing a 1.</p>	Value	Description	0	The FIFO has not overflowed.	1	The FIFO for Sample Sequencer 1 has hit an overflow condition, meaning that the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped.
Value	Description									
0	The FIFO has not overflowed.									
1	The FIFO for Sample Sequencer 1 has hit an overflow condition, meaning that the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped.									
0	OV0	RW1C	0	<p>SS0 FIFO Overflow</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The FIFO has not overflowed.</td> </tr> <tr> <td>1</td> <td>The FIFO for Sample Sequencer 0 has hit an overflow condition, meaning that the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped.</td> </tr> </tbody> </table> <p>This bit is cleared by writing a 1.</p>	Value	Description	0	The FIFO has not overflowed.	1	The FIFO for Sample Sequencer 0 has hit an overflow condition, meaning that the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped.
Value	Description									
0	The FIFO has not overflowed.									
1	The FIFO for Sample Sequencer 0 has hit an overflow condition, meaning that the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped.									

20.10 Reference voltages of Cortex M4 ADC

- 3.3 V
- Internally provided

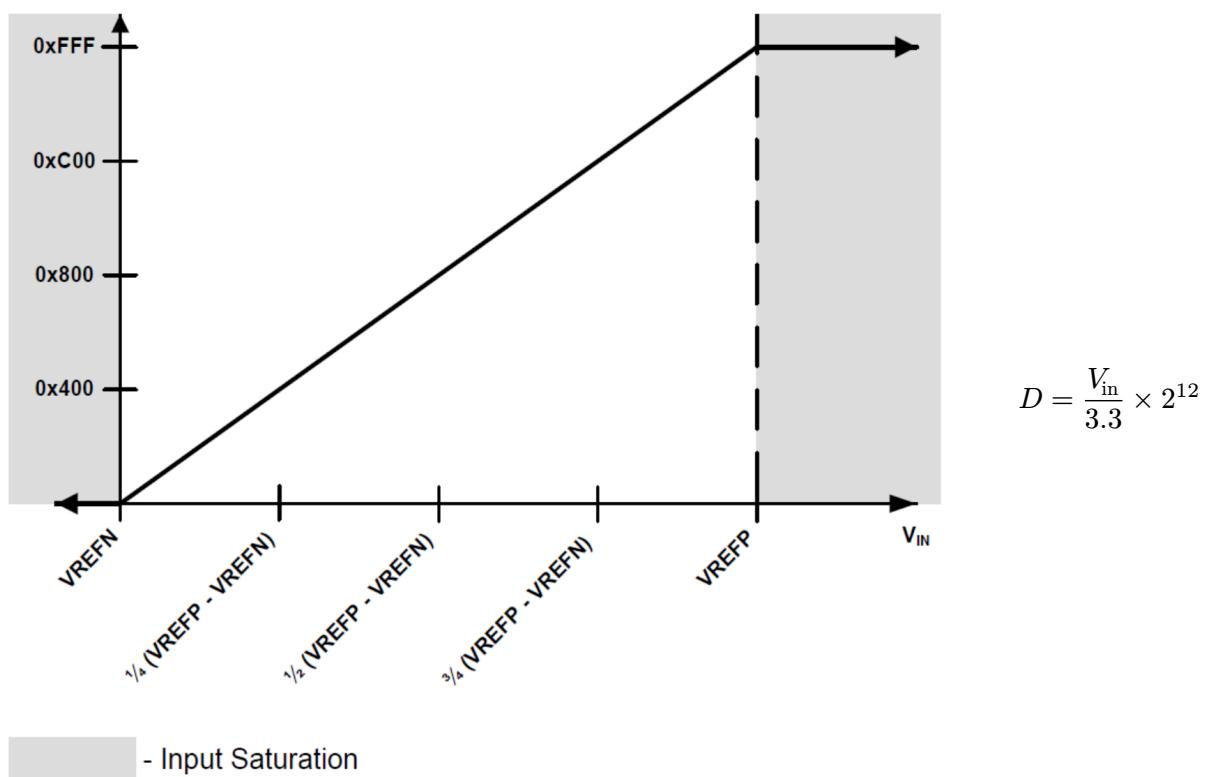


$$V_{\text{refp}} = 3.3 \text{ V}$$

$$V_{\text{refn}} = 0 \text{ V}$$

20.11 Output from Cortex M4 ADC

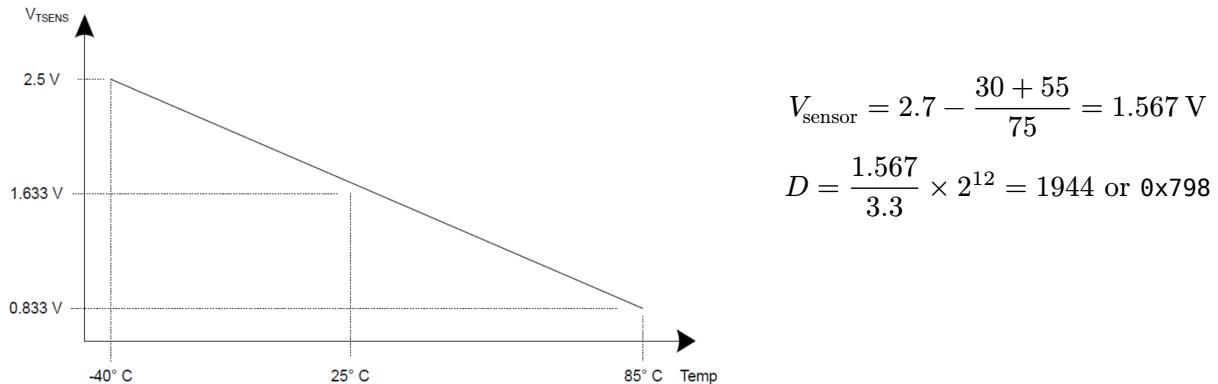
12-bit digital numbers: 0x000 to 0xFFFF



20.11.1 Exercise

The Cortex M4 has an internal temperature sensor. The temperature - voltage plot of the sensor is shown below. If the temperature is 30 °C, what is the digital output from the Cortex M4's ADC?

$$V_{TSENS} = 2.7 \text{ V} - \frac{\text{Temp} + 55}{75}$$



20.12 Initialisation

For the ADC module to be used, the following steps have to be done:

1. Enable the ADC clock using the **RCGCADC** ([page 171](#)) register.
2. Enable the clock to the appropriate GPIO modules via the **RCGCGPIO** ([page 133](#))
3. Set the **GPIOAFSEL** ([page 148](#)) bits for the ADC input pins. To determine which GPIOs to configure, see [page 164](#).
4. Configure the AINx signals to be analog inputs by **clearing** the corresponding DEN bit in the **GPIO Digital Enable** ([page 135](#)) register.
5. Disable the analogue isolation circuit for all ADC input pins that are to be used by writing a 1 to the appropriate bits of the **GPIOAMSEL** ([page 152](#)) register in the associated GPIO block.
6. If required by the application, reconfigure the sample sequencer priorities in the **ADCSSPRI** ([page 172](#)) register. The default configuration has sample sequencer 0 with the highest priority and Sample Sequencer 3 as the lowest priority.

20.12.1 Example

```
GPIO_PORTE_AFSEL_R    EQU 0x40024420
GPIO_PORTE_DEN_R      EQU 0x4002451C
GPIO_PORTE_AMSEL_R    EQU 0x40024528

ADC0_ACTSS_R          EQU 0x40038000
ADC0_PC_R              EQU 0x40038FC4
ADC0_SSPRI_R           EQU 0x40038020
ADC0_EMUX_R            EQU 0x40038014
ADC0_SSMUX3_R          EQU 0x400380A0
ADC0_SSCTL3_R          EQU 0x400380A4
ADC0_IM_R               EQU 0x40038008
ADC0_PSSI_R             EQU 0x40038028
ADC0_RIS_R              EQU 0x40038004
ADC0_SSFIFO3_R          EQU 0x400380A8
ADC0_ISC_R               EQU 0x4003800C

SYSCTL_RCGCGPIO_R     EQU 0x400FE608 ; GPIO run mode clock gating control
SYSCTL_RCGCADC_R       EQU 0x400FE638 ; ADC run mode clock gating control

; Initialize Port E
; Activate clock for Port E
    LDR R1, =SYSCTL_RCGCGPIO_R
    LDR R0, [R1]
    ORR R0, R0, #0x10          ; Turn on GPIO E clock
    STR R0, [R1]
    NOP                      ; Allow time for clock to finish
    NOP
    NOP

; Enable alternate function
; 1 for enable, and 0 for disable
    LDR R1, =GPIO_PORTE_AFSEL_R
    LDR R0, [R1]
    ORR R0, R0, #0x10          ; Enable alternate function on PE4
    STR R0, [R1]

; Disable digital port
; 1 for enable, and 0 for disable
    LDR R1, =GPIO_PORTE_DEN_R
    LDR R0, [R1]
    BIC R0, R0, #0x10          ; Disable digital I/O on PE4
    STR R0, [R1]

; Enable analogue functionality
; 1 for enable, 0 for disable
    LDR R1, =GPIO_PORTE_AMSEL_R
    LDR R0, [R1]
    ORR R0, R0, #0x10          ; Enable PE4 analogue function
    STR R0, [R1]
```

```

; Activate clock for ADC0
LDR R1, =SYSCTL_RCGCADC_R
LDR R0, [R1]
ORR R0, R0, #0x01      ; Activate ADC0
STR R0, [R1]
NOP                  ; Allow time for clock to finish

; Configure the sample rate
LDR R1, =ADC0_PC_R
LDR R0, [R1]
BIC R0, R0, #0x0F      ; Clear max sample rate field
ORR R0, R0, #0x1        ; Configure for 125K samples/sec
STR R0, [R1]

; Configure the sample sequencer priority
LDR R1, =ADC0_SSPRI_R
LDR R0, =0x0123        ; SS3 is highest priority
STR R0, [R1]

; Disable the sample sequencer
LDR R1, =ADC0_ACTSS_R
LDR R0, [R1]
BIC R0, R0, #0x08      ; Disable SS3 before configuration to
STR R0, [R1]            ; prevent erroneous execution

; Set the trigger
LDR R1, =ADC0_EMUX_R
LDR R0, [R1]
BIC R0, R0, #0xF000    ; SS3 is software trigger
STR R0, [R1]

; Set the channel to sample from
LDR R1, =ADC0_SSMUX3_R
LDR R0, [R1]
BIC R0, R0, #0x000F    ; Clear SS3 field
ADD R0, R0, #9          ; Set input pin AIN9
STR R0, [R1]

; Configure the sample
; 0x0006 = 1st sample, no temperature sensor, no differential sampling,
; raw interrupt signal at the end of conversion, first sample is last sample
LDR R1, =ADC0_SSCTL3_R
LDR R0, =0x0006
STR R0, [R1]

; Disable the interrupts on the sample sequencer
LDR R1, =ADC0_IM_R
LDR R0, [R1]
BIC R0, R0, #0x0008    ; Disable SS3 interrupts
STR R0, [R1]

; Enable the sample sequencer
LDR R1, =ADC0_ACTSS_R
LDR R0, [R1]
ORR R0, R0, #0x0008    ; Enable SS3
STR R0, [R1]

```

20.13 Quick reference

	Base address	RCGCADC (page 171) [Address: 0x400F.E638] value
ADC0	0x4003.8000	0x1
ADC0	0x4003.9000	0x2

ADC offsets and values:

- [ADCACTSS](#) ([page 200](#)): 0x00000000
- [ADCPC](#) ([page 170](#)): 0x00000FC4
 - 0x01: 125K samples per second
 - 0x03: 250K samples per second
 - 0x05: 500K samples per second
 - 0x07: 1M samples per second
- [ADCSSPRI](#) ([page 172](#)): 0x00000020
- [ADCEMUX](#) ([page 178](#)): 0x00000014
- [ADCIM](#) ([page 217](#)): 0x00000008
- [ADCPSSI](#) ([page 214](#)): 0x00000028
- [ADCRIS](#) ([page 222](#)): 0x0000.0004
- [ADCISC](#) ([page 226](#)): 0x0000.000C

Sample sequencer 0 offsets and values:

- [ADCSSMUX0](#) ([page 175](#)): 0x0000.0040
- [ADCSSCTL0](#) ([page 183](#)): 0x0000.0044
- [ADCSSFIFO0](#) ([page 219](#)): 0x0000.0048
- Value to set for [ADCACTSS](#) ([page 200](#)) and the value to clear for [ADCIM](#) ([page 217](#)): 0x01
- Sample sequencer 0 priority for [ADCSSPRI](#) ([page 172](#)): 0x3210
- Sample sequencer 0 trigger for [ADCEMUX](#) ([page 178](#)): 0x000F

Sample sequencer 1 offsets and values:

- [ADCSSMUX1](#) ([page 177](#)): 0x0000.0060
- [ADCSSCTL1](#) ([page 192](#)): 0x0000.0064
- [ADCSSFIFO1](#) ([page 219](#)): 0x0000.0068
- Value to set for [ADCACTSS](#) ([page 200](#)) and the value to clear for [ADCIM](#) ([page 217](#)): 0x02
- Sample sequencer 1 priority for [ADCSSPRI](#) ([page 172](#)): 0x2103
- Sample sequencer 1 trigger for [ADCEMUX](#) ([page 178](#)): 0x00F0

Sample sequencer 2 offsets and values:

- [ADCSSMUX2](#) ([page 177](#)): 0x0000.0080
- [ADCSSCTL2](#) ([page 192](#)): 0x0000.0084
- [ADCSSFIFO2](#) ([page 219](#)): 0x0000.0088
- Value to set for [ADCACTSS](#) ([page 200](#)) and the value to clear for [ADCIM](#) ([page 217](#)): 0x04
- Sample sequencer 2 priority for [ADCSSPRI](#) ([page 172](#)): 0x1023
- Sample sequencer 2 trigger for [ADCEMUX](#) ([page 178](#)): 0x0F00

Sample sequencer 3 offsets and values:

- [ADCSSMUX3](#) ([page 177](#)): 0x0000.00A0
- [ADCSSCTL3](#) ([page 198](#)): 0x0000.00A4
- [ADCSSFIFO3](#) ([page 219](#)): 0x0000.00A8
- Value to set for [ADCACTSS](#) ([page 200](#)) and the value to clear for [ADCIM](#) ([page 217](#)): 0x08
- Sample sequencer 3 priority for [ADCSSPRI](#) ([page 172](#)): 0x0123
- Sample sequencer 3 trigger for [ADCEMUX](#) ([page 178](#)): 0xF000

20.14 Differential sampling

In addition to traditional single-ended sampling, the ADC module supports differential sampling of two analogue input channels. To enable differential sampling, software must set the Dn bit in the **ADCSSCTLn** (page 183) register in a step's configuration nibble.

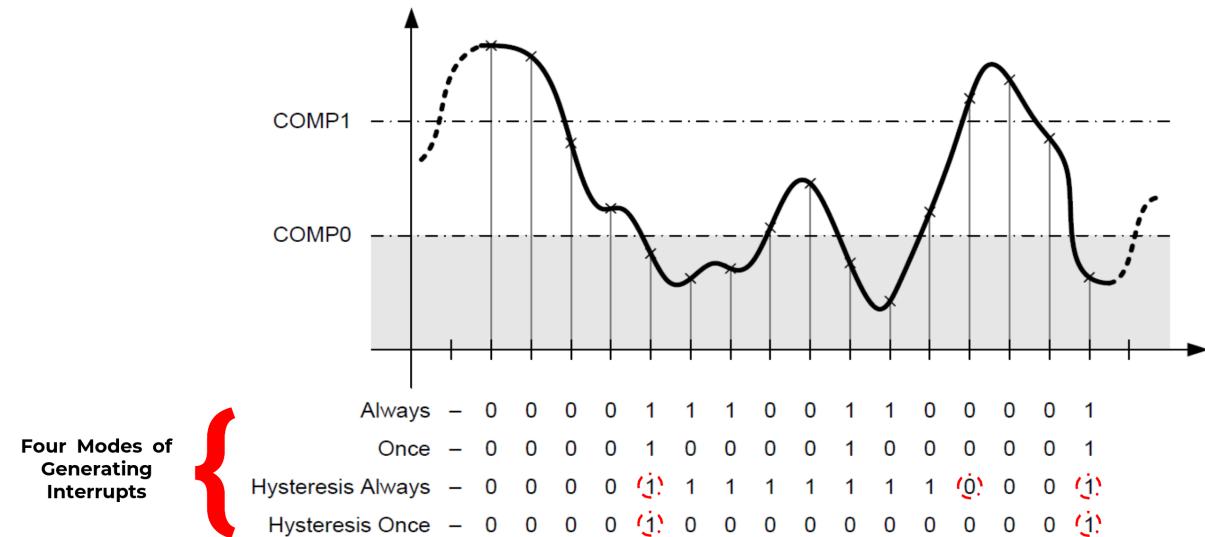
When a sequence step is configured for differential sampling, the input pair to sample must be configured in the **ADCSSMUXn** (page 175) register. Differential pair 0 samples analogue inputs 0 and 1; differential pair 1 samples analogue inputs 2 and 3; and so on. The ADC does not support other differential pairings such as analogue input 0 with analogue input 3.

20.14.1 Differential sampling pairs

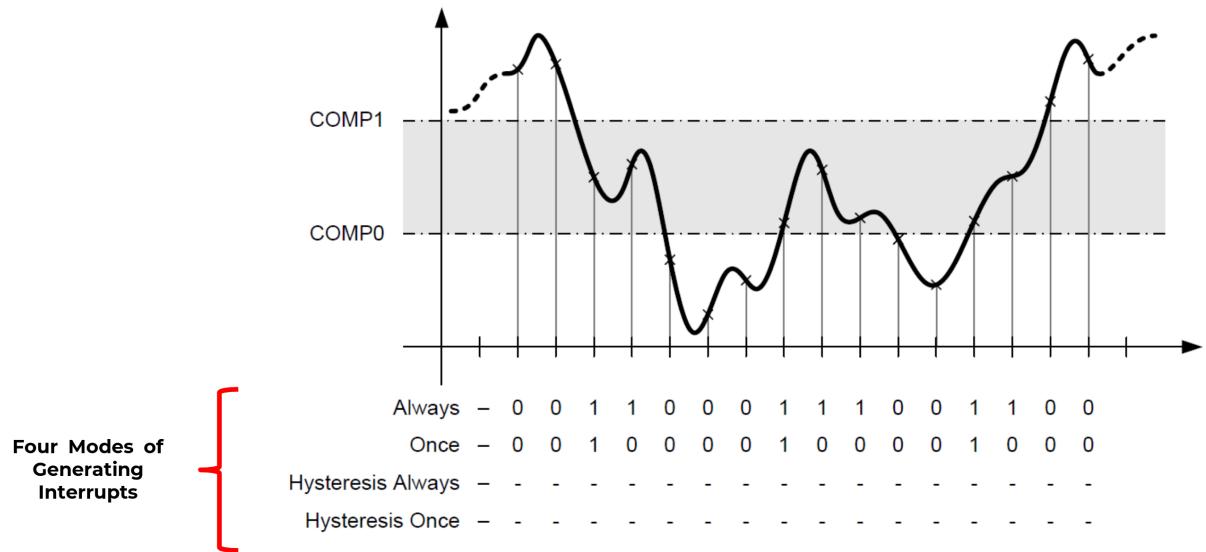
Differential Pair	Analogue Inputs
0	0 and 1
1	2 and 3
2	4 and 5
3	6 and 7
4	8 and 9
5	10 and 11

20.15 Output of interrupts

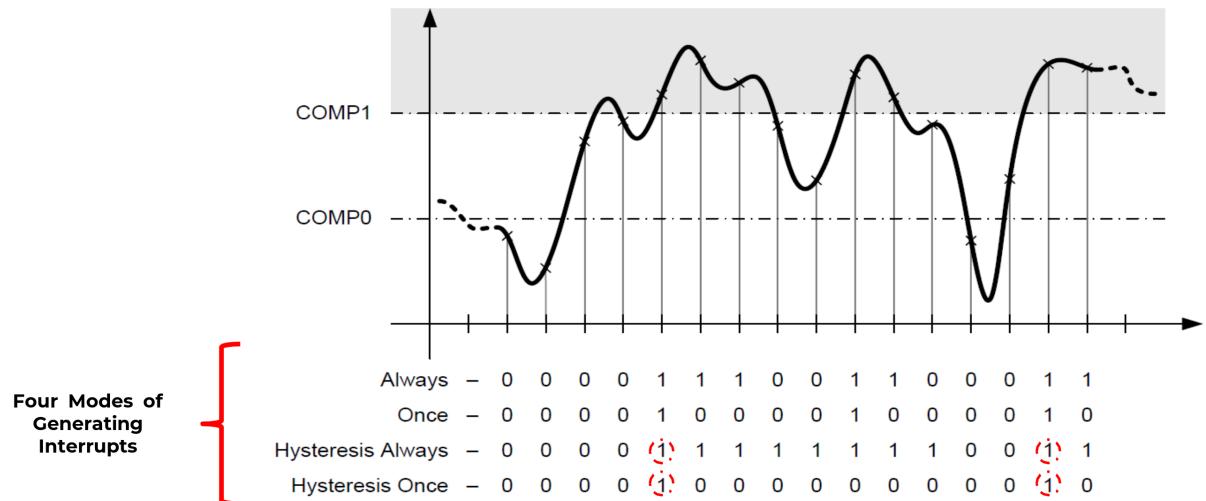
20.15.1 Low band operation



20.15.2 Mid band operation

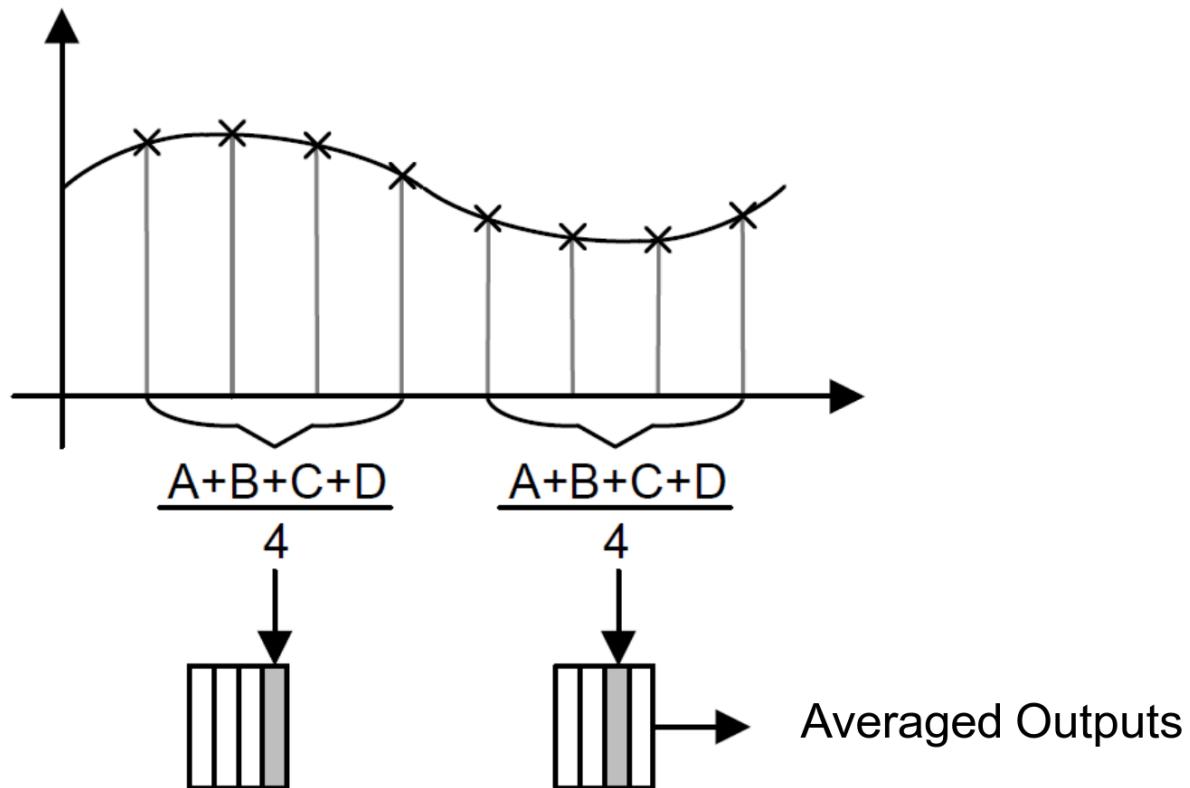


20.15.3 High band operation



20.15.4 Hardware output averaging

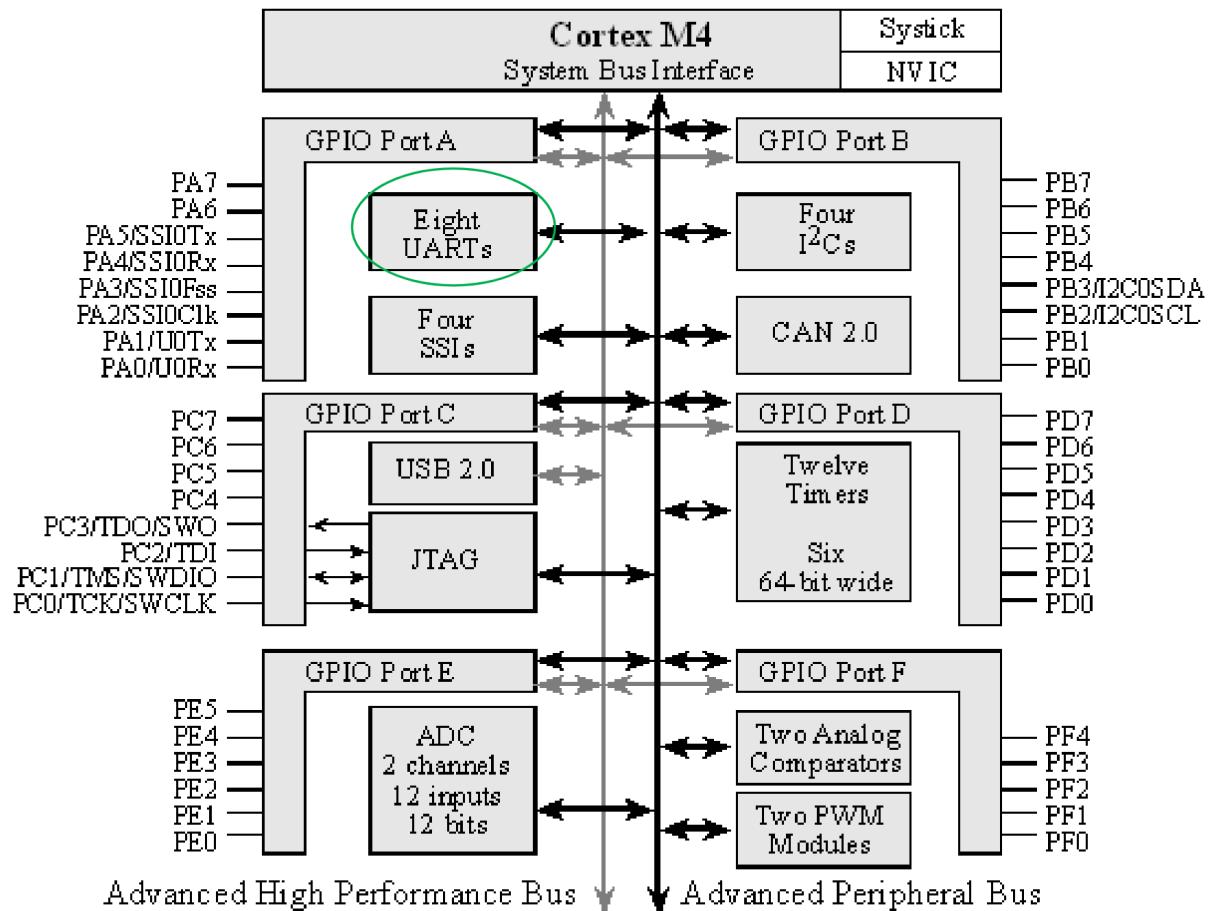
The Cortex M4 has the built-in circuit to support averaging of up to 64 digital outputs.



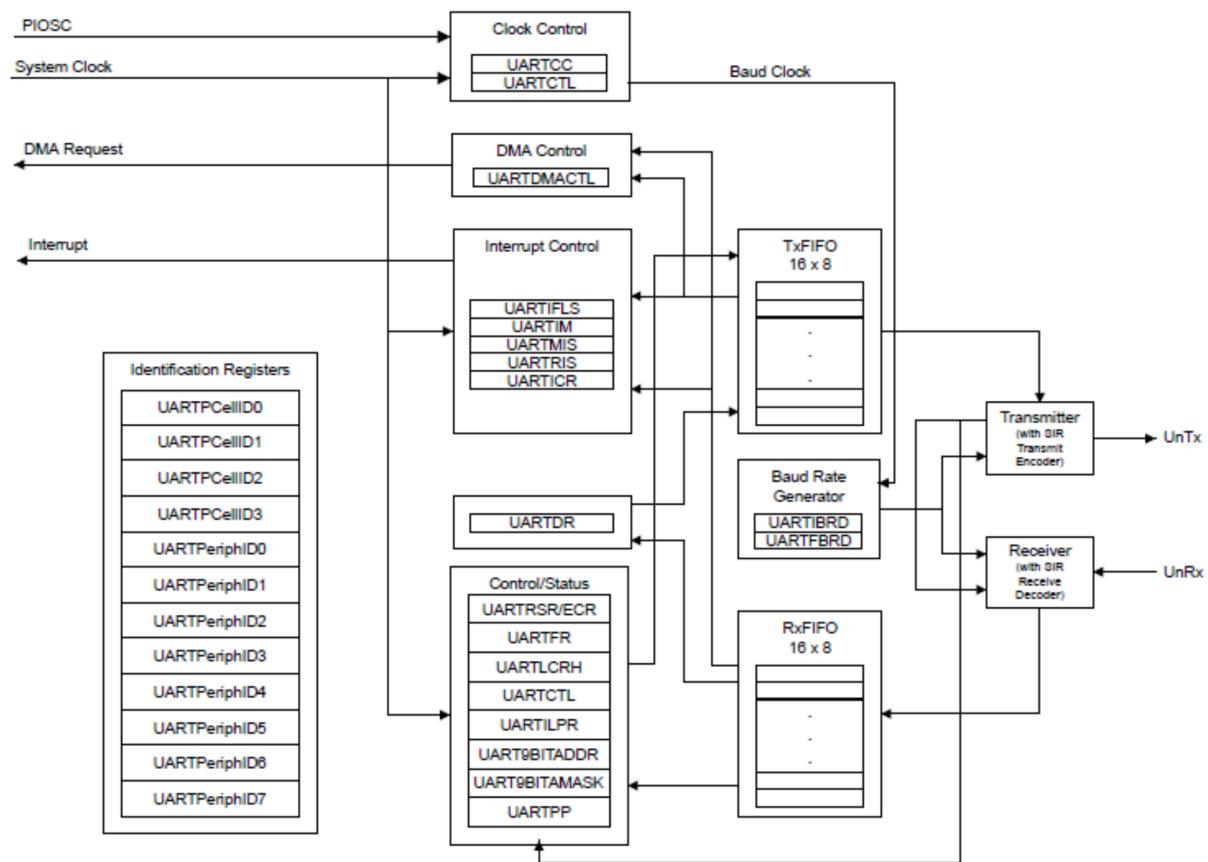
21 TM4C123GH6PM UART

- 8 Universal Asynchronous Receiver/Transmitter (UART) lines.
- Programmable baud-rate generator allowing speeds up to 5 Mbps for regular speed (divide by 16) and 10 Mbps for high speed (divide by 8).
- Separate 16x8 transmitting (TX) and receiving (RX) FIFOs to reduce the load of CPU interrupts.
- Programmable FIFO length, including 1-byte deep operation providing conventional double-buffered interface.
- FIFO trigger levels of $\frac{1}{8}$, $\frac{1}{4}$, $\frac{1}{2}$, $\frac{3}{4}$, and $\frac{7}{8}$.
- Standard asynchronous communication bits for start, stop, and parity line-break generation and detection.
- Line-break generation and detection.
- Fully programmable serial interface characteristics with 5, 6, 7 or 8 data bits.
- Even, odd, stick, or no-parity bit generation and detection.
- 1 or 2 stop bit generation.
- IrDA serial-IR (SIR) encoder/decoder providing programmable use of IrDA Serial Infrared (SIR) or UART input/output.
- Support of IrDA SIR encoder decoder functions for data rates up to 115.2 Kbps half-duplex.
- Support of normal $\frac{3}{16}$ and low-power (1.41 - 2.23 μ s) bit durations.
- Programmable internal clock generator enabling division of reference clock by 1 to 256 for low-power mode bit duration Support for communication with ISO 7816 smart cards.
- Modem flow control (on UART1).
- EIA-485 9-bit support.
- Standard FIFO-level and End-of-Transmission interrupts
- Efficient transfers using Micro Direct Memory Access Controller (μ DMA).
 - Separate channels for transmitting and receiving.
 - Receive “single request asserted” when data is in the FIFO, and “burst request asserted” at programmed FIFO level.
- Transmit “single request asserted” when there is space in the FIFO, and “burst request asserted” at programmed FIFO level.

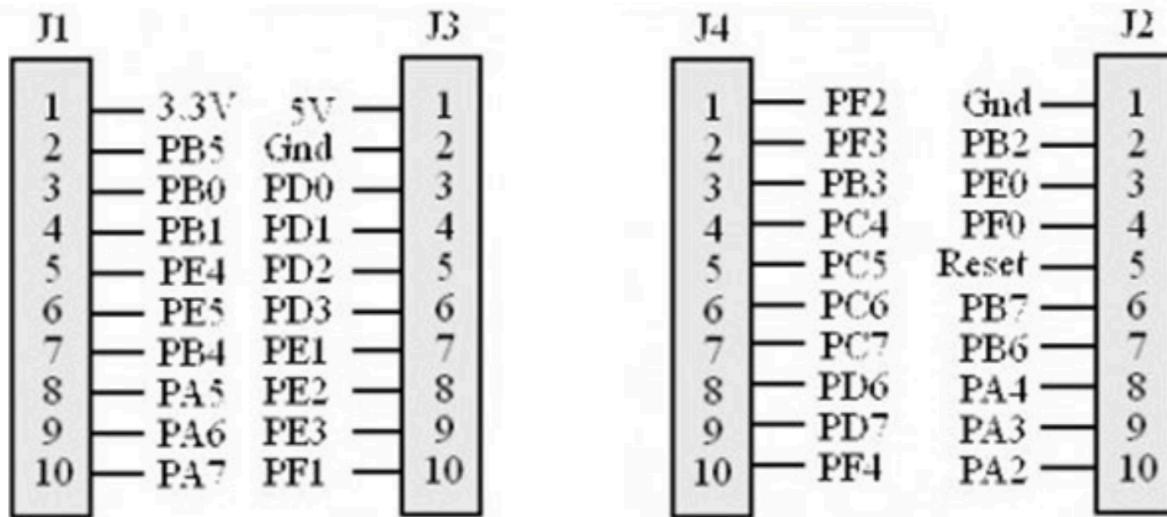
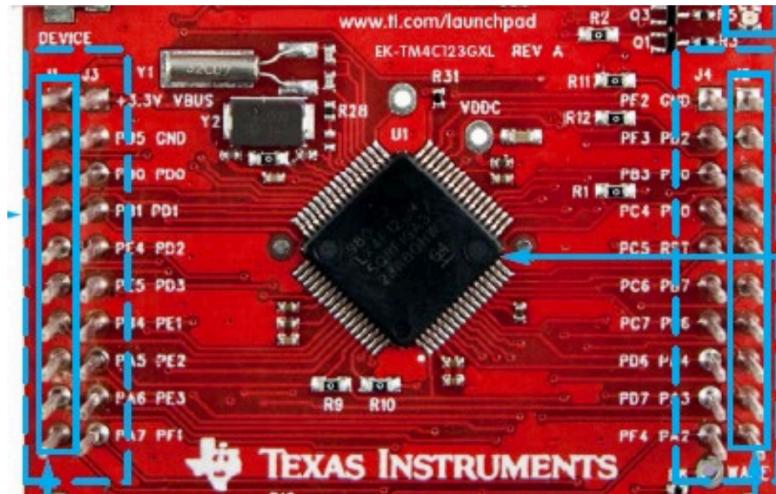
21.1 UARTs in Cortex M4



21.2 UART module block diagram



21.3 J connectors



21.3.1 J1 connector

J1 Pin	GPIO	Analog Function	On-board Function	Tiva C Series MCU Pin	GPIOPCTL Register Setting														
		GPIO AMSEL			1	2	3	4	5	6	7	8	9	14	15				
1.01					3.3 V														
1.02	PB5	AIN11	-	57	-	SSI2Fss	-	M0PWM3	-	-	T1CCP1	CAN0Tx	-	-	-				
1.03	PB0	USB0ID	-	45	U1Rx	-	-	-	-	-	T2CCP0	-	-	-	-				
1.04	PB1	USB0VBUS	-	46	U1Tx	-	-	-	-	-	T2CCP1	-	-	-	-				
1.05	PE4	AIN9	-	59	U5Rx	-	I2C2SCL	M0PWM4	M1PWM2	-	-	CAN0Rx	-	-	-				
1.06	PE5	AIN8	-	60	U5Tx	-	I2C2SDA	M0PWM5	M1PWM3	-	-	CAN0Tx	-	-	-				
1.07	PB4	AIN10	-	58	-	SSI2Clk	-	M0PWM2	-	-	T1CCP0	CAN0Rx	-	-	-				
1.08	PA5	-	-	22	-	SSI0Tx	-	-	-	-	-	-	-	-	-				
1.09	PA6	-	-	23	-	-	I2C1SCL	-	M1PWM2	-	-	-	-	-	-				
1.10	PA7	-	-	24	-	-	I2C1SDA	-	M1PWM3	-	-	-	-	-	-				

21.3.2 J2 connector

J2 Pin	GPIO	Analog Function	On-board Function	Tiva C Series MCU Pin	GPIOCTL Register Setting														
		GPIO AMSEL			1	2	3	4	5	6	7	8	9	14	15				
2.01					GND														
2.02	PB2	-	-	47	-	-	I2C0SCL	-	-	-	T3CCP0	-	-	-	-				
2.03	PE0	AIN3	-	9	U7Rx	-	-	-	-	-	-	-	-	-	-				
2.04	PF0	-	USR_SW2/ WAKE (R1)	28	U1RTS	SSI1Rx	CAN0Rx	-	M1PWM4	PhA0	T0CCP0	NMI	C0o	-	-				
2.05					RESET														
2.06	PB7	-	-	4	-	SSI2Tx	-	M0PWM1	-	-	T0CCP1	-	-	-	-				
	PD1	AIN6	Connected for MSP430 Compatibility (R10)	62	SSI3Fss	SSI1Fss	I2C3SDA	M0PWM7	M1PWM1	-	WT2CCP1	-	-	-	-				
2.07	PB6	-	-	1	-	SSI2Rx	-	M0PWM0	-	-	T0CCP0	-	-	-	-				
	PD0	AIN7	Connected for MSP430 Compatibility (R9)	61	SSI3Clk	SSI1Clk	I2C3SCL	M0PWM6	M1PWM0	-	WT2CCP2	-	-	-	-				
2.08	PA4	-	-	21	-	SSI0Rx	-	-	-	-	-	-	-	-	-	-	-	-	-
2.09	PA3	-	-	20	-	SSI0Fss	-	-	-	-	-	-	-	-	-	-	-	-	-
2.10	PA2	-	-	19	-	SSI0Clk	-	-	-	-	-	-	-	-	-	-	-	-	-

21.3.3 J3 connector

J3 Pin	GPIO	Analog Function	On-board Function	Tiva C Series MCU Pin	GPIOCTL Register Setting														
		GPIO AMSEL			1	2	3	4	5	6	7	8	9	14	15				
3.01					5.0 V														
3.02					GND														
3.03	PD0	AIN7	-	61	SSI3Clk	SSI1Clk	I2C3SCL	M0PWM6	M1PWM0	-	WT2CCP2	-	-	-	-				
	PB6	-	Connected for MSP430 Compatibility (R9)	1	-	SSI2Rx	-	M0PWM0	-	-	T0CCP0	-	-	-	-				
3.04	PD1	AIN6	-	92	SSI3Fss	SSI1Fss	I2C3SDA	M0PWM7	M1PWM1	-	WT2CCP1	-	-	-	-				
	PB7	-	Connected for MSP430 Compatibility (R10)	4	-	SSI2Tx	-	M0PWM1	-	-	T0CCP1	-	-	-	-				
3.05	PD2	AIN5	-	63	SSI3Rx	SSI1Rx	-	M0FAULT0	-	-	WT3CCP0	USB0EPEN	-	-	-				
3.06	PD3	AIN4	-	64	SSI3Tx	SSI1Tx	-	-	-	-	WT3CCP1	USB0PFLT	-	-	-				
3.07	PE1	AIN2	-	8	U7Tx	-	-	-	-	-	-	-	-	-	-				
3.08	PE2	AIN1	-	7	-	-	-	-	-	-	-	-	-	-	-				
3.09	PE3	AIN0	-	6	-	-	-	-	-	-	-	-	-	-	-				
3.10	PF1	-	-	29	U1CTS	SSI1Tx	-	-	M1PWM5	-	T0CCP1	-	C1o	TRD1	-				

21.3.4 J4 connector

J4 Pin	GPIO	Analog Function	On-board Function	Tiva C Series MCU Pin	GPIOCTL Register Setting										
					1	2	3	4	5	6	7	8	9	14	15
4.01	PF2	-	Blue LED (R11)	30	-	SSI1Clk	-	M0FAULT0	M1PWM6	-	T1CCP0	-	-	-	TRD0
4.02	PF3	-	Green LED (R12)	31	-	SSI1Fss	CAN0Tx	-	M1PWM7	-	T1CCP1	-	-	-	TRCLK
4.03	PB3	-	-	48	-	-	I2C0SDA	-	-	-	T3CCP1	-	-	-	-
4.04	PC4	C1-	-	16	U4Rx	U1Rx	-	M0PWM6	-	IDX1	WT0CCP0	U1RTS	-	-	-
4.05	PC5	C1+	-	15	U4Tx	U1Tx	-	M0PWM7	-	PhA1	WT0CCP1	U1CTS	-	-	-
4.06	PC6	C0+	-	14	U3Rx	-	-	-	-	PhB1	WT1CCP0	USB0EPEN	-	-	-
4.07	PC7	C0-	-	13	U3Tx	-	-	-	-	-	WT1CCP1	USB0PFLT	-	-	-
4.08	PD6	-	-	53	U2Rx	-	-	-	-	PhA0	WT5CCP0	-	-	-	-
4.09	PD7	-	-	10	U2Tx	-	-	-	-	PhB0	WT5CCP1	NMI	-	-	-
4.10	PF4	-	USR_SW1 (R13)	5	-	-	-	-	M1FAULT0	IDX0	T2CCP0	USB0EPEN	-	-	-

21.4 UART description and configuration

U0	U0TX → U0RX ←
U1	U1TX → U1RX ←
U2	U2TX → U2RX ←
U3	U3TX → U3RX ←
U4	U4TX → U4RX ←
U5	U5TX → U5RX ←
U6	U6TX → U6RX ←
U7	U7TX → U7RX ←

Register Map:

- UART0: 0x4000.C000
- UART1: 0x4000.D000
- UART2: 0x4000.E000
- UART3: 0x4000.F000
- UART4: 0x4001.0000
- UART5: 0x4001.1000
- UART6: 0x4001.2000
- UART7: 0x4001.3000

Above are the 8 UART lines of the TM4C123GH6PM controller.

21.5 UART register map

Offset	Name	Type	Reset	Description
0x000	UARTDR	RW	0x0000.0000	UART Data
0x004	UARTRSR/UARTECR	RW	0x0000.0000	UART Receive Status/Error Clear
0x018	UARTFR	RO	0x0000.0090	UART Flag
0x020	UARTILPR	RW	0x0000.0000	UART IrDA Low-Power Register
0x024	UARTIBRD	RW	0x0000.0000	UART Integer Baud-Rate Divisor
0x028	UARTFBRD	RW	0x0000.0000	UART Fractional Baud-Rate Divisor
0x02C	UARTLCRH	RW	0x0000.0000	UART Line Control
0x030	UARTCTL	RW	0x0000.0300	UART Control
0x034	UARTIFLS	RW	0x0000.0012	UART Interrupt FIFO Level Select
0x038	UARTIM	RW	0x0000.0000	UART Interrupt Mask
0x03C	UARTRIS	RO	0x0000.0000	UART Raw Interrupt Status
0x040	UARTMIS	RO	0x0000.0000	UART Masked Interrupt Status
0x044	UARTICR	W1C	0x0000.0000	UART Interrupt Clear
0x048	UARTDMACTL	RW	0x0000.0000	UART DMA Control
0x0A4	UART9BITADDR	RW	0x0000.0000	UART 9-Bit Self Address
0x0A8	UART9BITAMASK	RW	0x0000.00FF	UART 9-Bit Self Address Mask
0xFC0	UARTPP	RO	0x0000.0003	UART Peripheral Properties
0xFC8	UARTCC	RW	0x0000.0000	UART Clock Configuration

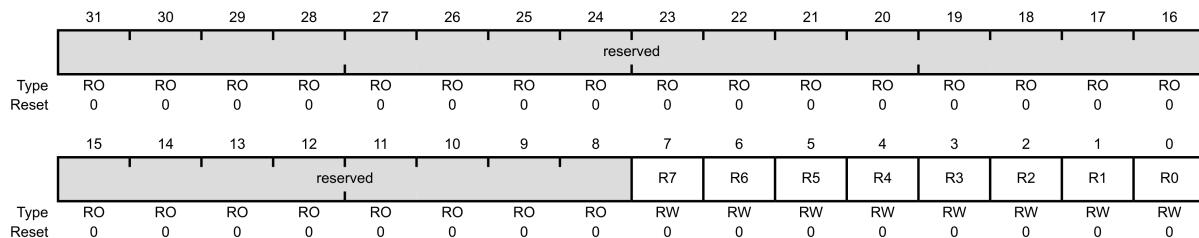
21.6 Essential registers

\$4000.C000	31-12	11	10	9	8	7-0	Name
	[OE]	[BE]	[PE]	[FE]	[DATA]		UART0_DR_R
\$4000.C004	31-3			3	2	1	0
	[OE]	[BE]	[PE]	[FE]			UART0_RSR_R
\$4000.C018	31-8	7	6	5	4	3	2-0
	[TXFE]	[RXFF]	[TXFF]	[RXFE]	[BUSY]		UART0_FR_R
\$4000.C024	31-16			15-0	DIVINT		UART0_IBRD_R
\$4000.C028	31-6			5-0	DIVFRAC		UART0_FBRD_R
\$4000.C02C	31-8	7	6-5	4	3	2	1
	[SPS]	[WPEN]	[FEN]	[STP2]	[EPS]	[PEN]	[BRK]
\$4000.C030	31-10	9	8	7	6-3	2	1
	[RXE]	[TXE]	[LBE]	[SIRLP]	[SIREN]	[UARTEN]	UART0_CTL_R

21.7 Registers

21.7.1 Universal Asynchronous Receiver/Transmitter Run Mode Clock Gating Control (RCCUART)

- Base `0x400F.E000`
- Offset `0x618`
- Type RW, reset `0x0000.0000`

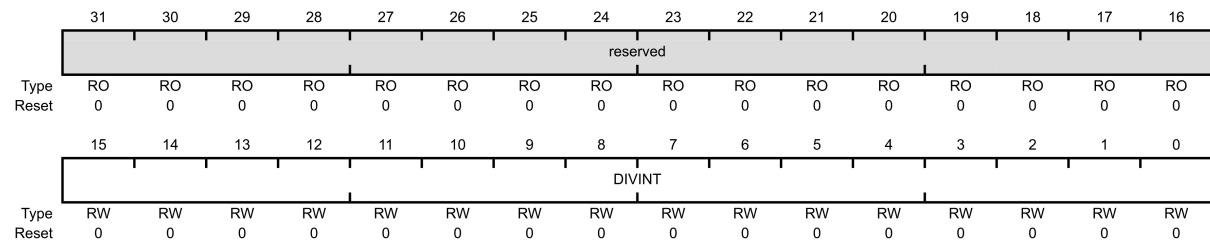


Bit/Field	Name	Type	Reset	Description											
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.											
7	R7	RW	0	UART Module 7 Run Mode Clock Gating Control Value Description 0 UART module 7 is disabled. 1 Enable and provide a clock to UART module 7 in Run mode.											
6	R6	RW	0	UART Module 6 Run Mode Clock Gating Control Value Description 0 UART module 6 is disabled. 1 Enable and provide a clock to UART module 6 in Run mode.											
5	R5	RW	0	UART Module 5 Run Mode Clock Gating Control Value Description 0 UART module 5 is disabled. 1 Enable and provide a clock to UART module 5 in Run mode.											
4	R4	RW	0	UART Module 4 Run Mode Clock Gating Control Value Description 0 UART module 4 is disabled. 1 Enable and provide a clock to UART module 4 in Run mode.											

Bit/Field	Name	Type	Reset	Description
3	R3	RW	0	UART Module 3 Run Mode Clock Gating Control Value Description 0 UART module 3 is disabled. 1 Enable and provide a clock to UART module 3 in Run mode.
2	R2	RW	0	UART Module 2 Run Mode Clock Gating Control Value Description 0 UART module 2 is disabled. 1 Enable and provide a clock to UART module 2 in Run mode.
1	R1	RW	0	UART Module 1 Run Mode Clock Gating Control Value Description 0 UART module 1 is disabled. 1 Enable and provide a clock to UART module 1 in Run mode.
0	R0	RW	0	UART Module 0 Run Mode Clock Gating Control Value Description 0 UART module 0 is disabled. 1 Enable and provide a clock to UART module 0 in Run mode.

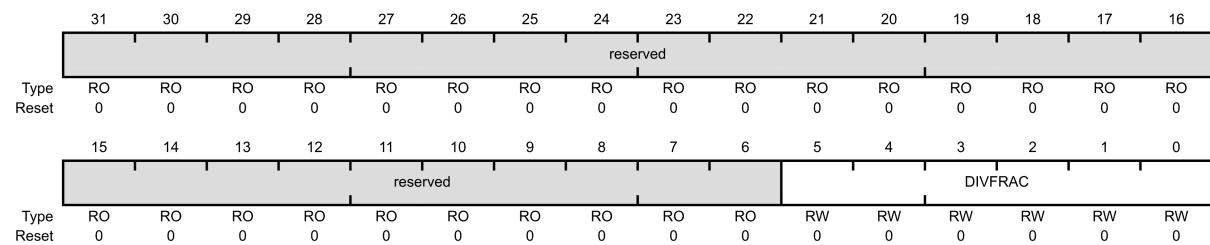
21.7.2 UART Integer Baud-Rate Divisor (UARTIBRD)

- UART0 base: 0x4000.C000
- UART1 base: 0x4000.D000
- UART2 base: 0x4000.E000
- UART3 base: 0x4000.F000
- UART4 base: 0x4001.0000
- UART5 base: 0x4001.1000
- UART6 base: 0x4001.2000
- UART7 base: 0x4001.3000
- Offset 0x024
- Type RW, reset 0x0000.0000



21.7.3 UART Fractional Baud-Rate Divisor (UARTFBRD)

- UART0 base: 0x4000.C000
- UART1 base: 0x4000.D000
- UART2 base: 0x4000.E000
- UART3 base: 0x4000.F000
- UART4 base: 0x4001.0000
- UART5 base: 0x4001.1000
- UART6 base: 0x4001.2000
- UART7 base: 0x4001.3000
- Offset 0x028
- Type RW, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:0	DIVFRAC	RW	0x0	Fractional Baud-Rate Divisor

21.7.4 UART Line Control (UARTLCRH)

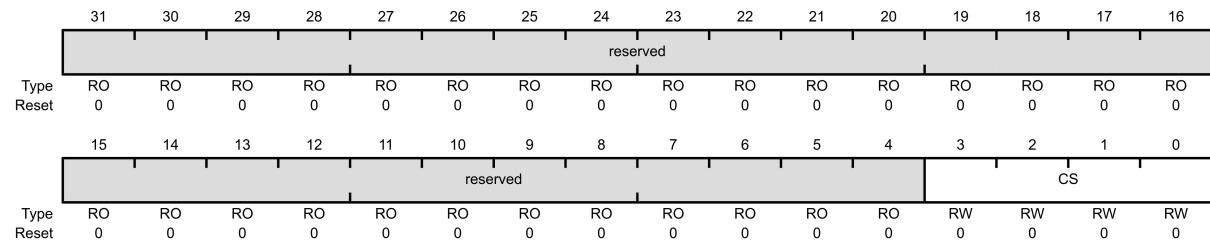
- UART0 base: 0x4000.C000
- UART1 base: 0x4000.D000
- UART2 base: 0x4000.E000
- UART3 base: 0x4000.F000
- UART4 base: 0x4001.0000
- UART5 base: 0x4001.1000
- UART6 base: 0x4001.2000
- UART7 base: 0x4001.3000
- Offset 0x02C
- Type RW, reset 0x0000.0000

Bit/Field	Name	Type	Reset	Description										
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.										
7	SPS	RW	0	UART Stick Parity Select When bits 1, 2, and 7 of UARTLCRH (page 251) are set, the parity bit is transmitted and checked as a 0. When bits 1 and 7 are set and 2 is cleared, the parity bit is transmitted and checked as a 1. When this bit is cleared, stick parity is disabled.										
6:5	WLEN	RW	0x0	UART Word Length The bits indicate the number of data bits transmitted or received in a frame as follows: <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>5 bits (default)</td> </tr> <tr> <td>0x1</td> <td>6 bits</td> </tr> <tr> <td>0x2</td> <td>7 bits</td> </tr> <tr> <td>0x3</td> <td>8 bits</td> </tr> </tbody> </table>	Value	Description	0x0	5 bits (default)	0x1	6 bits	0x2	7 bits	0x3	8 bits
Value	Description													
0x0	5 bits (default)													
0x1	6 bits													
0x2	7 bits													
0x3	8 bits													
4	FEN	RW	0	UART Enable FIFOs <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The FIFOs are disabled (Character mode). The FIFOs become 1-byte-deep holding registers.</td> </tr> <tr> <td>1</td> <td>The transmit and receive FIFO buffers are enabled (FIFO mode).</td> </tr> </tbody> </table>	Value	Description	0	The FIFOs are disabled (Character mode). The FIFOs become 1-byte-deep holding registers.	1	The transmit and receive FIFO buffers are enabled (FIFO mode).				
Value	Description													
0	The FIFOs are disabled (Character mode). The FIFOs become 1-byte-deep holding registers.													
1	The transmit and receive FIFO buffers are enabled (FIFO mode).													

Bit/Field	Name	Type	Reset	Description						
3	STP2	RW	0	<p>UART Two Stop Bits Select</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>One stop bit is transmitted at the end of a frame.</td></tr> <tr> <td>1</td><td>Two stop bits are transmitted at the end of a frame. The receive logic does not check for two stop bits being received.</td></tr> </tbody> </table> <p>When in 7816 smartcard mode (the SMART bit is set in the UARTCTL (page 255) register), the number of stop bits is forced to 2.</p>	Value	Description	0	One stop bit is transmitted at the end of a frame.	1	Two stop bits are transmitted at the end of a frame. The receive logic does not check for two stop bits being received.
Value	Description									
0	One stop bit is transmitted at the end of a frame.									
1	Two stop bits are transmitted at the end of a frame. The receive logic does not check for two stop bits being received.									
2	EPS	RW	0	<p>UART Even Parity Select</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Odd parity is performed, which checks for an odd number of 1s.</td></tr> <tr> <td>1</td><td>Even parity generation and checking is performed during transmission and reception, which checks for an even number of 1s in data and parity bits.</td></tr> </tbody> </table> <p>This bit has no effect when parity is disabled by the PEN bit.</p>	Value	Description	0	Odd parity is performed, which checks for an odd number of 1s.	1	Even parity generation and checking is performed during transmission and reception, which checks for an even number of 1s in data and parity bits.
Value	Description									
0	Odd parity is performed, which checks for an odd number of 1s.									
1	Even parity generation and checking is performed during transmission and reception, which checks for an even number of 1s in data and parity bits.									
1	PEN	RW	0	<p>UART Parity Enable</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Parity is disabled and no parity bit is added to the data frame.</td></tr> <tr> <td>1</td><td>Parity checking and generation is enabled.</td></tr> </tbody> </table>	Value	Description	0	Parity is disabled and no parity bit is added to the data frame.	1	Parity checking and generation is enabled.
Value	Description									
0	Parity is disabled and no parity bit is added to the data frame.									
1	Parity checking and generation is enabled.									
0	BRK	RW	0	<p>UART Send Break</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Normal use.</td></tr> <tr> <td>1</td><td>A Low level is continually output on the UnTx signal, after completing transmission of the current character. For the proper execution of the break command, software must set this bit for at least two frames (character periods).</td></tr> </tbody> </table>	Value	Description	0	Normal use.	1	A Low level is continually output on the UnTx signal, after completing transmission of the current character. For the proper execution of the break command, software must set this bit for at least two frames (character periods).
Value	Description									
0	Normal use.									
1	A Low level is continually output on the UnTx signal, after completing transmission of the current character. For the proper execution of the break command, software must set this bit for at least two frames (character periods).									

21.7.5 UART Clock Configuration (UARTCC)

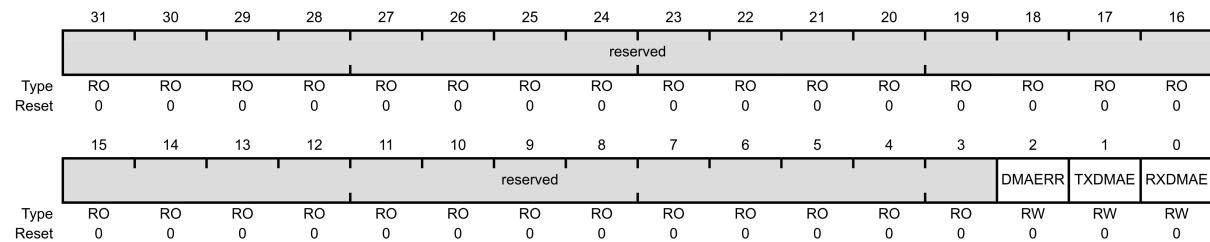
- UART0 base: 0x4000.C000
- UART1 base: 0x4000.D000
- UART2 base: 0x4000.E000
- UART3 base: 0x4000.F000
- UART4 base: 0x4001.0000
- UART5 base: 0x4001.1000
- UART6 base: 0x4001.2000
- UART7 base: 0x4001.3000
- Offset 0xFC8
- Type RW, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description										
31:4	reserved	RO	0x0000.000	<p>Software should not rely on the value of a reserved bit.</p> <p>To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p>										
3:0	CS	RW	0	<p>UART Baud Clock Source</p> <p>The following table specifies the source that generates for the UART baud clock:</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>System clock (based on clock source and divisor factor)</td> </tr> <tr> <td>0x1 - 0x4</td> <td>Reserved</td> </tr> <tr> <td>0x5</td> <td>PIOSC</td> </tr> <tr> <td>0x5 - 0xF</td> <td>Reserved</td> </tr> </tbody> </table>	Value	Description	0x0	System clock (based on clock source and divisor factor)	0x1 - 0x4	Reserved	0x5	PIOSC	0x5 - 0xF	Reserved
Value	Description													
0x0	System clock (based on clock source and divisor factor)													
0x1 - 0x4	Reserved													
0x5	PIOSC													
0x5 - 0xF	Reserved													

21.7.6 UART DMA Control (UARTDMACTL)

- UART0 base: 0x4000.C000
- UART1 base: 0x4000.D000
- UART2 base: 0x4000.E000
- UART3 base: 0x4000.F000
- UART4 base: 0x4001.0000
- UART5 base: 0x4001.1000
- UART6 base: 0x4001.2000
- UART7 base: 0x4001.3000
- Offset 0x048
- Type RW, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description						
31:3	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
2	DMAERR	RW	0	<p>DMA on Error</p> <table> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>μDMA receive requests are unaffected when a receive error occurs.</td> </tr> <tr> <td>1</td> <td>μDMA receive requests are automatically disabled when a receive error occurs.</td> </tr> </table>	Value	Description	0	μDMA receive requests are unaffected when a receive error occurs.	1	μDMA receive requests are automatically disabled when a receive error occurs.
Value	Description									
0	μDMA receive requests are unaffected when a receive error occurs.									
1	μDMA receive requests are automatically disabled when a receive error occurs.									
1	TXDMAE	RW	0	<p>Transmit DMA Enable</p> <table> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>μDMA for the transmit FIFO is disabled.</td> </tr> <tr> <td>1</td> <td>μDMA for the transmit FIFO is enabled.</td> </tr> </table>	Value	Description	0	μDMA for the transmit FIFO is disabled.	1	μDMA for the transmit FIFO is enabled.
Value	Description									
0	μDMA for the transmit FIFO is disabled.									
1	μDMA for the transmit FIFO is enabled.									
0	RXDMAE	RW	0	<p>Receive DMA Enable</p> <table> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>μDMA for the receive FIFO is disabled.</td> </tr> <tr> <td>1</td> <td>μDMA for the receive FIFO is enabled.</td> </tr> </table>	Value	Description	0	μDMA for the receive FIFO is disabled.	1	μDMA for the receive FIFO is enabled.
Value	Description									
0	μDMA for the receive FIFO is disabled.									
1	μDMA for the receive FIFO is enabled.									

21.7.7 UART Control (UARTCTL)

- UART0 base: 0x4000.C000
- UART1 base: 0x4000.D000
- UART2 base: 0x4000.E000
- UART3 base: 0x4000.F000
- UART4 base: 0x4001.0000
- UART5 base: 0x4001.1000
- UART6 base: 0x4001.2000
- UART7 base: 0x4001.3000
- Offset 0x030
- Type RW, reset 0x0000.0300

The UARTCTL register is the control register. All the bits are cleared on reset except for the Transmit Enable (TXE) and Receive Enable (RXE) bits, which are set.

- To enable the UART module, the **UARTEN** bit must be set.
- If software requires a configuration change in the module, the **UARTEN** bit must be cleared before the configuration changes are written.
- If the UART is disabled during a transmitting or receiving operation, the current transaction is completed prior to the UART stopping.
- **Note:** The **UARTCTL** ([page 255](#)) register should not be changed while the UART is enabled, otherwise the results may be unpredictable.

Making changes to the **UARTCTL** ([page 255](#)) register:

1. Disable the UART, by clearing the **UARTEN** bit.
2. Wait for the end of transmission or reception of the current character, by writing a few lines of NOP.
3. Flush the transmission FIFO by clearing bit 4 (FEN) FIFO enable in the line control register (**UARTLCRH** ([page 251](#))).
4. Reprogram the control register.
5. Enable the UART by setting the **UARTEN** bit.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CTSEN	RTSEN	reserved		RTS	reserved	RXE	TXE	LBE	reserved	HSE	EOT	SMART	SIRLP	SIREN	UARTEN
Type	RW	RW	RO	RO	RW	RO	RW	RW	RW	RO	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description						
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
15	CTSEN	RW	0	<p>Enable Clear To Send</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>CTS hardware flow control is disabled.</td> </tr> <tr> <td>1</td> <td>CTS hardware flow control is enabled. Data is only transmitted when the U1CTS signal is asserted.</td> </tr> </tbody> </table>	Value	Description	0	CTS hardware flow control is disabled.	1	CTS hardware flow control is enabled. Data is only transmitted when the U1CTS signal is asserted.
Value	Description									
0	CTS hardware flow control is disabled.									
1	CTS hardware flow control is enabled. Data is only transmitted when the U1CTS signal is asserted.									
14	RTSEN	RW	0	<p>Enable Request to Send</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>RTS hardware flow control is disabled.</td> </tr> <tr> <td>1</td> <td>RTS hardware flow control is enabled. Data is only requested (by asserting U1RTS) when the receive FIFO has available entries.</td> </tr> </tbody> </table>	Value	Description	0	RTS hardware flow control is disabled.	1	RTS hardware flow control is enabled. Data is only requested (by asserting U1RTS) when the receive FIFO has available entries.
Value	Description									
0	RTS hardware flow control is disabled.									
1	RTS hardware flow control is enabled. Data is only requested (by asserting U1RTS) when the receive FIFO has available entries.									
13:12	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
11	RTS	RW	0	Request to Send When RTSEN is clear, the status of this bit is reflected on the U1RTS signal. If RTSEN is set, this bit is ignored on a write and should be ignored on read.						
10	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
9	RXE	RW	1	<p>UART Receive Enable</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The receive section of the UART is disabled.</td> </tr> <tr> <td>1</td> <td>The receive section of the UART is enabled.</td> </tr> </tbody> </table> <p>If the UART is disabled in the middle of a receive, it completes the current character before stopping.</p> <p>Note: To enable reception, the UARTEN bit must also be set.</p>	Value	Description	0	The receive section of the UART is disabled.	1	The receive section of the UART is enabled.
Value	Description									
0	The receive section of the UART is disabled.									
1	The receive section of the UART is enabled.									

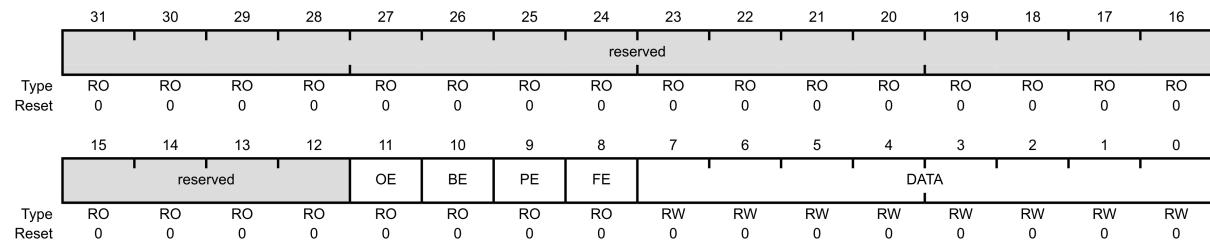
Bit/Field	Name	Type	Reset	Description						
8	TXE	RW	1	<p>UART Transmit Enable</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The transmit section of the UART is disabled.</td></tr> <tr> <td>1</td><td>The transmit section of the UART is enabled.</td></tr> </tbody> </table> <p>If the UART is disabled in the middle of a transmission, it completes the current character before stopping.</p> <p>Note: To enable transmission, the UARTEN bit must also be set.</p>	Value	Description	0	The transmit section of the UART is disabled.	1	The transmit section of the UART is enabled.
Value	Description									
0	The transmit section of the UART is disabled.									
1	The transmit section of the UART is enabled.									
7	LBE	RW	0	<p>UART Loop Back Enable</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Normal operation.</td></tr> <tr> <td>1</td><td>The UnTx path is fed through the UnRx path.</td></tr> </tbody> </table>	Value	Description	0	Normal operation.	1	The UnTx path is fed through the UnRx path.
Value	Description									
0	Normal operation.									
1	The UnTx path is fed through the UnRx path.									
6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
5	HSE	RW	0	<p>High-Speed Enable</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The UART is clocked using the system clock divided by 16.</td></tr> <tr> <td>1</td><td>The UART is clocked using the system clock divided by 8.</td></tr> </tbody> </table> <p>Note: System clock used is also dependent on the baud-rate divisor configuration (see page 249 and page 250).</p> <p>The state of this bit has no effect on clock generation in ISO 7816 smart card mode (the SMART bit is set).</p>	Value	Description	0	The UART is clocked using the system clock divided by 16.	1	The UART is clocked using the system clock divided by 8.
Value	Description									
0	The UART is clocked using the system clock divided by 16.									
1	The UART is clocked using the system clock divided by 8.									
4	EOT	RW	0	<p>End of Transmission</p> <p>This bit determines the behavior of the TXRIS bit in the UARTRIS (page 264) register.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The TXRIS bit is set when the transmit FIFO condition specified in UARTIFLS (page 276) is met.</td></tr> <tr> <td>1</td><td>The TXRIS bit is set only after all transmitted data, including stop bits, have cleared the serializer.</td></tr> </tbody> </table>	Value	Description	0	The TXRIS bit is set when the transmit FIFO condition specified in UARTIFLS (page 276) is met.	1	The TXRIS bit is set only after all transmitted data, including stop bits, have cleared the serializer.
Value	Description									
0	The TXRIS bit is set when the transmit FIFO condition specified in UARTIFLS (page 276) is met.									
1	The TXRIS bit is set only after all transmitted data, including stop bits, have cleared the serializer.									

Bit/Field	Name	Type	Reset	Description						
3	SMART	RW	0	<p>ISO 7816 Smart Card Support</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Normal operation.</td></tr> <tr> <td>1</td><td>The UART operates in Smart Card mode.</td></tr> </tbody> </table> <p>The application must ensure that it sets 8-bit word length (WLEN set to 0x3) and even parity (PEN set to 1, EPS set to 1, SPS set to 0) in UARTLCRH (page 251) when using ISO 7816 mode.</p> <p>In this mode, the value of the STP2 bit in UARTLCRH (page 251) is ignored and the number of stop bits is forced to 2. Note that the UART does not support automatic retransmission on parity errors. If a parity error is detected on transmission, all further transmit operations are aborted and software must handle retransmission of the affected byte or message.</p>	Value	Description	0	Normal operation.	1	The UART operates in Smart Card mode.
Value	Description									
0	Normal operation.									
1	The UART operates in Smart Card mode.									
2	SIRLP	RW	0	<p>UART SIR Low-Power Mode</p> <p>This bit selects the IrDA encoding mode.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Low-level bits are transmitted as an active High pulse with a width of 3/16th of the bit period.</td></tr> <tr> <td>1</td><td>The UART operates in SIR Low-Power mode. Low-level bits are transmitted with a pulse width which is 3 times the period of the IRLPBaud16 input signal, regardless of the selected bit rate.</td></tr> </tbody> </table> <p>Setting this bit uses less power, but might reduce transmission distances.</p>	Value	Description	0	Low-level bits are transmitted as an active High pulse with a width of 3/16th of the bit period.	1	The UART operates in SIR Low-Power mode. Low-level bits are transmitted with a pulse width which is 3 times the period of the IRLPBaud16 input signal, regardless of the selected bit rate.
Value	Description									
0	Low-level bits are transmitted as an active High pulse with a width of 3/16th of the bit period.									
1	The UART operates in SIR Low-Power mode. Low-level bits are transmitted with a pulse width which is 3 times the period of the IRLPBaud16 input signal, regardless of the selected bit rate.									
1	SIREN	RW	0	<p>UART SIR Enable</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Normal operation.</td></tr> <tr> <td>1</td><td>The IrDA SIR block is enabled, and the UART will transmit and receive data using SIR protocol.</td></tr> </tbody> </table>	Value	Description	0	Normal operation.	1	The IrDA SIR block is enabled, and the UART will transmit and receive data using SIR protocol.
Value	Description									
0	Normal operation.									
1	The IrDA SIR block is enabled, and the UART will transmit and receive data using SIR protocol.									

Bit/Field	Name	Type	Reset	Description						
0	UARTEN	RW	0	<p>UART Enable</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The UART is disabled.</td> </tr> <tr> <td>1</td> <td>The UART is enabled.</td> </tr> </tbody> </table> <p>If the UART is disabled in the middle of transmission or reception, it completes the current character before stopping.</p>	Value	Description	0	The UART is disabled.	1	The UART is enabled.
Value	Description									
0	The UART is disabled.									
1	The UART is enabled.									

21.7.8 UART Data (UARTDR)

- UART0 base: 0x4000.C000
- UART1 base: 0x4000.D000
- UART2 base: 0x4000.E000
- UART3 base: 0x4000.F000
- UART4 base: 0x4001.0000
- UART5 base: 0x4001.1000
- UART6 base: 0x4001.2000
- UART7 base: 0x4001.3000
- Offset 0x000
- Type RW, reset 0x0000.0000

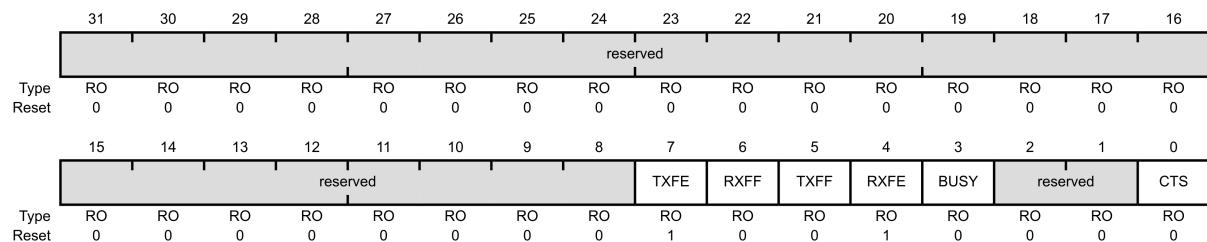


Bit/Field	Name	Type	Reset	Description						
31:12	reserved	RO	0x0000.0	<p>Software should not rely on the value of a reserved bit.</p> <p>To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p>						
11	OE	RO	0	<p>UART Overrun Error</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No data has been lost due to a FIFO overrun.</td> </tr> <tr> <td>1</td> <td>New data was received when the FIFO was full, resulting in data loss.</td> </tr> </tbody> </table>	Value	Description	0	No data has been lost due to a FIFO overrun.	1	New data was received when the FIFO was full, resulting in data loss.
Value	Description									
0	No data has been lost due to a FIFO overrun.									
1	New data was received when the FIFO was full, resulting in data loss.									
10	BE	RO	0	<p>UART Break Error</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No break condition has occurred</td> </tr> <tr> <td>1</td> <td>A break condition has been detected, indicating that the receive data input was held Low for longer than a full-word transmission time (defined as start, data, parity, and stop bits).</td> </tr> </tbody> </table> <p>In FIFO mode, this error is associated with the character at the top of the FIFO. When a break occurs, only one character is loaded into the FIFO. The next character is only enabled after the received data input goes to a 1 (marking state), and the next valid start bit is received.</p>	Value	Description	0	No break condition has occurred	1	A break condition has been detected, indicating that the receive data input was held Low for longer than a full-word transmission time (defined as start, data, parity, and stop bits).
Value	Description									
0	No break condition has occurred									
1	A break condition has been detected, indicating that the receive data input was held Low for longer than a full-word transmission time (defined as start, data, parity, and stop bits).									

Bit/Field	Name	Type	Reset	Description						
9	PE	RO	0	<p>UART Parity Error</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No parity error has occurred</td> </tr> <tr> <td>1</td> <td>The parity of the received data character does not match the parity defined by bits 2 and 7 of the UARTLCRH (page 251) register.</td> </tr> </tbody> </table> <p>In FIFO mode, this error is associated with the character at the top of the FIFO.</p>	Value	Description	0	No parity error has occurred	1	The parity of the received data character does not match the parity defined by bits 2 and 7 of the UARTLCRH (page 251) register.
Value	Description									
0	No parity error has occurred									
1	The parity of the received data character does not match the parity defined by bits 2 and 7 of the UARTLCRH (page 251) register.									
8	FE	RO	0	<p>UART Framing Error</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No framing error has occurred</td> </tr> <tr> <td>1</td> <td>The received character does not have a valid stop bit (a valid stop bit is 1).</td> </tr> </tbody> </table>	Value	Description	0	No framing error has occurred	1	The received character does not have a valid stop bit (a valid stop bit is 1).
Value	Description									
0	No framing error has occurred									
1	The received character does not have a valid stop bit (a valid stop bit is 1).									
7:0	DATA	RW	0x00	<p>Data Transmitted or Received</p> <p>Data that is to be transmitted via the UART is written to this field. When read, this field contains the data that was received by the UART.</p>						

21.7.9 UART Flag (UARTFR)

- UART0 base: 0x4000.C000
- UART1 base: 0x4000.D000
- UART2 base: 0x4000.E000
- UART3 base: 0x4000.F000
- UART4 base: 0x4001.0000
- UART5 base: 0x4001.1000
- UART6 base: 0x4001.2000
- UART7 base: 0x4001.3000
- Offset 0x018
- Type RO, reset 0x0000.0090

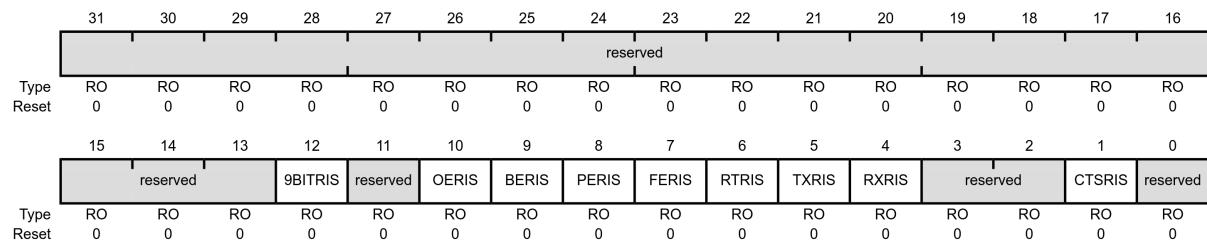


Bit/Field	Name	Type	Reset	Description						
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
7	TXFE	RO	1	<p>UART Transmit FIFO Empty</p> <p>The meaning of this bit depends on the state of the FEN bit in the UARTLCRH (page 251) register.</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The transmitter has data to transmit.</td> </tr> <tr> <td>1</td> <td>If the FIFO is disabled (FEN is 0), the transmit holding register is empty. If the FIFO is enabled (FEN is 1), the transmit FIFO is empty.</td> </tr> </tbody> </table>	Value	Description	0	The transmitter has data to transmit.	1	If the FIFO is disabled (FEN is 0), the transmit holding register is empty. If the FIFO is enabled (FEN is 1), the transmit FIFO is empty.
Value	Description									
0	The transmitter has data to transmit.									
1	If the FIFO is disabled (FEN is 0), the transmit holding register is empty. If the FIFO is enabled (FEN is 1), the transmit FIFO is empty.									
6	RXFF	RO	0	<p>UART Receive FIFO Full</p> <p>The meaning of this bit depends on the state of the FEN bit in the UARTLCRH (page 251) register.</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The receiver can receive data.</td> </tr> <tr> <td>1</td> <td>If the FIFO is disabled (FEN is 0), the receive holding register is full. If the FIFO is enabled (FEN is 1), the receive FIFO is full.</td> </tr> </tbody> </table>	Value	Description	0	The receiver can receive data.	1	If the FIFO is disabled (FEN is 0), the receive holding register is full. If the FIFO is enabled (FEN is 1), the receive FIFO is full.
Value	Description									
0	The receiver can receive data.									
1	If the FIFO is disabled (FEN is 0), the receive holding register is full. If the FIFO is enabled (FEN is 1), the receive FIFO is full.									

Bit/Field	Name	Type	Reset	Description						
5	TXFF	RO	0	<p>UART Transmit FIFO Full</p> <p>The meaning of this bit depends on the state of the FEN bit in the UARTLCRH (page 251) register.</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The transmitter is not full.</td> </tr> <tr> <td>1</td> <td>If the FIFO is disabled (FEN is 0), the transmit holding register is full. If the FIFO is enabled (FEN is 1), the transmit FIFO is full.</td> </tr> </tbody> </table>	Value	Description	0	The transmitter is not full.	1	If the FIFO is disabled (FEN is 0), the transmit holding register is full. If the FIFO is enabled (FEN is 1), the transmit FIFO is full.
Value	Description									
0	The transmitter is not full.									
1	If the FIFO is disabled (FEN is 0), the transmit holding register is full. If the FIFO is enabled (FEN is 1), the transmit FIFO is full.									
4	RXFE	RO	1	<p>UART Receive FIFO Empty</p> <p>The meaning of this bit depends on the state of the FEN bit in the UARTLCRH (page 251) register.</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The receiver is not empty.</td> </tr> <tr> <td>1</td> <td>If the FIFO is disabled (FEN is 0), the receive holding register is empty. If the FIFO is enabled (FEN is 1), the receive FIFO is empty.</td> </tr> </tbody> </table>	Value	Description	0	The receiver is not empty.	1	If the FIFO is disabled (FEN is 0), the receive holding register is empty. If the FIFO is enabled (FEN is 1), the receive FIFO is empty.
Value	Description									
0	The receiver is not empty.									
1	If the FIFO is disabled (FEN is 0), the receive holding register is empty. If the FIFO is enabled (FEN is 1), the receive FIFO is empty.									
3	BUSY	RO	0	<p>UART Busy</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The UART is not busy.</td> </tr> <tr> <td>1</td> <td>The UART is busy transmitting data. This bit remains set until the complete byte, including all stop bits, has been sent from the shift register.</td> </tr> </tbody> </table> <p>This bit is set as soon as the transmit FIFO becomes non-empty (regardless of whether UART is enabled).</p>	Value	Description	0	The UART is not busy.	1	The UART is busy transmitting data. This bit remains set until the complete byte, including all stop bits, has been sent from the shift register.
Value	Description									
0	The UART is not busy.									
1	The UART is busy transmitting data. This bit remains set until the complete byte, including all stop bits, has been sent from the shift register.									
2:1	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
0	CTS	RO	0	<p>Clear To Send</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The U1CTS signal is not asserted.</td> </tr> <tr> <td>1</td> <td>The U1CTS signal is asserted.</td> </tr> </tbody> </table>	Value	Description	0	The U1CTS signal is not asserted.	1	The U1CTS signal is asserted.
Value	Description									
0	The U1CTS signal is not asserted.									
1	The U1CTS signal is asserted.									

21.7.10 UART Raw Interrupt Status (UARTRIS)

- UART0 base: 0x4000.C000
- UART1 base: 0x4000.D000
- UART2 base: 0x4000.E000
- UART3 base: 0x4000.F000
- UART4 base: 0x4001.0000
- UART5 base: 0x4001.1000
- UART6 base: 0x4001.2000
- UART7 base: 0x4001.3000
- Offset 0x03C
- Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description						
31:13	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
12	9BITRIS	RO	0	<p>9-Bit Mode Raw Interrupt Status</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No interrupt</td> </tr> <tr> <td>1</td> <td>A receive address match has occurred.</td> </tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the 9BITIC bit in the UARTICR (page 274) register.</p>	Value	Description	0	No interrupt	1	A receive address match has occurred.
Value	Description									
0	No interrupt									
1	A receive address match has occurred.									
11	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
10	OERIS	RO	0	<p>UART Overrun Error Raw Interrupt Status</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No interrupt</td> </tr> <tr> <td>1</td> <td>An overrun error has occurred.</td> </tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the OEIC bit in the UARTICR (page 274) register.</p>	Value	Description	0	No interrupt	1	An overrun error has occurred.
Value	Description									
0	No interrupt									
1	An overrun error has occurred.									

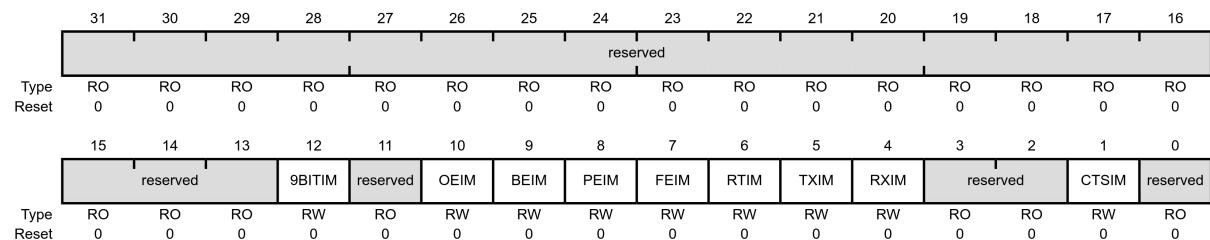
Bit/Field	Name	Type	Reset	Description						
9	BERIS	RO	0	<p>UART Break Error Raw Interrupt Status</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No interrupt</td> </tr> <tr> <td>1</td> <td>A break error has occurred.</td> </tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the BEIC bit in the UARTICR (page 274) register.</p>	Value	Description	0	No interrupt	1	A break error has occurred.
Value	Description									
0	No interrupt									
1	A break error has occurred.									
8	PERIS	RO	0	<p>UART Parity Error Raw Interrupt Status</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No interrupt</td> </tr> <tr> <td>1</td> <td>A parity error has occurred.</td> </tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the PEIC bit in the UARTICR (page 274) register.</p>	Value	Description	0	No interrupt	1	A parity error has occurred.
Value	Description									
0	No interrupt									
1	A parity error has occurred.									
7	FERIS	RO	0	<p>UART Framing Error Raw Interrupt Status</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No interrupt</td> </tr> <tr> <td>1</td> <td>A framing error has occurred.</td> </tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the FEIC bit in the UARTICR (page 274) register.</p>	Value	Description	0	No interrupt	1	A framing error has occurred.
Value	Description									
0	No interrupt									
1	A framing error has occurred.									
6	RTRIS	RO	0	<p>UART Receive Time-Out Raw Interrupt Status</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No interrupt</td> </tr> <tr> <td>1</td> <td>A receive time out has occurred.</td> </tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the RTIC bit in the UARTICR (page 274) register. For receive timeout, the RTIM bit in the UARTIM (page 268) register must be set to see the RTRIS status.</p>	Value	Description	0	No interrupt	1	A receive time out has occurred.
Value	Description									
0	No interrupt									
1	A receive time out has occurred.									

Bit/Field	Name	Type	Reset	Description						
5	TXRIS	RO	0	<p>UART Transmit Raw Interrupt Status</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>No interrupt</td></tr> <tr> <td>1</td><td>If the EOT bit in the UARTCTL (page 255) register is clear, the transmit FIFO level has passed through the condition defined in the UARTIFLS (page 276) register.</td></tr> </tbody> </table> <p>If the EOT bit is set, the last bit of all transmitted data and flags has left the serializer.</p> <p>This bit is cleared by writing a 1 to the TXIC bit in the UARTICR (page 274) register or by writing data to the transmit FIFO until it becomes greater than the trigger level, if the FIFO is enabled, or by reading a single byte if the FIFO is disabled.</p>	Value	Description	0	No interrupt	1	If the EOT bit in the UARTCTL (page 255) register is clear, the transmit FIFO level has passed through the condition defined in the UARTIFLS (page 276) register.
Value	Description									
0	No interrupt									
1	If the EOT bit in the UARTCTL (page 255) register is clear, the transmit FIFO level has passed through the condition defined in the UARTIFLS (page 276) register.									
4	RXRIS	RO	0	<p>UART Receive Raw Interrupt Status</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>No interrupt</td></tr> <tr> <td>1</td><td>The receive FIFO level has passed through the condition defined in the UARTIFLS (page 276) register.</td></tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the RXIC bit in the UARTICR (page 274) register or by reading data from the receive FIFO until it becomes less than the trigger level, if the FIFO is enabled, or by reading a single byte if the FIFO is disabled.</p>	Value	Description	0	No interrupt	1	The receive FIFO level has passed through the condition defined in the UARTIFLS (page 276) register.
Value	Description									
0	No interrupt									
1	The receive FIFO level has passed through the condition defined in the UARTIFLS (page 276) register.									
3:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
1	CTSRIS	RO	0	<p>UART Clear to Send Modem Raw Interrupt Status</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>No interrupt</td></tr> <tr> <td>1</td><td>Clear to Send used for software flow control.</td></tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the CTSIC bit in the UARTICR (page 274) register.</p> <p>This bit is implemented only on UART1 and is reserved for UART0 and UART2.</p>	Value	Description	0	No interrupt	1	Clear to Send used for software flow control.
Value	Description									
0	No interrupt									
1	Clear to Send used for software flow control.									

Bit/Field	Name	Type	Reset	Description
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

21.7.11 UART Interrupt Mask (UARTIM)

- UART0 base: 0x4000.C000
- UART1 base: 0x4000.D000
- UART2 base: 0x4000.E000
- UART3 base: 0x4000.F000
- UART4 base: 0x4001.0000
- UART5 base: 0x4001.1000
- UART6 base: 0x4001.2000
- UART7 base: 0x4001.3000
- Offset 0x038
- Type RW, reset 0x0000.0000



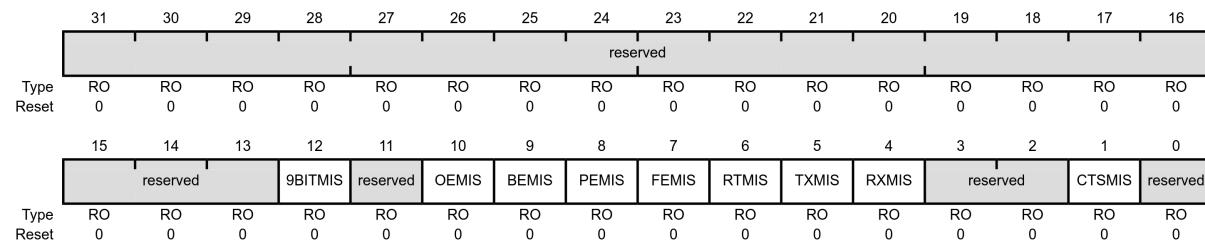
Bit/Field	Name	Type	Reset	Description						
31:13	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
12	9BITIM	RW	0	<p>9-Bit Mode Interrupt Mask</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The 9BITRIS interrupt is suppressed and not sent to the interrupt controller.</td> </tr> <tr> <td>1</td> <td>An interrupt is sent to the interrupt controller when the 9BITRIS bit in the UARTRIS (page 264) register is set.</td> </tr> </tbody> </table>	Value	Description	0	The 9BITRIS interrupt is suppressed and not sent to the interrupt controller.	1	An interrupt is sent to the interrupt controller when the 9BITRIS bit in the UARTRIS (page 264) register is set.
Value	Description									
0	The 9BITRIS interrupt is suppressed and not sent to the interrupt controller.									
1	An interrupt is sent to the interrupt controller when the 9BITRIS bit in the UARTRIS (page 264) register is set.									
11	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
10	OEIM	RW	0	<p>UART Overrun Error Interrupt Mask</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The OERIS interrupt is suppressed and not sent to the interrupt controller.</td> </tr> <tr> <td>1</td> <td>An interrupt is sent to the interrupt controller when the OERIS bit in the UARTRIS (page 264) register is set.</td> </tr> </tbody> </table>	Value	Description	0	The OERIS interrupt is suppressed and not sent to the interrupt controller.	1	An interrupt is sent to the interrupt controller when the OERIS bit in the UARTRIS (page 264) register is set.
Value	Description									
0	The OERIS interrupt is suppressed and not sent to the interrupt controller.									
1	An interrupt is sent to the interrupt controller when the OERIS bit in the UARTRIS (page 264) register is set.									

Bit/Field	Name	Type	Reset	Description						
9	BEIM	RW	0	<p>UART Break Error Interrupt Mask</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The BERIS interrupt is suppressed and not sent to the interrupt controller.</td> </tr> <tr> <td>1</td> <td>An interrupt is sent to the interrupt controller when the BERIS bit in the UARTRIS (page 264) register is set.</td> </tr> </tbody> </table>	Value	Description	0	The BERIS interrupt is suppressed and not sent to the interrupt controller.	1	An interrupt is sent to the interrupt controller when the BERIS bit in the UARTRIS (page 264) register is set.
Value	Description									
0	The BERIS interrupt is suppressed and not sent to the interrupt controller.									
1	An interrupt is sent to the interrupt controller when the BERIS bit in the UARTRIS (page 264) register is set.									
8	PEIM	RW	0	<p>UART Parity Error Interrupt Mask</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The PERIS interrupt is suppressed and not sent to the interrupt controller.</td> </tr> <tr> <td>1</td> <td>An interrupt is sent to the interrupt controller when the PERIS bit in the UARTRIS (page 264) register is set.</td> </tr> </tbody> </table>	Value	Description	0	The PERIS interrupt is suppressed and not sent to the interrupt controller.	1	An interrupt is sent to the interrupt controller when the PERIS bit in the UARTRIS (page 264) register is set.
Value	Description									
0	The PERIS interrupt is suppressed and not sent to the interrupt controller.									
1	An interrupt is sent to the interrupt controller when the PERIS bit in the UARTRIS (page 264) register is set.									
7	FEIM	RW	0	<p>UART Framing Error Interrupt Mask</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The FERIS interrupt is suppressed and not sent to the interrupt controller.</td> </tr> <tr> <td>1</td> <td>An interrupt is sent to the interrupt controller when the FERIS bit in the UARTRIS (page 264) register is set.</td> </tr> </tbody> </table>	Value	Description	0	The FERIS interrupt is suppressed and not sent to the interrupt controller.	1	An interrupt is sent to the interrupt controller when the FERIS bit in the UARTRIS (page 264) register is set.
Value	Description									
0	The FERIS interrupt is suppressed and not sent to the interrupt controller.									
1	An interrupt is sent to the interrupt controller when the FERIS bit in the UARTRIS (page 264) register is set.									
6	RTIM	RW	0	<p>UART Receive Time-Out Interrupt Mask</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The RTRIS interrupt is suppressed and not sent to the interrupt controller.</td> </tr> <tr> <td>1</td> <td>An interrupt is sent to the interrupt controller when the RTRIS bit in the UARTRIS (page 264) register is set.</td> </tr> </tbody> </table>	Value	Description	0	The RTRIS interrupt is suppressed and not sent to the interrupt controller.	1	An interrupt is sent to the interrupt controller when the RTRIS bit in the UARTRIS (page 264) register is set.
Value	Description									
0	The RTRIS interrupt is suppressed and not sent to the interrupt controller.									
1	An interrupt is sent to the interrupt controller when the RTRIS bit in the UARTRIS (page 264) register is set.									
5	TXIM	RW	0	<p>UART Transmit Interrupt Mask</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The TXRIS interrupt is suppressed and not sent to the interrupt controller.</td> </tr> <tr> <td>1</td> <td>An interrupt is sent to the interrupt controller when the TXRIS bit in the UARTRIS (page 264) register is set.</td> </tr> </tbody> </table>	Value	Description	0	The TXRIS interrupt is suppressed and not sent to the interrupt controller.	1	An interrupt is sent to the interrupt controller when the TXRIS bit in the UARTRIS (page 264) register is set.
Value	Description									
0	The TXRIS interrupt is suppressed and not sent to the interrupt controller.									
1	An interrupt is sent to the interrupt controller when the TXRIS bit in the UARTRIS (page 264) register is set.									

Bit/Field	Name	Type	Reset	Description						
4	RXIM	RW	0	<p>UART Receive Interrupt Mask</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The RXRIS interrupt is suppressed and not sent to the interrupt controller.</td> </tr> <tr> <td>1</td> <td>An interrupt is sent to the interrupt controller when the RXRIS bit in the UARTRIS (page 264) register is set.</td> </tr> </tbody> </table>	Value	Description	0	The RXRIS interrupt is suppressed and not sent to the interrupt controller.	1	An interrupt is sent to the interrupt controller when the RXRIS bit in the UARTRIS (page 264) register is set.
Value	Description									
0	The RXRIS interrupt is suppressed and not sent to the interrupt controller.									
1	An interrupt is sent to the interrupt controller when the RXRIS bit in the UARTRIS (page 264) register is set.									
3:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
1	CTSIM	RW	0	<p>UART Clear to Send Modem Interrupt Mask</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The CTSRIS interrupt is suppressed and not sent to the interrupt controller.</td> </tr> <tr> <td>1</td> <td>An interrupt is sent to the interrupt controller when the CTSRIS bit in the UARTRIS (page 264) register is set.</td> </tr> </tbody> </table> <p>This bit is implemented only on UART1 and is reserved for UART0 and UART2.</p>	Value	Description	0	The CTSRIS interrupt is suppressed and not sent to the interrupt controller.	1	An interrupt is sent to the interrupt controller when the CTSRIS bit in the UARTRIS (page 264) register is set.
Value	Description									
0	The CTSRIS interrupt is suppressed and not sent to the interrupt controller.									
1	An interrupt is sent to the interrupt controller when the CTSRIS bit in the UARTRIS (page 264) register is set.									
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						

21.7.12 UART Masked Interrupt Status (UARTMIS)

- UART0 base: 0x4000.C000
- UART1 base: 0x4000.D000
- UART2 base: 0x4000.E000
- UART3 base: 0x4000.F000
- UART4 base: 0x4001.0000
- UART5 base: 0x4001.1000
- UART6 base: 0x4001.2000
- UART7 base: 0x4001.3000
- Offset 0x040
- Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description						
31:13	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
12	9BITMIS	RO	0	<p>9-Bit Mode Masked Interrupt Status</p> <table> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>An interrupt has not occurred or is masked.</td> </tr> <tr> <td>1</td> <td>An unmasked interrupt was signaled due to a receive address match.</td> </tr> </table> <p>This bit is cleared by writing a 1 to the 9BITIC bit in the UARTICR (page 274) register.</p>	Value	Description	0	An interrupt has not occurred or is masked.	1	An unmasked interrupt was signaled due to a receive address match.
Value	Description									
0	An interrupt has not occurred or is masked.									
1	An unmasked interrupt was signaled due to a receive address match.									
11	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
10	OEMIS	RO	0	<p>UART Overrun Error Masked Interrupt Status</p> <table> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>An interrupt has not occurred or is masked.</td> </tr> <tr> <td>1</td> <td>An unmasked interrupt was signaled due to an overrun error.</td> </tr> </table> <p>This bit is cleared by writing a 1 to the 0EIC bit in the UARTICR (page 274) register.</p>	Value	Description	0	An interrupt has not occurred or is masked.	1	An unmasked interrupt was signaled due to an overrun error.
Value	Description									
0	An interrupt has not occurred or is masked.									
1	An unmasked interrupt was signaled due to an overrun error.									

Bit/Field	Name	Type	Reset	Description						
9	BEMIS	RO	0	<p>UART Break Error Masked Interrupt Status</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>An interrupt has not occurred or is masked.</td> </tr> <tr> <td>1</td> <td>An unmasked interrupt was signaled due to a break error.</td> </tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the BEIC bit in the UARTICR (page 274) register.</p>	Value	Description	0	An interrupt has not occurred or is masked.	1	An unmasked interrupt was signaled due to a break error.
Value	Description									
0	An interrupt has not occurred or is masked.									
1	An unmasked interrupt was signaled due to a break error.									
8	PEMIS	RO	0	<p>UART Parity Error Masked Interrupt Status</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>An interrupt has not occurred or is masked.</td> </tr> <tr> <td>1</td> <td>An unmasked interrupt was signaled due to a parity error.</td> </tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the PEIC bit in the UARTICR (page 274) register.</p>	Value	Description	0	An interrupt has not occurred or is masked.	1	An unmasked interrupt was signaled due to a parity error.
Value	Description									
0	An interrupt has not occurred or is masked.									
1	An unmasked interrupt was signaled due to a parity error.									
7	FEMIS	RO	0	<p>UART Framing Error Masked Interrupt Status</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>An interrupt has not occurred or is masked.</td> </tr> <tr> <td>1</td> <td>An unmasked interrupt was signaled due to a framing error.</td> </tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the FEIC bit in the UARTICR (page 274) register.</p>	Value	Description	0	An interrupt has not occurred or is masked.	1	An unmasked interrupt was signaled due to a framing error.
Value	Description									
0	An interrupt has not occurred or is masked.									
1	An unmasked interrupt was signaled due to a framing error.									
6	RTMIS	RO	0	<p>UART Receive Time-Out Masked Interrupt Status</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>An interrupt has not occurred or is masked.</td> </tr> <tr> <td>1</td> <td>An unmasked interrupt was signaled due to a receive time out.</td> </tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the RTIC bit in the UARTICR (page 274) register. For receive timeout, the RTIM bit in the UARTIM (page 268) register must be set to see the RTMIS status.</p>	Value	Description	0	An interrupt has not occurred or is masked.	1	An unmasked interrupt was signaled due to a receive time out.
Value	Description									
0	An interrupt has not occurred or is masked.									
1	An unmasked interrupt was signaled due to a receive time out.									

Bit/Field	Name	Type	Reset	Description						
5	TXMIS	RO	0	<p>UART Transmit Masked Interrupt Status</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>An interrupt has not occurred or is masked.</td></tr> <tr> <td>1</td><td>An unmasked interrupt was signaled due to passing through the specified transmit FIFO level (if the EOT bit is clear) or due to the transmission of the last data bit (if the EOT bit is set).</td></tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the TXIC bit in the UARTICR (page 274) register or by writing data to the transmit FIFO until it becomes greater than the trigger level, if the FIFO is enabled, or by writing a single byte if the FIFO is disabled.</p>	Value	Description	0	An interrupt has not occurred or is masked.	1	An unmasked interrupt was signaled due to passing through the specified transmit FIFO level (if the EOT bit is clear) or due to the transmission of the last data bit (if the EOT bit is set).
Value	Description									
0	An interrupt has not occurred or is masked.									
1	An unmasked interrupt was signaled due to passing through the specified transmit FIFO level (if the EOT bit is clear) or due to the transmission of the last data bit (if the EOT bit is set).									
4	RXMIS	RO	0	<p>UART Receive Masked Interrupt Status</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>An interrupt has not occurred or is masked.</td></tr> <tr> <td>1</td><td>An unmasked interrupt was signaled due to passing through the specified receive FIFO level.</td></tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the RXIC bit in the UARTICR (page 274) register or by reading data from the receive FIFO until it becomes less than the trigger level, if the FIFO is enabled, or by reading a single byte if the FIFO is disabled.</p>	Value	Description	0	An interrupt has not occurred or is masked.	1	An unmasked interrupt was signaled due to passing through the specified receive FIFO level.
Value	Description									
0	An interrupt has not occurred or is masked.									
1	An unmasked interrupt was signaled due to passing through the specified receive FIFO level.									
3:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
1	CTSMIS	RO	0	<p>UART Clear to Send Modem Masked Interrupt Status</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>An interrupt has not occurred or is masked.</td></tr> <tr> <td>1</td><td>An unmasked interrupt was signaled due to Clear to Send.</td></tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the CTSIC bit in the UARTICR (page 274) register. This bit is implemented only on UART1 and is reserved for UART0 and UART2.</p>	Value	Description	0	An interrupt has not occurred or is masked.	1	An unmasked interrupt was signaled due to Clear to Send.
Value	Description									
0	An interrupt has not occurred or is masked.									
1	An unmasked interrupt was signaled due to Clear to Send.									
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						

21.7.13 UART Interrupt Clear (UARTICR)

- UART0 base: 0x4000.C000
- UART1 base: 0x4000.D000
- UART2 base: 0x4000.E000
- UART3 base: 0x4000.F000
- UART4 base: 0x4001.0000
- UART5 base: 0x4001.1000
- UART6 base: 0x4001.2000
- UART7 base: 0x4001.3000
- Offset 0x044
- Type W1C, reset 0x0000.0000

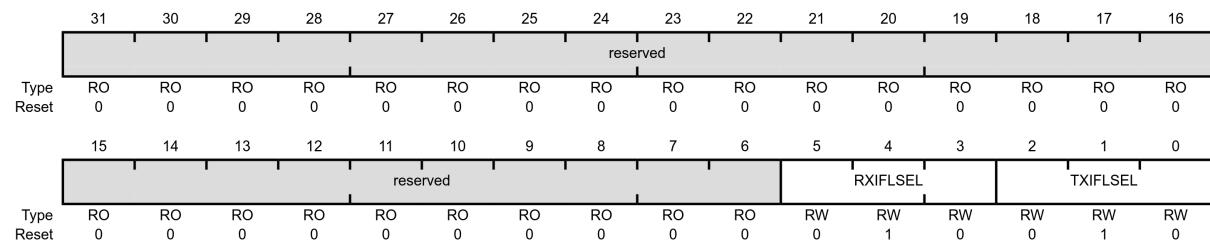
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved			9BITIC	reserved	OEIC	BEIC	PEIC	FEIC	RTIC	TXIC	RXIC	reserved		CTSMIC	reserved
Type	RO	RO	RO	RW	RO	W1C	RO	RO	W1C	RO						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:13	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	9BITIC	RW	0	9-Bit Mode Interrupt Clear Writing a 1 to this bit clears the 9BITRIS bit in the UARTRIS (page 264) register and the 9BITMIS bit in the UARTMIS (page 271) register.
11	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	OEIC	W1C	0	Overrun Error Interrupt Clear Writing a 1 to this bit clears the OERIS bit in the UARTRIS (page 264) register and the OEMIS bit in the UARTMIS (page 271) register.
9	BEIC	W1C	0	Break Error Interrupt Clear Writing a 1 to this bit clears the BERIS bit in the UARTRIS (page 264) register and the BEMIS bit in the UARTMIS (page 271) register.
8	PEIC	W1C	0	Parity Error Interrupt Clear Writing a 1 to this bit clears the PERIS bit in the UARTRIS (page 264) register and the PEMIS bit in the UARTMIS (page 271) register.

Bit/Field	Name	Type	Reset	Description
7	FEIC	W1C	0	Framing Error Interrupt Clear Writing a 1 to this bit clears the FERIS bit in the UARTRIS (page 264) register and the FEMIS bit in the UARTMIS (page 271) register.
6	RTIC	W1C	0	Receive Time-Out Interrupt Clear Writing a 1 to this bit clears the RTRIS bit in the UARTRIS (page 264) register and the RTMIS bit in the UARTMIS (page 271) register.
5	TXIC	W1C	0	Transmit Interrupt Clear Writing a 1 to this bit clears the TXRIS bit in the UARTRIS (page 264) register and the TXIM bit in the UARTMIS (page 271) register.
4	RXIC	W1C	0	Receive Interrupt Clear Writing a 1 to this bit clears the RXRIS bit in the UARTRIS (page 264) register and the RXIM bit in the UARTMIS (page 271) register.
3:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	CTSMIC	W1C	0	UART Clear to Send Modem Interrupt Clear Writing a 1 to this bit clears the CTSRIS bit in the UARTRIS (page 264) register and the CTSMIS bit in the UARTMIS (page 271) register. This bit is implemented only on UART1 and is reserved for UART0 and UART2.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

21.7.14 UART Interrupt FIFO Level Select (UARTIFLS)

- UART0 base: 0x4000.C000
- UART1 base: 0x4000.D000
- UART2 base: 0x4000.E000
- UART3 base: 0x4000.F000
- UART4 base: 0x4001.0000
- UART5 base: 0x4001.1000
- UART6 base: 0x4001.2000
- UART7 base: 0x4001.3000
- Offset 0x034
- Type RW, reset 0x0000.0012



Bit/Field	Name	Type	Reset	Description														
31:6	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.														
5:3	RXIFLSEL	RW	0x2	<p>UART Receive Interrupt FIFO Level Select</p> <p>The trigger points for the receive interrupt are as follows:</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>RX FIFO $\geq \frac{1}{8}$ full</td> </tr> <tr> <td>0x1</td> <td>RX FIFO $\geq \frac{1}{4}$ full</td> </tr> <tr> <td>0x2</td> <td>RX FIFO $\geq \frac{1}{2}$ full (default)</td> </tr> <tr> <td>0x3</td> <td>RX FIFO $\geq \frac{3}{4}$ full</td> </tr> <tr> <td>0x4</td> <td>RX FIFO $\geq \frac{7}{8}$ full</td> </tr> <tr> <td>0x5 - 0x7</td> <td>Reserved</td> </tr> </tbody> </table>	Value	Description	0x0	RX FIFO $\geq \frac{1}{8}$ full	0x1	RX FIFO $\geq \frac{1}{4}$ full	0x2	RX FIFO $\geq \frac{1}{2}$ full (default)	0x3	RX FIFO $\geq \frac{3}{4}$ full	0x4	RX FIFO $\geq \frac{7}{8}$ full	0x5 - 0x7	Reserved
Value	Description																	
0x0	RX FIFO $\geq \frac{1}{8}$ full																	
0x1	RX FIFO $\geq \frac{1}{4}$ full																	
0x2	RX FIFO $\geq \frac{1}{2}$ full (default)																	
0x3	RX FIFO $\geq \frac{3}{4}$ full																	
0x4	RX FIFO $\geq \frac{7}{8}$ full																	
0x5 - 0x7	Reserved																	

Bit/Field	Name	Type	Reset	Description														
2:0	TXIFLSEL	RW	0x2	<p>UART Transmit Interrupt FIFO Level Select</p> <p>The trigger points for the transmit interrupt are as follows:</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>TX FIFO $\leq \frac{7}{8}$ empty</td> </tr> <tr> <td>0x1</td> <td>TX FIFO $\leq \frac{3}{4}$ empty</td> </tr> <tr> <td>0x2</td> <td>TX FIFO $\leq \frac{1}{2}$ empty (default)</td> </tr> <tr> <td>0x3</td> <td>TX FIFO $\leq \frac{1}{4}$ empty</td> </tr> <tr> <td>0x4</td> <td>TX FIFO $\leq \frac{1}{8}$ empty</td> </tr> <tr> <td>0x5 - 0x7</td> <td>Reserved</td> </tr> </tbody> </table> <p>Note: If the EOT bit in UARTCTL (page 255) is set, the transmit interrupt is generated once the FIFO is completely empty and all data including stop bits have left the transmit serializer. In this case, the setting of TXIFLSEL is ignored.</p>	Value	Description	0x0	TX FIFO $\leq \frac{7}{8}$ empty	0x1	TX FIFO $\leq \frac{3}{4}$ empty	0x2	TX FIFO $\leq \frac{1}{2}$ empty (default)	0x3	TX FIFO $\leq \frac{1}{4}$ empty	0x4	TX FIFO $\leq \frac{1}{8}$ empty	0x5 - 0x7	Reserved
Value	Description																	
0x0	TX FIFO $\leq \frac{7}{8}$ empty																	
0x1	TX FIFO $\leq \frac{3}{4}$ empty																	
0x2	TX FIFO $\leq \frac{1}{2}$ empty (default)																	
0x3	TX FIFO $\leq \frac{1}{4}$ empty																	
0x4	TX FIFO $\leq \frac{1}{8}$ empty																	
0x5 - 0x7	Reserved																	

21.8 Initialisation and configuration

To initialise the UART, the follow steps are necessary:

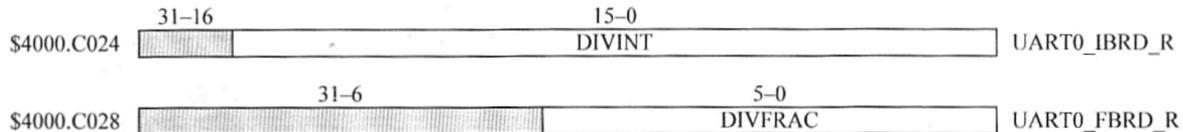
1. Enable the UART module with the **RCGCUART** ([page 247](#)) register.
2. Enable the clock to the appropriate GPIO module via the **RCGCGPIO** ([page 133](#)) register.
3. Set the **GPIOAFSEL** ([page 148](#)) bits for the appropriate pins.
4. Configure the GPIO current (in mA) level and slew rate as specified for the mode selected.
5. Configure the Port Mux Control (PMCn) fields in the **GPIOPCTL** ([page 158](#)) register to assign the UART signals to the appropriate pins.
6. Write the desired serial parameters to the **UARTLCRH** ([page 251](#)) register.
7. Configure the UART clock source by writing to the **UARTCC** ([page 253](#)) register. Bits 3:0 set to 0 to use system clock.
8. Optionally, configure the µDMA channel and enable the DMA options in the **UARTDMACTL** ([page 254](#)) register.

21.9 UART baud rate settings

- When programming the UART baud rate, you need to calculate the baud-rate divider (BRD) first.
- Then the **UARTIBRD** ([page 249](#)) and **UARTFBRD** ([page 250](#)) registers are determined.
- Finally, write to the **UART Line Control (UARTLCRH)** ([page 251](#)) register.

21.10 Baud rate setting

- Registers **UARTIBRD** ([page 249](#)) and **UARTFBRD** ([page 250](#)) set the baud rate.



- 22 (16.6) bit binary fixed point value with 2^{-6} resolution.
 - **DIVINT** (integer -16 binary places)
 - **DIVFRAC** (fractional -6 binary places)
- **Baud16** clock is created with a frequency of $\frac{\text{Bus clock frequency}}{\text{Divider}}$
- **Baud16** clock is 16x of Baud rate.

$$\text{Baud rate divider} = \text{Bus clock} \div 16 \div \text{Baud rate}$$

21.10.1 Example

If the bus clock is 8 MHz, then the baud rate required is 19200 bits s^{-1} . Then:

$$\text{Baud rate divider} = 8,000,000 \div 16 \div 19200 = 26.04167$$

$$\text{Integer part : } 26 = 11010_2 = 0x1A$$

$$\text{Fractional part : } 0.04167$$

21.10.1.1 How to convert fractions in decimal to binary?

0.04167 to binary:

$$0.04167 * 2 = 0.08334 \rightarrow 0$$

$$0.08334 * 2 = 0.16668 \rightarrow 0$$

$$0.16668 * 2 = 0.33336 \rightarrow 0$$

$$0.33336 * 2 = 0.66672 \rightarrow 0$$

$$0.66672 * 2 = 1.33344 \rightarrow 1$$

$$0.33344 * 2 = 0.66688 \rightarrow 0$$

$$0.66688 * 2 = 1.33376 \rightarrow 1$$

Hence:

$$0.04167 = 0.0000101_2 \text{ or } 0.000011_2 \text{ (6 places)}$$

Faster way:

1. Multiply the fractional part by 2 to the power of the number of decimal places you need, i.e.

$$\text{Fractional part} \times 2^n$$

Where n is the number of decimal places needed.

2. Round the result to the nearest whole number and convert the result to binary.

Example:

$$\begin{aligned} 0.04167 \times 2^6 &= 2.667 \\ &\approx 3_{10} = 000011_2 = 0x03_{16} \end{aligned}$$

21.10.1.2 Continuing the example

$$26.04167 = 11010.000011_2$$

To have 19200 baud rate:

- Set UARTIBRD ([page 249](#)) to 11010 (maximum 16 places)
- Set UARTFBRD ([page 250](#)) to 000011 (maximum 6 places)

However, there is an error, as:

$$11010.000011_2 = 26\frac{3}{4} = 26.046875$$

$$\frac{26.046875}{26.04167} = 1.0002 \text{ or } 0.02\% \text{ error}$$

The baud rate must be within 5% for the channel to operate properly, otherwise a framing error will occur when the stop bit is incorrectly captured.

21.10.2 Procedure

With the baud-rate divider (BRD) values determined, the UART configuration is written to the module in the following order:

1. Disable the UART by clearing the UARTEN bit in the UARTCTL ([page 255](#)) register.
2. Write the integer portion of the BRD to the UARTIBRD ([page 249](#)) register.
3. Write the fractional portion of the BRD to the UARTFBRD ([page 250](#)) register.

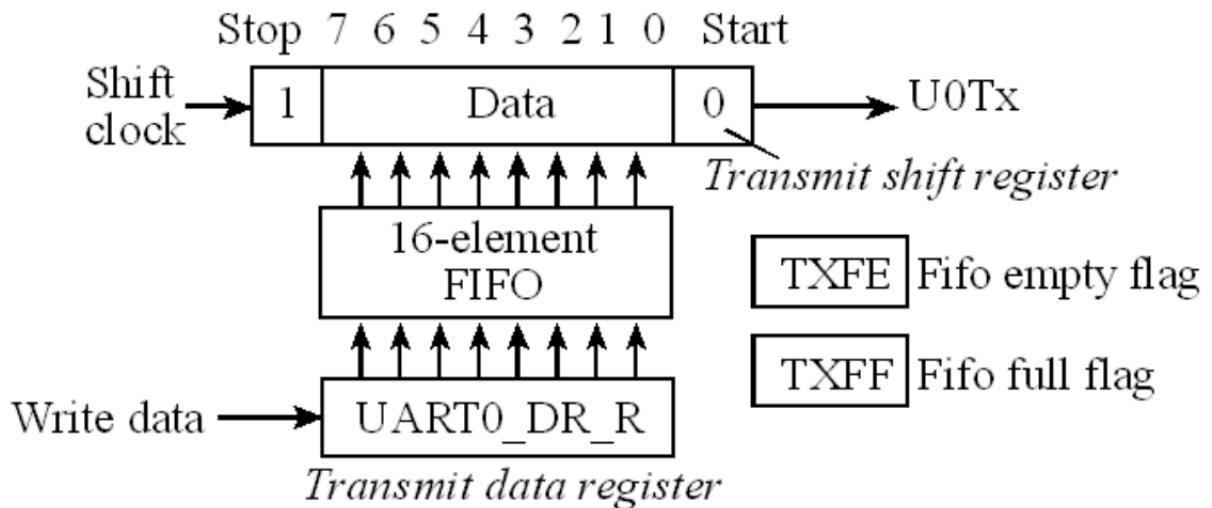
21.11 UART transmission procedure

- Enable the UART by setting the **UARTEN** bit in the **UARTCTL** (page 255) register.

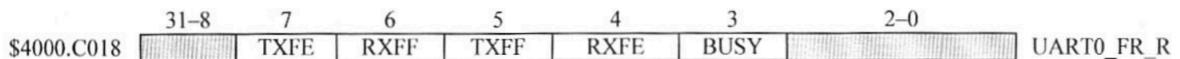


- The UART is configured for transmitting or receiving via the **RXE** and **TXE** bits of the **UARTCTL** (page 255) register.
- Transmitting and receiving are both enabled out of reset. Before any control registers are programmed, the UART must be disabled by clearing the **UARTEN** bit in **UARTCTL** (page 255).
- If the UART is disabled during a TX or RX operation, the current transaction is completed prior to the UART stopping.

21.12 UART transmission sequence



- Program reads **UARTFR** (page 262) that **TXFF** = 0. It will wait if **TXFF** = 1.

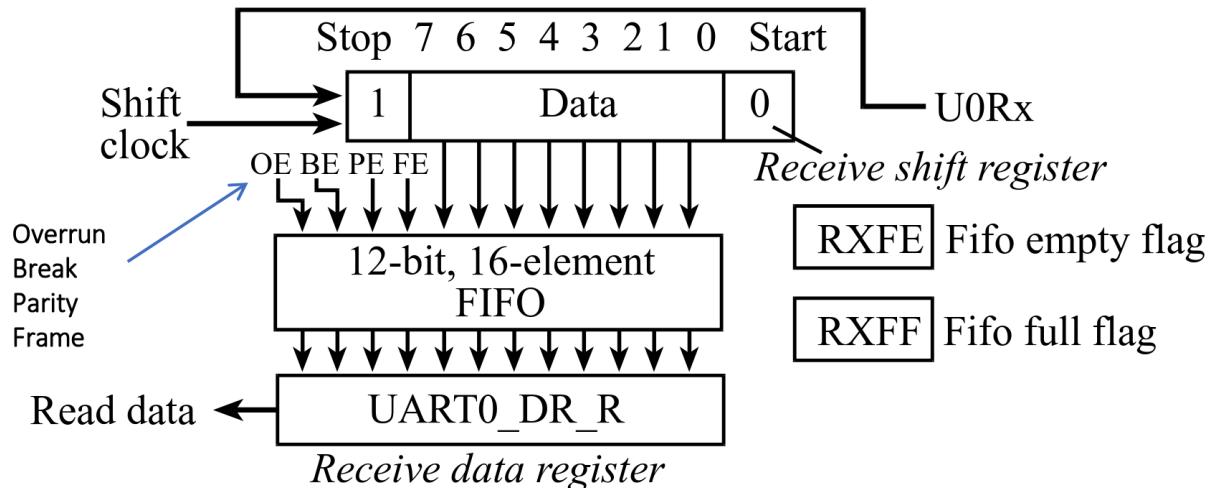


- New byte written to the **UARTDR** (page 260) register, which is then placed in the FIFO.

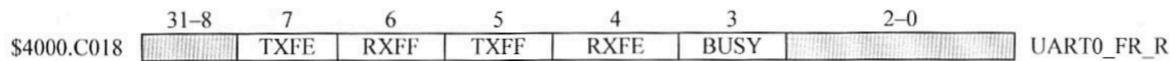


- 10 bit transmission shift register includes the start bit, 8 data bits, and 1 stop bit.
- The data frame is shifted out by the start bit, then from least significant bit (LSB) to most significant bit (MSB), then 1 stop bit, to the U0Tx line.

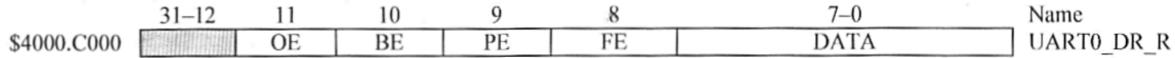
21.13 UART receiving sequence



1. 10 bit receiving shift register and 16 element FIFO, both 12 bits wide.
2. The UARTDR (page 260) register is read only when RXE bit 9 of UARTCTL (page 255) is enabled.
3. Bits from U0Rx is shifted in with the start bit, 8 data bits and then the stop bit.
4. Start and stop bits are removed and checked for framing error.
5. Program reads UARTFR (page 262) that RXFE = 0, which means the FIFO is not empty, and new data has reached the receiver.



6. 8 bits of data and 4 bits of status are put into the UARTDR (page 260) register.



7. Read the UARTDR (page 260) register bits 7:0 for the data and check the status condition bits 11:8 for errors.

22 TM4C123GH6PM SSI

The TM4C123GH6PM SSI modules have the follow features:

- Has 4 synchronous serial interface (SSI) modules.
- Programmable interface operation for Freescale SPI, MICROWIRE, or Texas Instruments.
- Synchronous serial interfaces.
- Master or slave operation.
- Programmable clock bit rate and prescaler.
- Separate transmission and reception FIFOs, each 16 bits wide and 8 locations deep.
- Programmable data frame size from 4 to 16 bits.
- Internal loopback test mode for diagnostics or debug testing.
- Standard FIFO-based interrupts and End-of-Transmission interrupt.
- Efficient transfers using Micro Direct Memory Access Controller (μ DMA).
 - Separate channels for transmission and reception.
 - Receive “single request asserted” when data is in the FIFO, and “burst request asserted” when the FIFO contains 4 entries.
 - Transmit “single request asserted” when there is space in the FIFO, and “burst request asserted” when 4 or more entries are available to be written in the FIFO.

22.1 SSI pin names

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type	Description
SSI0Clk	19	PA2 (2)	I/O	TTL	SSI module 0 clock.
SSI0Fss	20	PA3 (2)	I/O	TTL	SSI module 0 frame signal.
SSI0Rx	21	PA4 (2)	I	TTL	SSI module 0 receive.
SSI0Tx	22	PA5 (2)	O	TTL	SSI module 0 transmit.
SSI1Clk	30	PF2 (2)	I/O	TTL	SSI module 1 clock.
	61	PD0 (2)			
SSI1Fss	31	PF3 (2)	I/O	TTL	SSI module 1 frame signal.
	62	PD1 (2)			
SSI1Rx	28	PF0 (2)	I	TTL	SSI module 1 receive.
	63	PD2 (2)			
SSI1Tx	29	PF1 (2)	O	TTL	SSI module 1 transmit.
	64	PD3 (2)			
SSI2Clk	58	PB4 (2)	I/O	TTL	SSI module 2 clock.
SSI2Fss	57	PB5 (2)	I/O	TTL	SSI module 2 frame signal.
SSI2Rx	1	PB6 (2)	I	TTL	SSI module 2 receive.
SSI2Tx	4	PB7 (2)	O	TTL	SSI module 2 transmit.
SSI3Clk	61	PD0 (1)	I/O	TTL	SSI module 3 clock.
SSI3Fss	62	PD1 (1)	I/O	TTL	SSI module 3 frame signal.
SSI3Rx	63	PD2 (1)	I	TTL	SSI module 3 receive.
SSI3Tx	64	PD3 (1)	O	TTL	SSI module 3 transmit.

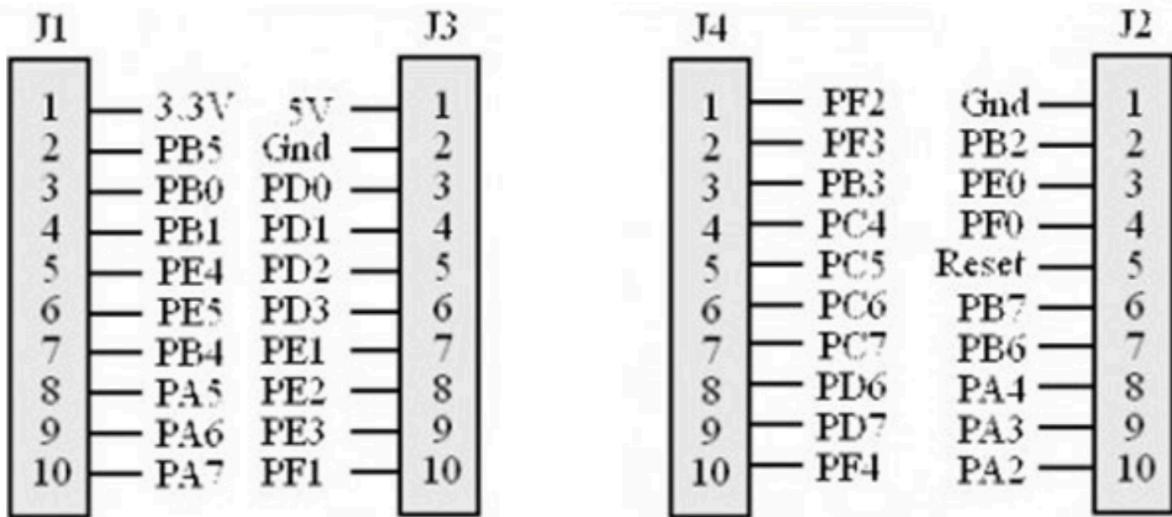
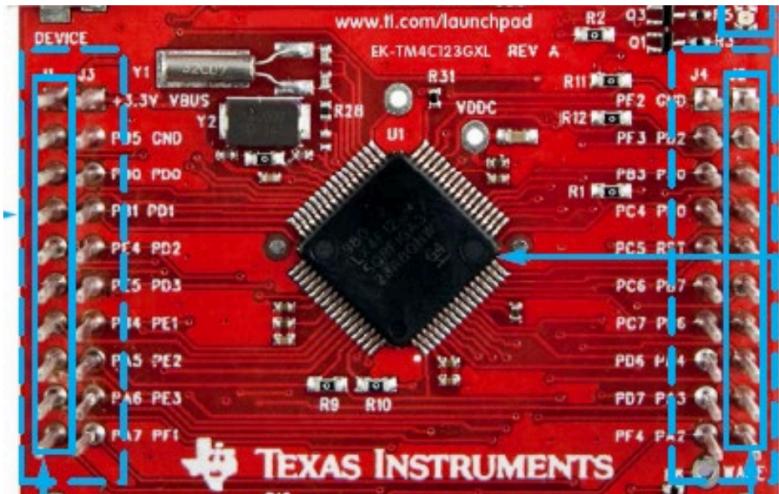
22.2 SSI module

SSI0	SSI0Tx → SSI0Rx ← SSI0Fss → SSI0Clk →
SSI1	SSI1Tx → SSI1Rx ← SSI1Fss → SSI1Clk →
SSI2	SSI2Tx → SSI2Rx ← SSI2Fss → SSI2Clk →
SSI3	SSI3Tx → SSI3Rx ← SSI3Fss → SSI3Clk →

SSI protocol has 4 I/O lines:

- SSI0Fss: Slave select is an optional negative logic control from master to slave to indicate that the channel is active.
- SSI0Clk: 50% duty cycle generated by the master.
- SSI0Tx: Data line driven by the master and received by the slave (master out, slave in, MOSI).
- SSI0Rx: Data line driven by the slave and received by the master (master in, slave out, MISO).

22.3 J connectors



22.3.1 J1 connector

J1 Pin	GPIO	Analog Function	On-board Function	Tiva C Series MCU Pin	GPIOCTL Register Setting													
		GPIO AMSEL			1	2	3	4	5	6	7	8	9	14	15			
1.01					3.3 V													
1.02	PB5	AIN11	-	57	-	SSI2Fss	-	M0PWM3	-	-	T1CCP1	CAN0Tx	-	-	-			
1.03	PB0	USB0ID	-	45	U1Rx	-	-	-	-	-	T2CCP0	-	-	-	-			
1.04	PB1	USB0VBUS	-	46	U1Tx	-	-	-	-	-	T2CCP1	-	-	-	-			
1.05	PE4	AIN9	-	59	U5Rx	-	I2C2SCL	M0PWM4	M1PWM2	-	-	CAN0Rx	-	-	-			
1.06	PE5	AIN8	-	60	U5Tx	-	I2C2SDA	M0PWM5	M1PWM3	-	-	CAN0Tx	-	-	-			
1.07	PB4	AIN10	-	58	-	SSI2Clk	-	M0PWM2	-	-	T1CCP0	CAN0Rx	-	-	-			
1.08	PA5	-	-	22	-	SSI0Tx	-	-	-	-	-	-	-	-	-			
1.09	PA6	-	-	23	-	-	I2C1SCL	-	M1PWM2	-	-	-	-	-	-			
1.10	PA7	-	-	24	-	-	I2C1SDA	-	M1PWM3	-	-	-	-	-	-			

22.3.2 J2 connector

J2 Pin	GPIO	Analog Function	On-board Function	Tiva C Series MCU Pin	GPIOCTL Register Setting														
		GPIO AMSEL			1	2	3	4	5	6	7	8	9	14	15				
2.01					GND														
2.02	PB2	-	-	47	-	-	I2C0SCL	-	-	-	T3CCP0	-	-	-	-				
2.03	PE0	AIN3	-	9	U7Rx	-	-	-	-	-	-	-	-	-	-				
2.04	PF0	-	USR_SW2/ WAKE (R1)	28	U1RTS	SSI1Rx	CAN0Rx	-	M1PWM4	PhA0	T0CCP0	NMI	C0o	-	-				
2.05					RESET														
2.06	PB7	-	-	4	-	SSI2Tx	-	M0PWM1	-	-	T0CCP1	-	-	-	-				
	PD1	AIN6	Connected for MSP430 Compatibility (R10)	62	SSI3Fss	SSI1Fss	I2C3SDA	M0PWM7	M1PWM1	-	WT2CCP1	-	-	-	-				
2.07	PB6	-	-	1	-	SSI2Rx	-	M0PWM0	-	-	T0CCP0	-	-	-	-				
	PD0	AIN7	Connected for MSP430 Compatibility (R9)	61	SSI3Clk	SSI1Clk	I2C3SCL	M0PWM6	M1PWM0	-	WT2CCP2	-	-	-	-				
2.08	PA4	-	-	21	-	SSI0Rx	-	-	-	-	-	-	-	-	-	-	-	-	
2.09	PA3	-	-	20	-	SSI0Fss	-	-	-	-	-	-	-	-	-	-	-	-	
2.10	PA2	-	-	19	-	SSI0Clk	-	-	-	-	-	-	-	-	-	-	-	-	

22.3.3 J3 connector

J3 Pin	GPIO	Analog Function	On-board Function	Tiva C Series MCU Pin	GPIOCTL Register Setting														
		GPIO AMSEL			1	2	3	4	5	6	7	8	9	14	15				
3.01					5.0 V														
3.02					GND														
3.03	PD0	AIN7	-	61	SSI3Clk	SSI1Clk	I2C3SCL	M0PWM6	M1PWM0	-	WT2CCP2	-	-	-	-				
	PB6	-	Connected for MSP430 Compatibility (R9)	1	-	SSI2Rx	-	M0PWM0	-	-	T0CCP0	-	-	-	-				
3.04	PD1	AIN6	-	92	SSI3Fss	SSI1Fss	I2C3SDA	M0PWM7	M1PWM1	-	WT2CCP1	-	-	-	-				
	PB7	-	Connected for MSP430 Compatibility (R10)	4	-	SSI2Tx	-	M0PWM1	-	-	T0CCP1	-	-	-	-				
3.05	PD2	AIN5	-	63	SSI3Rx	SSI1Rx	-	M0FAULT0	-	-	WT3CCP0	USB0EPEN	-	-	-				
3.06	PD3	AIN4	-	64	SSI3Tx	SSI1Tx	-	-	-	-	WT3CCP1	USB0PFLT	-	-	-				
3.07	PE1	AIN2	-	8	U7Tx	-	-	-	-	-	-	-	-	-	-				
3.08	PE2	AIN1	-	7	-	-	-	-	-	-	-	-	-	-	-				
3.09	PE3	AIN0	-	6	-	-	-	-	-	-	-	-	-	-	-				
3.10	PF1	-	-	29	U1CTS	SSI1Tx	-	-	M1PWM5	-	T0CCP1	-	C1o	TRD1	-				

22.3.4 J4 connector

J4 Pin	GPIO	Analog Function	On-board Function	Tiva C Series MCU Pin	GPIOCTL Register Setting										
					1	2	3	4	5	6	7	8	9	14	15
4.01	PF2	-	Blue LED (R11)	30	-	SSI1Clk	-	M0FAULT0	M1PWM6	-	T1CCP0	-	-	-	TRD0
4.02	PF3	-	Green LED (R12)	31	-	SSI1Fss	CAN0Tx	-	M1PWM7	-	T1CCP1	-	-	-	TRCLK
4.03	PB3	-	-	48	-	-	I2C0SDA	-	-	-	T3CCP1	-	-	-	-
4.04	PC4	C1-	-	16	U4Rx	U1Rx	-	M0PWM6	-	IDX1	WT0CCP0	U1RTS	-	-	-
4.05	PC5	C1+	-	15	U4Tx	U1Tx	-	M0PWM7	-	PhA1	WT0CCP1	U1CTS	-	-	-
4.06	PC6	C0+	-	14	U3Rx	-	-	-	-	PhB1	WT1CCP0	USB0EPEN	-	-	-
4.07	PC7	C0-	-	13	U3Tx	-	-	-	-	-	WT1CCP1	USB0PFLT	-	-	-
4.08	PD6	-	-	53	U2Rx	-	-	-	-	PhA0	WT5CCP0	-	-	-	-
4.09	PD7	-	-	10	U2Tx	-	-	-	-	PhB0	WT5CCP1	NMI	-	-	-
4.10	PF4	-	USR_SW1 (R13)	5	-	-	-	-	M1FAULT0	IDX0	T2CCP0	USB0EPEN	-	-	-

22.4 SSI register map

Offset	Name	Type	Reset	Description
0x000	SSICR0	RW	0x0000.0000	SSI Control 0
0x004	SSICR1	RW	0x0000.0000	SSI Control 1
0x008	SSIDR	RW	0x0000.0000	SSI Data
0x00C	SSISR	RO	0x0000.0003	SSI Status
0x010	SSICPSR	RW	0x0000.0000	SSI Clock Prescale
0x014	SSIIM	RW	0x0000.0000	SSI Interrupt Mask
0x018	SSIRIS	RO	0x0000.0008	SSI Raw Interrupt Status
0x01C	SSIMIS	RO	0x0000.0000	SSI Masked Interrupt Status
0x020	SSIICR	W1C	0x0000.0000	SSI Interrupt Clear
0x024	SSIDMACTL	RW	0x0000.0000	SSI DMA Control
0xFC8	SSICC	RW	0x0000.0000	SSI Clock Configuration
0xFD0	SSIPeriphID4	RO	0x0000.0000	SSI Peripheral Identification 4
0xFD4	SSIPeriphID5	RO	0x0000.0000	SSI Peripheral Identification 5
0xFD8	SSIPeriphID6	RO	0x0000.0000	SSI Peripheral Identification 6
0xFDC	SSIPeriphID7	RO	0x0000.0000	SSI Peripheral Identification 7
0xFE0	SSIPeriphID0	RO	0x0000.0022	SSI Peripheral Identification 0
0xFE4	SSIPeriphID1	RO	0x0000.0000	SSI Peripheral Identification 1
0xFE8	SSIPeriphID2	RO	0x0000.0018	SSI Peripheral Identification 2
0xFEC	SSIPeriphID3	RO	0x0000.0001	SSI Peripheral Identification 3
0xFF0	SSIPCellID0	RO	0x0000.000D	SSI PrimeCell Identification 0
0xFF4	SSIPCellID1	RO	0x0000.00F0	SSI PrimeCell Identification 1

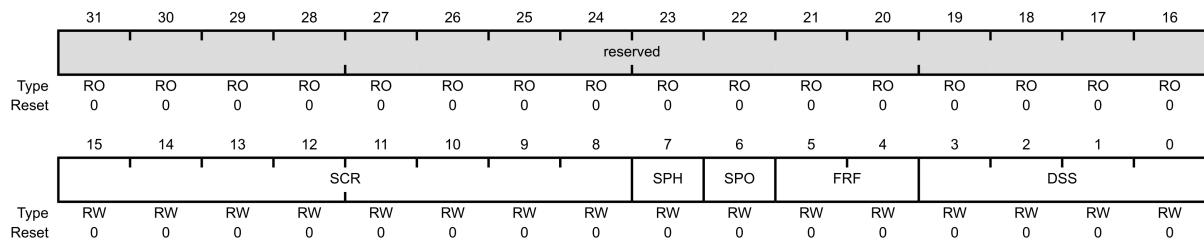
22.5 SSI register names

Address	31-6				3	2	1	0	Name
\$400F.E61C					SSI3	SSI2	SSI1	SSI0	SYSCTL_RCGCSSI_R
	31-16	15-8		7	6	5-4	3-0		
\$4000.8000		SCR		SPH	SPO	FRF	DSS		SSI0_CRO_R
	31-16	15-0							
\$4000.8008		Data							SSI0_DR_R
	7	6	5	4	3	2	1	0	
\$4000.8004				EOT		MS	SSE	LBM	SSI0_CRI_R
\$4000.800C				BSY	RFF	RNE	TNF	TFE	SSI0_SR_R
\$4000.8010				CPSDVSR					SSI0_CPSR_R
\$4000.8014					TXIM	RXIM	RTIM	RORIM	SSI0_IM_R
\$4000.8018					TXRIS	RXRIS	RTRIS	RORRIS	SSI0_RIS_R
\$4000.801C					TXMIS	RXMIS	RTMIS	RORMIS	SSI0_MIS_R
\$4000.8020							RTIC	RORIC	SSI0_ICR_R
\$4005.8420	SEL	SEL	SEL	SEL	SEL	SEL	SEL	SEL	GPIO_PORTA_AFSEL_R
\$4005.841C	DEN	DEN	DEN	DEN	DEN	DEN	DEN	DEN	GPIO_PORTA_DEN_R
\$4005.8400	DIR	DIR	DIR	DIR	DIR	DIR	DIR	DIR	GPIO_PORTA_DIR_R
\$400F.E608		GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA		SYSCTL_RCGCGPIO_R

22.6 Registers

22.6.1 SSI Control 0 (SSICR0)

- SSI0 base: 0x4000.8000
- SSI1 base: 0x4000.9000
- SSI2 base: 0x4000.A000
- SSI3 base: 0x4000.B000
- Offset 0x000
- Type RW, reset 0x0000.0000

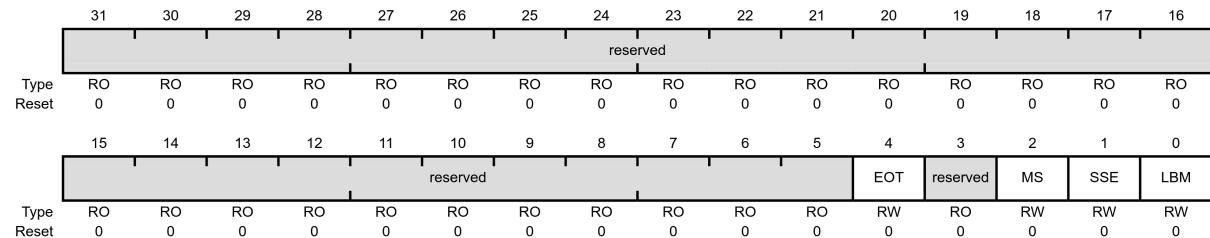


Bit/Field	Name	Type	Reset	Description						
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
15:8	SCR	RW	0x00	<p>SSI Serial Clock Rate</p> <p>This bit field is used to generate the transmit and receive bit rate of the SSI. The bit rate is:</p> $BR = \frac{\text{SysClk}}{\text{CPSDVSR} \times (1 + SCR)}$ <p>Where CPSDVSR is an even value from 2-254 programmed in the SSICPSR (page 293) register, and SCR is a value from 0-255.</p>						
7	SPH	RW	0	<p>SSI Serial Clock Phase</p> <p>This bit is only applicable to the Freescale SPI Format.</p> <p>The SPH control bit selects the clock edge that captures data and allows it to change state. This bit has the most impact on the first bit transmitted by either allowing or not allowing a clock transition before the first data capture edge.</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Data is captured on the first clock edge transition.</td> </tr> <tr> <td>1</td> <td>Data is captured on the second clock edge transition.</td> </tr> </tbody> </table>	Value	Description	0	Data is captured on the first clock edge transition.	1	Data is captured on the second clock edge transition.
Value	Description									
0	Data is captured on the first clock edge transition.									
1	Data is captured on the second clock edge transition.									

Bit/Field	Name	Type	Reset	Description																														
6	SPO	RW	0	<p>SSI Serial Clock Polarity</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>A steady state Low value is placed on the SSInClk pin.</td> </tr> <tr> <td>1</td> <td>A steady state High value is placed on the SSInClk pin when data is not being transferred.</td> </tr> </tbody> </table> <p>Note: If this bit is set, then software must also configure the GPIO port pin corresponding to the SSInClk signal as a pull-up in the GPIO Pull-Up Select (GPIOPUR) (page 145) register.</p>	Value	Description	0	A steady state Low value is placed on the SSInClk pin.	1	A steady state High value is placed on the SSInClk pin when data is not being transferred.																								
Value	Description																																	
0	A steady state Low value is placed on the SSInClk pin.																																	
1	A steady state High value is placed on the SSInClk pin when data is not being transferred.																																	
5:4	FRF	RW	0x0	<p>SSI Frame Format Select</p> <table> <thead> <tr> <th>Value</th> <th>Frame Format</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Freescale SPI Frame Format</td> </tr> <tr> <td>0x1</td> <td>Texas Instruments Synchronous Serial Frame Format</td> </tr> <tr> <td>0x2</td> <td>MICROWIRE Frame Format</td> </tr> <tr> <td>0x3</td> <td>Reserved</td> </tr> </tbody> </table>	Value	Frame Format	0x0	Freescale SPI Frame Format	0x1	Texas Instruments Synchronous Serial Frame Format	0x2	MICROWIRE Frame Format	0x3	Reserved																				
Value	Frame Format																																	
0x0	Freescale SPI Frame Format																																	
0x1	Texas Instruments Synchronous Serial Frame Format																																	
0x2	MICROWIRE Frame Format																																	
0x3	Reserved																																	
3:0	DSS	RW	0x0	<p>SSI Data Size Select</p> <table> <thead> <tr> <th>Value</th> <th>Data Size</th> </tr> </thead> <tbody> <tr> <td>0x0 - 0x2</td> <td>Reserved</td> </tr> <tr> <td>0x3</td> <td>4-bit data</td> </tr> <tr> <td>0x4</td> <td>5-bit data</td> </tr> <tr> <td>0x5</td> <td>6-bit data</td> </tr> <tr> <td>0x6</td> <td>7-bit data</td> </tr> <tr> <td>0x7</td> <td>8-bit data</td> </tr> <tr> <td>0x8</td> <td>9-bit data</td> </tr> <tr> <td>0x9</td> <td>10-bit data</td> </tr> <tr> <td>0xA</td> <td>11-bit data</td> </tr> <tr> <td>0xB</td> <td>12-bit data</td> </tr> <tr> <td>0xC</td> <td>13-bit data</td> </tr> <tr> <td>0xD</td> <td>14-bit data</td> </tr> <tr> <td>0xE</td> <td>15-bit data</td> </tr> <tr> <td>0xF</td> <td>16-bit data</td> </tr> </tbody> </table>	Value	Data Size	0x0 - 0x2	Reserved	0x3	4-bit data	0x4	5-bit data	0x5	6-bit data	0x6	7-bit data	0x7	8-bit data	0x8	9-bit data	0x9	10-bit data	0xA	11-bit data	0xB	12-bit data	0xC	13-bit data	0xD	14-bit data	0xE	15-bit data	0xF	16-bit data
Value	Data Size																																	
0x0 - 0x2	Reserved																																	
0x3	4-bit data																																	
0x4	5-bit data																																	
0x5	6-bit data																																	
0x6	7-bit data																																	
0x7	8-bit data																																	
0x8	9-bit data																																	
0x9	10-bit data																																	
0xA	11-bit data																																	
0xB	12-bit data																																	
0xC	13-bit data																																	
0xD	14-bit data																																	
0xE	15-bit data																																	
0xF	16-bit data																																	

22.6.2 SSI Control 1 (SSICR1)

- SSI0 base: 0x4000.8000
- SSI1 base: 0x4000.9000
- SSI2 base: 0x4000.A000
- SSI3 base: 0x4000.B000
- Offset 0x004
- Type RW, reset 0x0000.0000

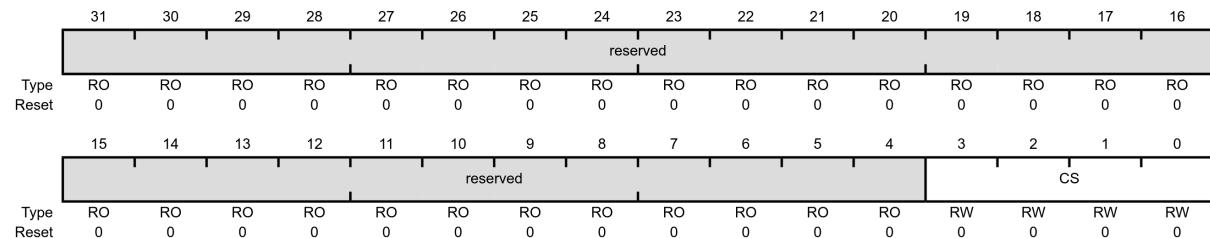


Bit/Field	Name	Type	Reset	Description						
31:5	reserved	RO	0x000.0	<p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p>						
4	EOT	RW	0	<p>End of Transmission</p> <p>This bit is only valid for Master mode devices and operations (MS = 0x0).</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The TXRIS interrupt indicates that the transmit FIFO is half full or less.</td> </tr> <tr> <td>1</td> <td>The End of Transmit interrupt mode for the TXRIS interrupt is enabled.</td> </tr> </tbody> </table> <p>Note: In Freescale SPI mode only, a condition can be created where an EOT interrupt is generated for every byte transferred even if the FIFO is full. If the EOT bit has been set to 0 in an integrated slave SSI and the µDMA has been configured to transfer data from this SSI to a Master SSI on the device using external loopback, an EOT interrupt is generated by the SSI slave for every byte even if the FIFO is full.</p>	Value	Description	0	The TXRIS interrupt indicates that the transmit FIFO is half full or less.	1	The End of Transmit interrupt mode for the TXRIS interrupt is enabled.
Value	Description									
0	The TXRIS interrupt indicates that the transmit FIFO is half full or less.									
1	The End of Transmit interrupt mode for the TXRIS interrupt is enabled.									
3	reserved	RO	0	<p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p>						

Bit/Field	Name	Type	Reset	Description						
2	MS	RW	0	<p>SSI Master/Slave Select</p> <p>This bit selects Master or Slave mode and can be modified only when the SSI is disabled (SSE=0).</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The SSI is configured as a master.</td> </tr> <tr> <td>1</td> <td>The SSI is configured as a slave.</td> </tr> </tbody> </table>	Value	Description	0	The SSI is configured as a master.	1	The SSI is configured as a slave.
Value	Description									
0	The SSI is configured as a master.									
1	The SSI is configured as a slave.									
1	SSE	RW	0	<p>SSI Synchronous Serial Port Enable</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>SSI operation is disabled.</td> </tr> <tr> <td>1</td> <td>SSI operation is enabled.</td> </tr> </tbody> </table> <p>Note: This bit must be cleared before any control registers are reprogrammed.</p>	Value	Description	0	SSI operation is disabled.	1	SSI operation is enabled.
Value	Description									
0	SSI operation is disabled.									
1	SSI operation is enabled.									
0	LBM	RW	0	<p>SSI Loopback Mode</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Normal serial port operation enabled.</td> </tr> <tr> <td>1</td> <td>Output of the transmit serial shift register is connected internally to the input of the receive serial shift register.</td> </tr> </tbody> </table>	Value	Description	0	Normal serial port operation enabled.	1	Output of the transmit serial shift register is connected internally to the input of the receive serial shift register.
Value	Description									
0	Normal serial port operation enabled.									
1	Output of the transmit serial shift register is connected internally to the input of the receive serial shift register.									

22.6.3 SSI Clock Configuration (SSICC)

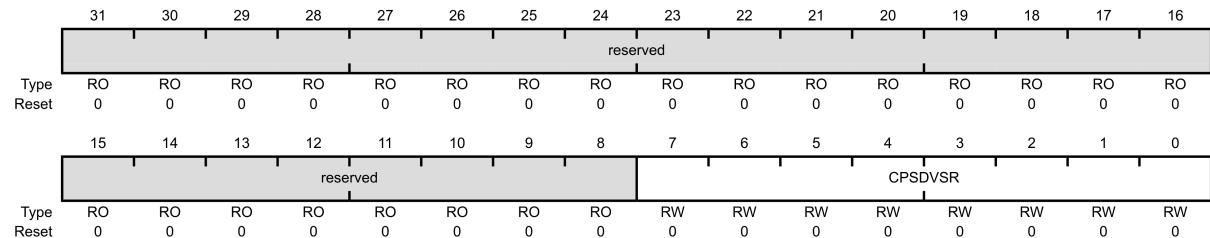
- SSI0 base: 0x4000.8000
- SSI1 base: 0x4000.9000
- SSI2 base: 0x4000.A000
- SSI3 base: 0x4000.B000
- Offset 0xFC8
- Type RW, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description										
31:4	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.										
3:0	CS	RW	0	SSI Baud Clock Source The following table specifies the source that generates the SSI baud clock: <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>System clock (based on clock source and divisor factor)</td> </tr> <tr> <td>0x1-0x4</td> <td>Reserved</td> </tr> <tr> <td>0x5</td> <td>PIO5C</td> </tr> <tr> <td>0x6-0xF</td> <td>Reserved</td> </tr> </tbody> </table>	Value	Description	0x0	System clock (based on clock source and divisor factor)	0x1-0x4	Reserved	0x5	PIO5C	0x6-0xF	Reserved
Value	Description													
0x0	System clock (based on clock source and divisor factor)													
0x1-0x4	Reserved													
0x5	PIO5C													
0x6-0xF	Reserved													

22.6.4 SSI Clock Prescale (SSICPSR)

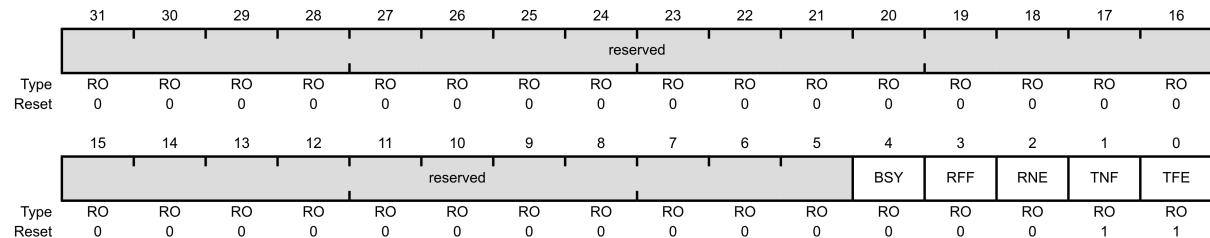
- SSI0 base: 0x4000.8000
- SSI1 base: 0x4000.9000
- SSI2 base: 0x4000.A000
- SSI3 base: 0x4000.B000
- Offset 0x010
- Type RW, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CPSDVSR	RW	0x00	SSI Clock Prescale Divisor This value must be an even number from 2 to 254, depending on the frequency of SSInClk. The LSB always returns 0 on reads.

22.6.5 SSI Status (SSISR)

- SSI0 base: 0x4000.8000
- SSI1 base: 0x4000.9000
- SSI2 base: 0x4000.A000
- SSI3 base: 0x4000.B000
- Offset 0x00C
- Type RO, reset 0x0000.0003

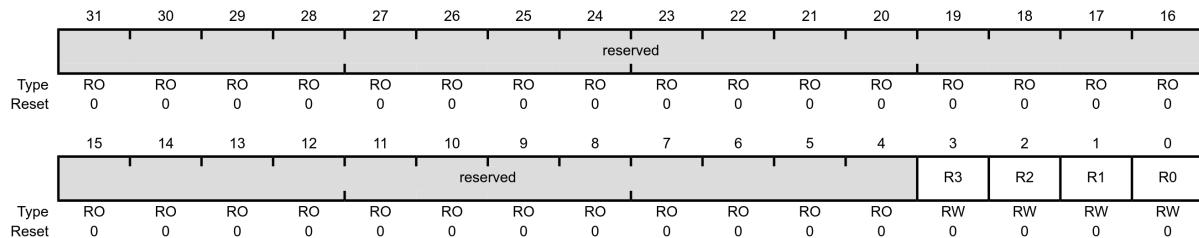


Bit/Field	Name	Type	Reset	Description						
31:5	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
4	BSY	RO	0	<p>SSI Busy Bit</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The SSI is idle.</td> </tr> <tr> <td>1</td> <td>The SSI is currently transmitting and/or receiving a frame, or the transmit FIFO is not empty.</td> </tr> </tbody> </table>	Value	Description	0	The SSI is idle.	1	The SSI is currently transmitting and/or receiving a frame, or the transmit FIFO is not empty.
Value	Description									
0	The SSI is idle.									
1	The SSI is currently transmitting and/or receiving a frame, or the transmit FIFO is not empty.									
3	RFF	RO	0	<p>SSI Receive FIFO Full</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The receive FIFO is not full.</td> </tr> <tr> <td>1</td> <td>The receive FIFO is full.</td> </tr> </tbody> </table>	Value	Description	0	The receive FIFO is not full.	1	The receive FIFO is full.
Value	Description									
0	The receive FIFO is not full.									
1	The receive FIFO is full.									
2	RNE	RO	0	<p>SSI Receive FIFO Not Empty</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The receive FIFO is empty.</td> </tr> <tr> <td>1</td> <td>The receive FIFO is not empty.</td> </tr> </tbody> </table>	Value	Description	0	The receive FIFO is empty.	1	The receive FIFO is not empty.
Value	Description									
0	The receive FIFO is empty.									
1	The receive FIFO is not empty.									
1	TNF	RO	1	<p>SSI Transmit FIFO Not Full</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The transmit FIFO is full.</td> </tr> <tr> <td>1</td> <td>The transmit FIFO is not full.</td> </tr> </tbody> </table>	Value	Description	0	The transmit FIFO is full.	1	The transmit FIFO is not full.
Value	Description									
0	The transmit FIFO is full.									
1	The transmit FIFO is not full.									

Bit/Field	Name	Type	Reset	Description						
0	TNE	RO	1	<p>SSI Transmit FIFO Empty</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The transmit FIFO is not empty.</td> </tr> <tr> <td>1</td> <td>The transmit FIFO is empty.</td> </tr> </tbody> </table>	Value	Description	0	The transmit FIFO is not empty.	1	The transmit FIFO is empty.
Value	Description									
0	The transmit FIFO is not empty.									
1	The transmit FIFO is empty.									

22.6.6 Synchronous Serial Interface Run Mode Clock Gating Control (RCGCSSI)

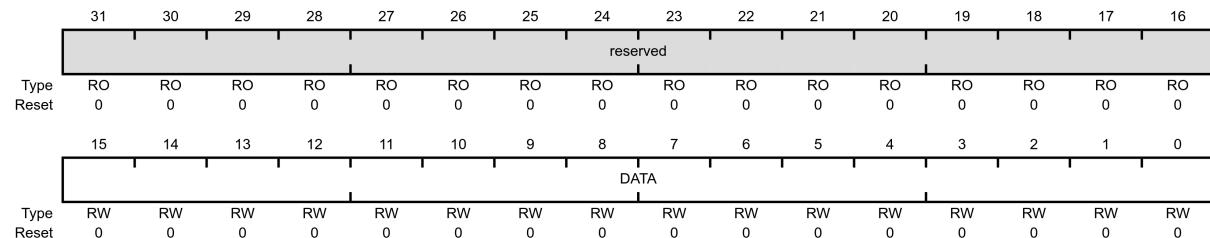
- Base 0x400F.E000
- Offset 0x61C
- Type RW, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	R3	RW	0	SSI Module 3 Run Mode Clock Gating Control Value Description 0 SSI module 3 is disabled. 1 Enable and provide a clock to SSI module 3 in Run mode.
2	R2	RW	0	SSI Module 2 Run Mode Clock Gating Control Value Description 0 SSI module 2 is disabled. 1 Enable and provide a clock to SSI module 2 in Run mode.
1	R1	RW	0	SSI Module 1 Run Mode Clock Gating Control Value Description 0 SSI module 1 is disabled. 1 Enable and provide a clock to SSI module 1 in Run mode.
0	R0	RW	0	SSI Module 0 Run Mode Clock Gating Control Value Description 0 SSI module 0 is disabled. 1 Enable and provide a clock to SSI module 0 in Run mode.

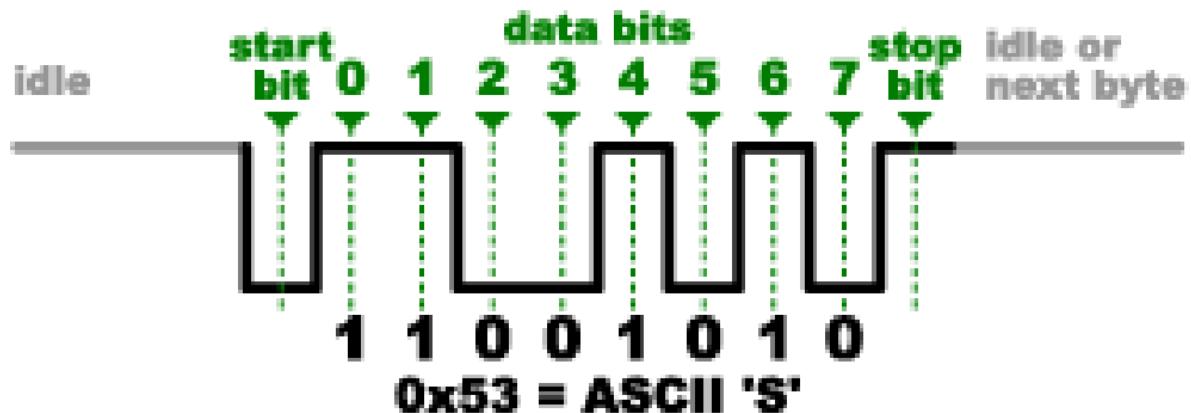
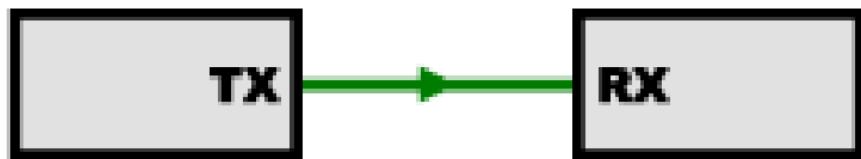
22.6.7 SSI Data (SSIDR)

- SSI0 base: 0x4000.8000
- SSI1 base: 0x4000.9000
- SSI2 base: 0x4000.A000
- SSI3 base: 0x4000.B000
- Offset 0x008
- Type RW, reset 0x0000.0000

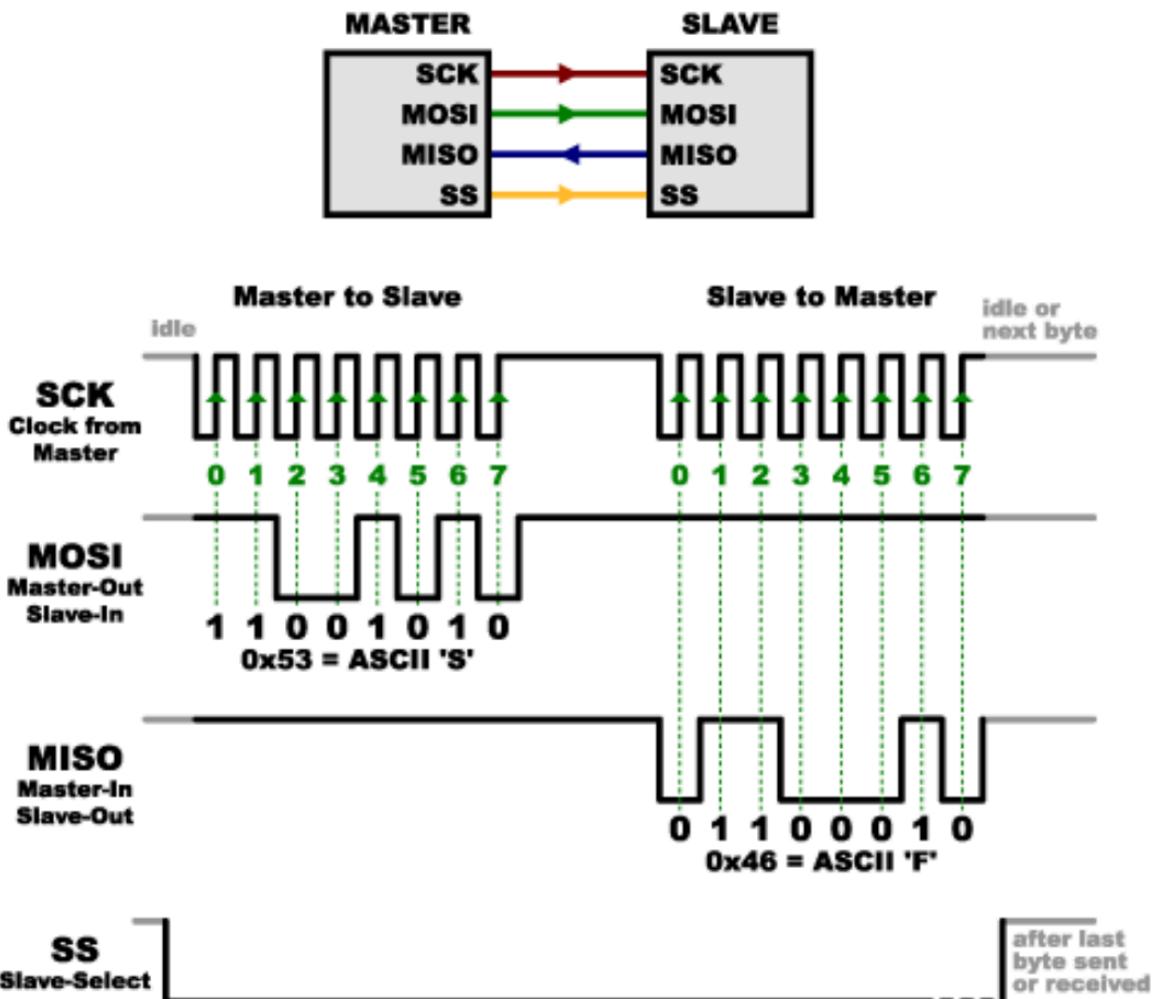


Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	DATA	RW	0x0000	<p>SSI Receive/Transmit Data</p> <p>A read operation reads the receive FIFO. A write operation writes the transmit FIFO.</p> <p>Software must right-justify data when the SSI is programmed for a data size that is less than 16 bits.</p> <p>Unused bits at the top are ignored by the transmit logic.</p> <p>The receive logic automatically right-justifies the data.</p>

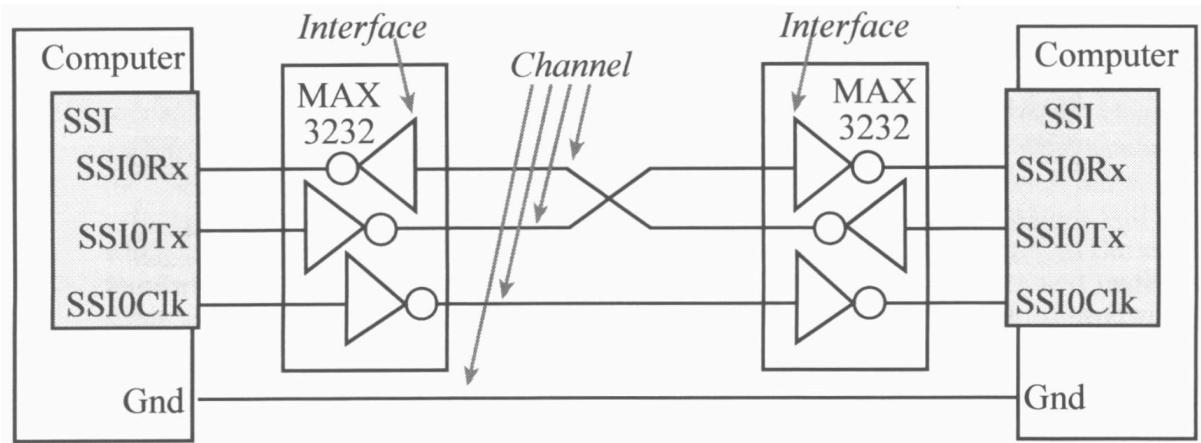
22.7 Connection to I/O device



- Serial protocols will often send the least significant bits first, so the smallest bit is on the far left. The lower nibble is actually 0011 = 0x3, and the upper nibble is 0101 = 0x5.



- The side that generates the clock is called the “master”, and the other side is called the “slave”.



- TI MAX3232 chip line driver is used to improve bandwidth, increase range and reduce noise, as well as to bring the voltages to transistor-transistor logic (TTL) requirements.

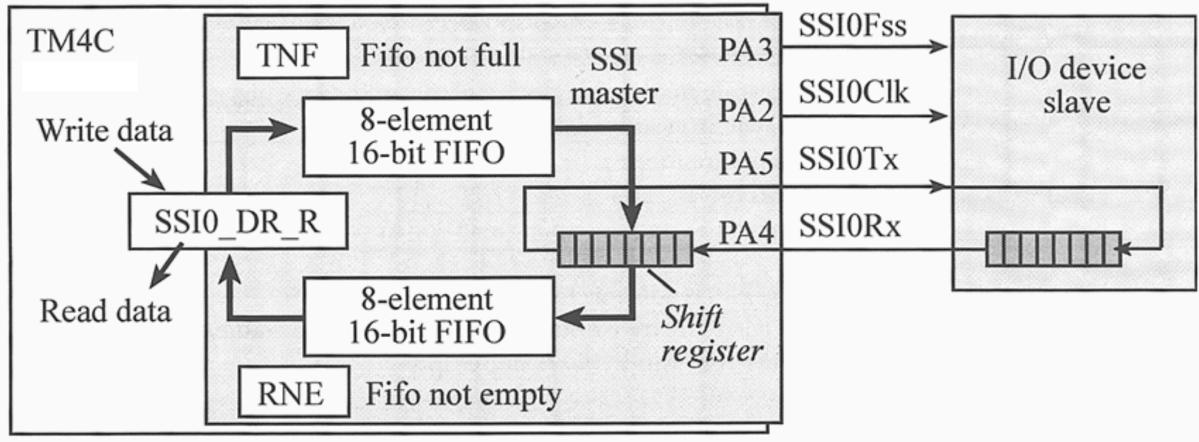


Figure 6: TM4C123G SSI connection between the microcontroller and an I/O device.

- Shift register can be configured from 4 to 16 bits (SSICR0) ([page 288](#))
- Shift register in the master and shift register in the slave are linked to form a distributed register.
- Data is written to the SSIDR ([page 297](#)) register is transmit, and is read from the SSIDR ([page 297](#)) register to receive.
- SSI on the TM4C123G has 2 16-bit FIFOs with 8 element depth.
- Data is transmitted by writing to the SSIDR ([page 297](#)) register, which puts the data in the transmission FIFO.
- Data is received by reading the SSIDR ([page 297](#)), which gets data from the receiving FIFO.
- SSI transmits and receives data at the same time.
- Data is shifted from master to slave, and from slave to master.
- Depending on the configured data of the FIFO (either 4 to 16 bits wide), data is serially shifted by 4 bits to 16 bits positions by the SCK clock, so data is exchanged between master and slave.

22.8 SSI clock frequency configuration

	31-16	15-8	7	6	5-4	3-0	
	\$4000.8000	SCR	SPH	SPO	FRF	DSS	SSI0_CRO_R

- SSI serial clock rate (frequency) set by SCR (8 bit) in the SSICR0 ([page 288](#)) register.
- SCR is any 8 bit value from 0 to 255.

\$4000.8004	7	6	5	4	3	2	1	0	SSI0_CRI_R
\$4000.800C				BSY	SOD	MS	SSE	LBM	SSI0_SR_R
\$4000.8010					RFF	RNE	TNF	TFE	SSI0_CPSR_R
				CPSDVSR					

- Clock prescale divisor CPDVSR (8 bit) is in the SSICPSR ([page 293](#)) register.
- CPDVSR is any **even** number from 2 to 254.

$$F_{\text{bus}} = \text{Frequency of bus clock}$$

$$F_{\text{SSI}} = \frac{F_{\text{bus}}}{\text{CPDVSR} \times (1 + \text{SCR})}$$

$$\therefore \text{SSInClk} = \frac{\text{SysClk}}{\text{CPDVSR} \times (1 + \text{SCR})}$$

22.9 SSI mode control bits

- There are 3 mode control bits, MS, SP0, and SPH to configure the transmission protocol.

\$4000.8004	7	6	5	4	3	2	1	0	SSI0_CRI_R
-------------	---	---	---	---	---	---	---	---	------------

- If MS = 0, the device is set to master mode, so the device generates the SCLK line and outputs data to the SSI0Tx line and receives input on the SSI0Rx line.

\$4000.8000	31-16	15-8	7	6	5-4	3-0	DSS	SSI0_CR0_R
-------------	-------	------	---	---	-----	-----	-----	------------

- The SP0 controls polarity of the SCLK.
- The SPH controls the phase timing of the first bit to be transferred and received.
- When SPH = 0, the device will shift data on the 1st, 3rd, 5th, ... clock cycle, i.e. odd clock cycles.
- When SPH = 1, the device will shift data on the 2nd, 4th, 6th, ... clock cycle, i.e. even clock cycles.

22.10 Initialisation

1. Enable the SSI module using the **RCCGSSI** (page 296) register.
2. Enable the clock to the appropriate GPIO module via the **RCGCGPIO** (page 133) register.
3. Set the **GPIOAFSEL** (page 148) bits for the appropriate pins.
4. Configure the PMCn fields in the **GPIOPCTL** (page 158) register to assign the SSI signals to the appropriate pins.
5. Program the **GPIODEN** (page 135) register to enable the pin's digital function. In addition, the drive strength, drain select and pull-up/pull-down functions must be configured.

Note: Pull-ups can be used to avoid unnecessary toggles on the SSI pins, which can take the slave into a wrong state. In addition, if the SSIClk signal is programmed to steady state High through the SP0 bit in the **SSICR0** (page 288) register, then software must also configure the GPIO port pin corresponding to the SSInClk signal as a pull-up in the **GPIO Pull-Up Select (GPIOPUR)** (page 145) register.

22.11 Configuration

For each of the frame formats, the SSI is configured using the following steps:

1. Ensure that the SSE bit in the SSICR1 ([page 290](#)) register is clear before making any configuration changes.
2. Select whether the SSI is a master or slave:
 - a. For master operations, set the SSICR1 ([page 290](#)) register to 0x0000.0000.
 - b. For slave mode (output enabled), set the SSICR1 ([page 290](#)) register to 0x0000.0004.
3. Configure the SSI clock source by writing to the SSICC ([page 292](#)) register.
4. Configure the clock prescale divisor by writing to the SSICPSR ([page 293](#)) register.
5. Write the SSICR0 ([page 288](#)) register with the following configuration:
 - Serial clock rate (SCR).
 - Desired clock phase and polarity, if using Freescale SPI mode (SPH and SP0).
 - The protocol mode: Freescale SPI, TI SSF, MICROWAVE (FRF).
 - The data size (DSS).
6. Enable the SSI by setting the SSE bit in the SSICR1 ([page 290](#)) register.

22.11.1 Example

The SSI must be configured to operate with the following parameters:

- Master operation
- Freescale SPI mode (SP0 = 1, SPH = 1)
- 1 Mbps bit rate
- 8 data bits

Assuming the system clock is 20 MHz, the bit rate calculation would be:

$$\text{SSIInClk} = \frac{\text{SysClk}}{\text{CPSDVSR} \times (1 + \text{SCR})}$$

$$1 \times 10^6 = \frac{20 \times 10^6}{\text{CPSDVSR} \times (1 + \text{SCR})}$$

In this case, if CPSDVSR = 0x2, SCR must be 0x9.

The configuration sequence would be as follows:

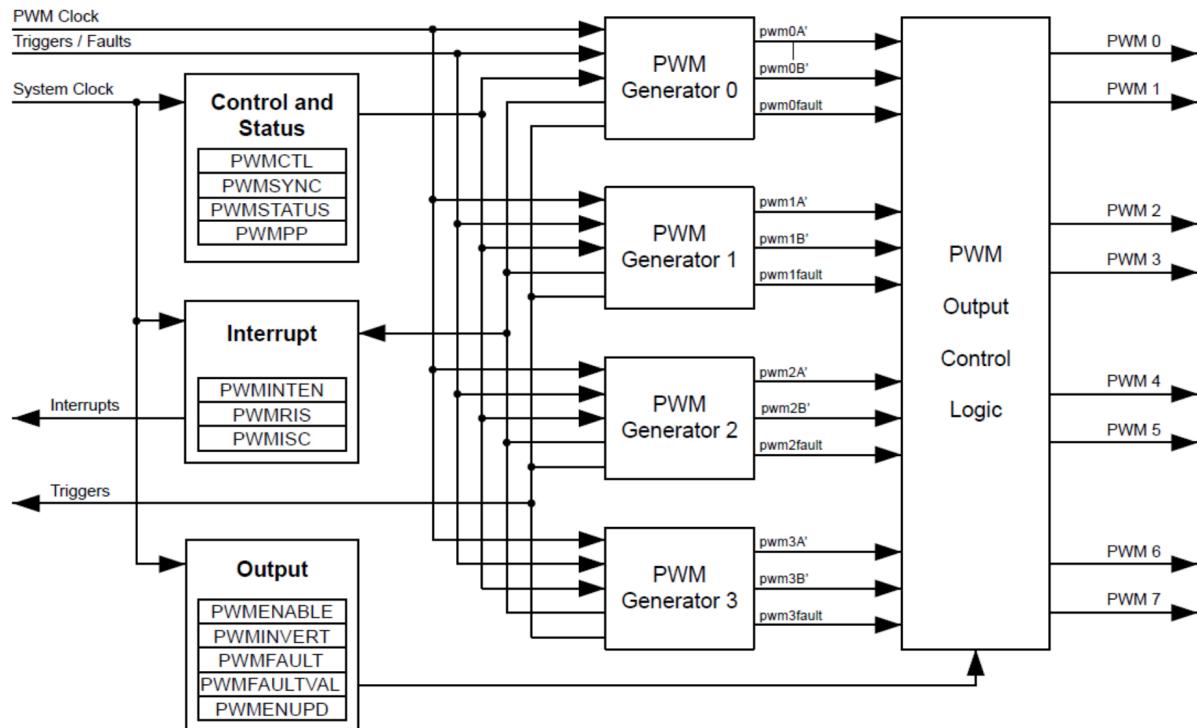
1. Ensure that the SSE bit in the SSICR1 ([page 290](#)) register is clear.
2. Write to the SSICR1 (SSI control) ([page 290](#)) register with a value of 0x0000.0000, which disables SSI.
3. Write to the SSICPSR (SSI clock prescale) ([page 293](#)) register with a value of 0x0000.0002.
4. Write the SSICR0 (SSI control 0) ([page 288](#)) register with a value of 0x0000.09C7.
5. The SSI is then enabled by setting the SSE bit in the SSICR1 ([page 290](#)) register.

23 TM4C123GH6PM PWM

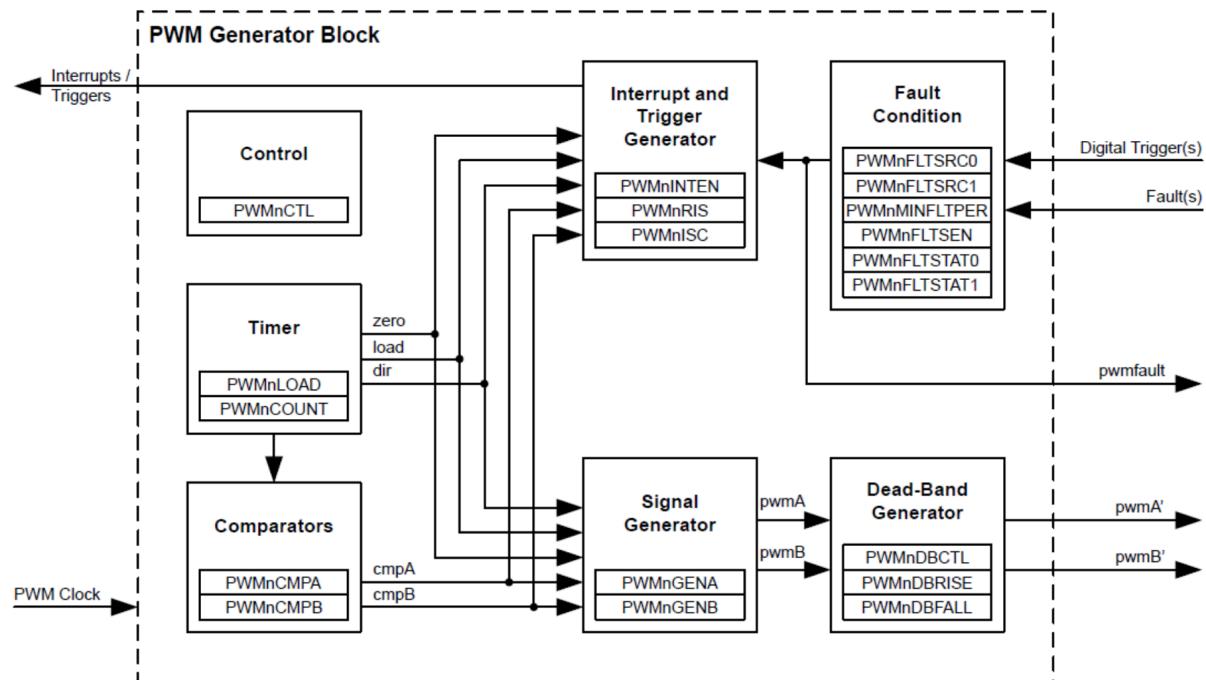
- 2 PWM modules.
- Each PWM module has 4 generator blocks and a control block.
- Each generator block can produce 2 PWM signals, so the total is $2 \times 4 \times 2 = 16$ PWM outputs.
- Each generator block has:
 - One fault-condition handling inputs to quickly provide low-latency shutdown and prevent damage to the motor being controlled, for a total of two inputs.
 - One 16-bit counter.
 - The ability to run in down or up/down mode.
 - Has output frequency controlled by a 16-bit load value.
 - Load value updates that can be synchronised.
 - The ability to produce output signals at zero and load value.
 - Two PWM comparators.
 - Optional output inversion of each PWM signal, or polarity control.
 - Optional fault handling for each PWM signal.
 - Synchronisation of timers in the PWM generator blocks.
 - Synchronisation of timer or comparator updates across the PWM generator blocks.
 - Extended PWM synchronisation of timer or comparator updates across the PWM generator blocks.
 - Interrupt status summary of the PWM generator blocks.
 - Extended PWM fault handling, with multiple fault signals, programmable polarities, and filtering.
 - PWM generators can be operated independently or synchronised with other generators.

23.1 Block diagrams

23.1.1 PWM module



23.1.2 PWM generator block



23.2 PWM pins

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type	Description
M0FAULT0	30	PF2 (4)	I	TTL	Motion Control Module 0 PWM Fault 0.
	53	PB6 (4)			
	63	PD2 (4)			
M0PWM0	1	PB6 (4)	O	TTL	Motion Control Module 0 PWM 0. This signal is controlled by Module 0 PWM Generator 0.
M0PWM1	4	PB7 (4)	O	TTL	Motion Control Module 0 PWM 1. This signal is controlled by Module 0 PWM Generator 0.
M0PWM2	58	PB4 (4)	O	TTL	Motion Control Module 0 PWM 2. This signal is controlled by Module 0 PWM Generator 1.
M0PWM3	57	PB5 (4)	O	TTL	Motion Control Module 0 PWM 3. This signal is controlled by Module 0 PWM Generator 1.
M0PWM4	59	PE4 (4)	O	TTL	Motion Control Module 0 PWM 4. This signal is controlled by Module 0 PWM Generator 2.
M0PWM5	60	PE5 (4)	O	TTL	Motion Control Module 0 PWM 5. This signal is controlled by Module 0 PWM Generator 2.
M0PWM6	16	PC4 (4)	O	TTL	Motion Control Module 0 PWM 6. This signal is controlled by Module 0 PWM Generator 3.
	61	PD0 (4)			
M0PWM7	15	PC5 (4)	O	TTL	Motion Control Module 0 PWM 7. This signal is controlled by Module 0 PWM Generator 3.
	62	PD1 (4)			
M1FAULT0	5	PF4 (5)	I	TTL	Motion Control Module 1 PWM Fault 0.
M1PWM0	61	PD0 (5)	O	TTL	Motion Control Module 1 PWM 0. This signal is controlled by Module 1 PWM Generator 0.

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type	Description
M1PWM1	62	PD1 (5)	O	TTL	Motion Control Module 1 PWM 1. This signal is controlled by Module 1 PWM Generator 0.
M1PWM2	23	PA6 (5)	O	TTL	Motion Control Module 1 PWM 2. This signal is controlled by Module 1 PWM Generator 1.
	59	PE4 (5)			
M1PWM3	24	PA7 (5)	O	TTL	Motion Control Module 1 PWM 3. This signal is controlled by Module 1 PWM Generator 1.
	60	PE5 (5)			
M1PWM4	28	PF0 (5)	O	TTL	Motion Control Module 1 PWM 4. This signal is controlled by Module 1 PWM Generator 2.
M1PWM5	29	PF1 (5)	O	TTL	Motion Control Module 1 PWM 5. This signal is controlled by Module 1 PWM Generator 2.
M1PWM6	30	PF2 (5)	O	TTL	Motion Control Module 1 PWM 6. This signal is controlled by Module 1 PWM Generator 3.
M1PWM7	31	PF3 (5)	O	TTL	Motion Control Module 1 PWM 7. This signal is controlled by Module 1 PWM Generator 3.

23.3 PWM register map

Offset	Name	Type	Reset	Description
0x000	PWMCTL	RW	0x0000.0000	PWM Master Control
0x004	PWMSYNC	RW	0x0000.0000	PWM Time Base Sync
0x008	PWMENABLE	RW	0x0000.0000	PWM Output Enable
0x00C	PWMINVERT	RW	0x0000.0000	PWM Output Inversion
0x010	PWMFAULT	RW	0x0000.0000	PWM Output Fault
0x014	PWMINTEN	RW	0x0000.0000	PWM Interrupt Enable
0x018	PWMRIS	RO	0x0000.0000	PWM Raw Interrupt Status
0x01C	PWMISC	RW1C	0x0000.0000	PWM Interrupt Status and Clear
0x020	PWMSTATUS	RO	0x0000.0000	PWM Status
0x024	PWMFAULTVAL	RW	0x0000.0000	PWM Fault Condition Value
0x028	PWMENUPD	RW	0x0000.0000	PWM Enable Update
0x040	PWM0CTL	RW	0x0000.0000	PWM0 Control
0x044	PWM0INTEN	RW	0x0000.0000	PWM0 Interrupt and Trigger Enable
0x048	PWM0RIS	RO	0x0000.0000	PWM0 Raw Interrupt Status
0x04C	PWM0ISC	RW1C	0x0000.0000	PWM0 Interrupt Status and Clear
0x050	PWM0LOAD	RW	0x0000.0000	PWM0 Load
0x054	PWM0COUNT	RO	0x0000.0000	PWM0 Counter
0x058	PWM0CMPA	RW	0x0000.0000	PWM0 Compare A
0x05C	PWM0CMPB	RW	0x0000.0000	PWM0 Compare B
0x060	PWM0GENA	RW	0x0000.0000	PWM0 Generator A Control
0x064	PWM0GENB	RW	0x0000.0000	PWM0 Generator B Control
0x068	PWM0DBCTL	RW	0x0000.0000	PWM0 Dead-Band Control
0x06C	PWM0DBRISE	RW	0x0000.0000	PWM0 Dead-Band Rising-Edge Delay
0x070	PWM0DBFALL	RW	0x0000.0000	PWM0 Dead-Band Falling-Edge-Delay
0x074	PWM0FLTSRC0	RW	0x0000.0000	PWM0 Fault Source 0
0x078	PWM0FLTSRC1	RW	0x0000.0000	PWM0 Fault Source 1
0x07C	PWM0MINFLTPER	RW	0x0000.0000	PWM0 Minimum Fault Period
0x080	PWM1CTL	RW	0x0000.0000	PWM1 Control
0x084	PWM1INTEN	RW	0x0000.0000	PWM1 Interrupt and Trigger Enable
0x088	PWM1RIS	RO	0x0000.0000	PWM1 Raw Interrupt Status
0x08C	PWM1ISC	RW1C	0x0000.0000	PWM1 Interrupt Status and Clear
0x090	PWM1LOAD	RW	0x0000.0000	PWM1 Load
0x094	PWM1COUNT	RO	0x0000.0000	PWM1 Counter
0x098	PWM1CMPA	RW	0x0000.0000	PWM1 Compare A
0x09C	PWM1CMPB	RW	0x0000.0000	PWM1 Compare B
0x0A0	PWM1GENA	RW	0x0000.0000	PWM1 Generator A Control
0x0A4	PWM1GENB	RW	0x0000.0000	PWM1 Generator B Control
0x0A8	PWM1DBCTL	RW	0x0000.0000	PWM1 Dead-Band Control

Offset	Name	Type	Reset	Description
0x0AC	PWM1DBRISE	RW	0x0000.0000	PWM1 Dead-Band Rising-Edge Delay
0x0B0	PWM1DBFALL	RW	0x0000.0000	PWM1 Dead-Band Falling-Edge-Delay
0x0B4	PWM1FLTSRC0	RW	0x0000.0000	PWM1 Fault Source 0
0x0B8	PWM1FLTSRC1	RW	0x0000.0000	PWM1 Fault Source 1
0x0BC	PWM1MINFLTPER	RW	0x0000.0000	PWM1 Minimum Fault Period
0x0C0	PWM2CTL	RW	0x0000.0000	PWM2 Control
0x0C4	PWM2INTEN	RW	0x0000.0000	PWM2 Interrupt and Trigger Enable
0x0C8	PWM2RIS	RO	0x0000.0000	PWM2 Raw Interrupt Status
0x0CC	PWM2ISC	RW1C	0x0000.0000	PWM2 Interrupt Status and Clear
0x0D0	PWM2LOAD	RW	0x0000.0000	PWM2 Load
0x0D4	PWM2COUNT	RO	0x0000.0000	PWM2 Counter
0x0D8	PWM2CMPA	RW	0x0000.0000	PWM2 Compare A
0x0DC	PWM2CMPB	RW	0x0000.0000	PWM2 Compare B
0x0E0	PWM2GENA	RW	0x0000.0000	PWM2 Generator A Control
0x0E4	PWM2GENB	RW	0x0000.0000	PWM2 Generator B Control
0x0E8	PWM2DBCTL	RW	0x0000.0000	PWM2 Dead-Band Control
0x0EC	PWM2DBRISE	RW	0x0000.0000	PWM2 Dead-Band Rising-Edge Delay
0x0F0	PWM2DBFALL	RW	0x0000.0000	PWM2 Dead-Band Falling-Edge-Delay
0x0F4	PWM2FLTSRC0	RW	0x0000.0000	PWM2 Fault Source 0
0x0F8	PWM2FLTSRC1	RW	0x0000.0000	PWM2 Fault Source 1
0x0FC	PWM2MINFLTPER	RW	0x0000.0000	PWM2 Minimum Fault Period
0x100	PWM3CTL	RW	0x0000.0000	PWM3 Control
0x104	PWM3INTEN	RW	0x0000.0000	PWM3 Interrupt and Trigger Enable
0x108	PWM3RIS	RO	0x0000.0000	PWM3 Raw Interrupt Status
0x10C	PWM3ISC	RW1C	0x0000.0000	PWM3 Interrupt Status and Clear
0x110	PWM3LOAD	RW	0x0000.0000	PWM3 Load
0x114	PWM3COUNT	RO	0x0000.0000	PWM3 Counter
0x118	PWM3CMPA	RW	0x0000.0000	PWM3 Compare A
0x11C	PWM3CMPB	RW	0x0000.0000	PWM3 Compare B
0x120	PWM3GENA	RW	0x0000.0000	PWM3 Generator A Control
0x124	PWM3GENB	RW	0x0000.0000	PWM3 Generator B Control
0x128	PWM3DBCTL	RW	0x0000.0000	PWM3 Dead-Band Control
0x12C	PWM3DBRISE	RW	0x0000.0000	PWM3 Dead-Band Rising-Edge Delay
0x130	PWM3DBFALL	RW	0x0000.0000	PWM3 Dead-Band Falling-Edge-Delay
0x134	PWM3FLTSRC0	RW	0x0000.0000	PWM3 Fault Source 0
0x138	PWM3FLTSRC1	RW	0x0000.0000	PWM3 Fault Source 1
0x13C	PWM3MINFLTPER	RW	0x0000.0000	PWM3 Minimum Fault Period
0x800	PWM0FLTSEN	RW	0x0000.0000	PWM0 Fault Pin Logic Sense
0x804	PWM0FLTSTAT0	-	0x0000.0000	PWM0 Fault Status 0

Offset	Name	Type	Reset	Description
0x808	PWM0FLTSTAT1	-	0x0000.0000	PWM0 Fault Status 1
0x880	PWM1FLTSEN	RW	0x0000.0000	PWM1 Fault Pin Logic Sense
0x884	PWM1FLTSTAT0	-	0x0000.0000	PWM1 Fault Status 0
0x888	PWM1FLTSTAT1	-	0x0000.0000	PWM1 Fault Status 1
0x904	PWM2FLTSTAT0	-	0x0000.0000	PWM2 Fault Status 0
0x908	PWM2FLTSTAT1	-	0x0000.0000	PWM2 Fault Status 1
0x984	PWM3FLTSTAT0	-	0x0000.0000	PWM3 Fault Status 0
0x988	PWM3FLTSTAT1	-	0x0000.0000	PWM3 Fault Status 1
0xFC0	PWMPP	RO	0x0000.0314	PWM Peripheral Properties

23.4 Registers

23.4.1 Run-Mode Clock Configuration (RCC)

- Base **0x400F.E000**
- Offset **0x060**
- Type RW, reset **0x078E.3AD1**

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved				ACG	SYSDIV				USESYSDIV	reserved	USEPWMDIV	PWMDIV			
Type	RO	RO	RO	RO	RW	RW	RW	RW	RW	RO	RO	RW	RW	RW	RW	RO
Reset	0	0	0	0	0	1	1	1	1	0	0	0	1	1	1	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved				PWRDN	reserved	BYPASS	XTAL				OSCSRC	reserved			
Type	RO	RO	RW	RO	RW	RW	RW	RW	RW	RW	RW	RW	RO	RO	RO	RW
Reset	0	0	1	1	1	0	1	0	1	1	0	1	0	0	0	1

Bit/Field	Name	Type	Reset	Description						
31:28	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
27	ACG	RW	0	<p>Auto Clock Gating</p> <p>This bit specifies whether the system uses the Sleep-Mode Clock Gating Control (SCGCn) registers and Deep-Sleep-Mode Clock Gating Control (DCGCn) registers if the microcontroller enters a Sleep or Deep-Sleep mode (respectively).</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The Run-Mode Clock Gating Control (RCGCn) (page 317) registers are used when the microcontroller enters a sleep mode.</td> </tr> <tr> <td>1</td> <td>The SCGCn or DCGCn registers are used to control the clocks distributed to the peripherals when the microcontroller is in a sleep mode. The SCGCn and DCGCn registers allow unused peripherals to consume less power when the microcontroller is in a sleep mode.</td> </tr> </tbody> </table> <p>The RCGCn (page 317) registers are always used to control the clocks in Run mode.</p>	Value	Description	0	The Run-Mode Clock Gating Control (RCGCn) (page 317) registers are used when the microcontroller enters a sleep mode.	1	The SCGCn or DCGCn registers are used to control the clocks distributed to the peripherals when the microcontroller is in a sleep mode. The SCGCn and DCGCn registers allow unused peripherals to consume less power when the microcontroller is in a sleep mode.
Value	Description									
0	The Run-Mode Clock Gating Control (RCGCn) (page 317) registers are used when the microcontroller enters a sleep mode.									
1	The SCGCn or DCGCn registers are used to control the clocks distributed to the peripherals when the microcontroller is in a sleep mode. The SCGCn and DCGCn registers allow unused peripherals to consume less power when the microcontroller is in a sleep mode.									

Bit/Field	Name	Type	Reset	Description						
26:23	SYSDIV	RW	0xF	<p>System Clock Divisor</p> <p>Specifies which divisor is used to generate the system clock from either the PLL output or the oscillator source (depending on how the BYPASS bit in this register is configured). See page 316 for bit encodings.</p> <p>If the SYSDIV value is less than MINSYSDIV, and the PLL is being used, then the MINSYSDIV value is used as the divisor.</p> <p>If the PLL is not being used, the SYSDIV value can be less than MINSYSDIV.</p>						
22	USESYS DIV	RW	0	<p>Enable System Clock Divider</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The system clock is used undivided.</td> </tr> <tr> <td>1</td> <td>The system clock divider is the source for the system clock. The system clock divider is forced to be used when the PLL is selected as the source.</td> </tr> </tbody> </table> <p>If the USERCC2 bit in the RCC2 register is set, then the SYSDIV2 field in the RCC2 register is used as the system clock divider rather than the SYSDIV field in this register.</p>	Value	Description	0	The system clock is used undivided.	1	The system clock divider is the source for the system clock. The system clock divider is forced to be used when the PLL is selected as the source.
Value	Description									
0	The system clock is used undivided.									
1	The system clock divider is the source for the system clock. The system clock divider is forced to be used when the PLL is selected as the source.									
21	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
20	USEPWMDIV	RW	0	<p>Enable PWM Clock Divider</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The system clock is the source for the PWM clock.</td> </tr> <tr> <td>1</td> <td>The PWM clock divider is the source for the PWM clock.</td> </tr> </tbody> </table> <p>Note that when the PWM divisor is used, it is applied to the clock for both PWM modules.</p>	Value	Description	0	The system clock is the source for the PWM clock.	1	The PWM clock divider is the source for the PWM clock.
Value	Description									
0	The system clock is the source for the PWM clock.									
1	The PWM clock divider is the source for the PWM clock.									

Bit/Field	Name	Type	Reset	Description																		
19:17	PWMDIV	RW	0x7	<p>PWM Unit Clock Divisor</p> <p>This field specifies the binary divisor used to predivide the system clock down for use as the timing reference for the PWM module. The rising edge of this clock is synchronous with the system clock.</p> <table> <thead> <tr> <th>Value</th> <th>Divisor</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>/2</td> </tr> <tr> <td>0x1</td> <td>/4</td> </tr> <tr> <td>0x2</td> <td>/8</td> </tr> <tr> <td>0x3</td> <td>/16</td> </tr> <tr> <td>0x4</td> <td>/32</td> </tr> <tr> <td>0x5</td> <td>/64</td> </tr> <tr> <td>0x6</td> <td>/64</td> </tr> <tr> <td>0x7</td> <td>/64 (default)</td> </tr> </tbody> </table>	Value	Divisor	0x0	/2	0x1	/4	0x2	/8	0x3	/16	0x4	/32	0x5	/64	0x6	/64	0x7	/64 (default)
Value	Divisor																					
0x0	/2																					
0x1	/4																					
0x2	/8																					
0x3	/16																					
0x4	/32																					
0x5	/64																					
0x6	/64																					
0x7	/64 (default)																					
16:14	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.																		
13	PWRDN	RW	1	<p>PLL Power Down</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The PLL is operating normally.</td> </tr> <tr> <td>1</td> <td>The PLL is powered down. Care must be taken to ensure that another clock source is functioning and that the BYPASS bit is set before setting this bit.</td> </tr> </tbody> </table>	Value	Description	0	The PLL is operating normally.	1	The PLL is powered down. Care must be taken to ensure that another clock source is functioning and that the BYPASS bit is set before setting this bit.												
Value	Description																					
0	The PLL is operating normally.																					
1	The PLL is powered down. Care must be taken to ensure that another clock source is functioning and that the BYPASS bit is set before setting this bit.																					
12	reserved	RO	1	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.																		

Bit/Field	Name	Type	Reset	Description						
11	BYPASS	RW	1	<p>PLL Bypass</p> <table> <thead> <tr> <th data-bbox="738 314 817 348">Value</th> <th data-bbox="817 314 976 348">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="738 359 817 471">0</td> <td data-bbox="817 359 1341 471">The system clock is the PLL output clock divided by the divisor specified by SYSDIV (page 316).</td> </tr> <tr> <td data-bbox="738 482 817 595">1</td> <td data-bbox="817 482 1349 595">The system clock is derived from the OSC source and divided by the divisor specified by SYSDIV.</td> </tr> </tbody> </table> <p>Note: The ADC must be clocked from the PLL or directly from a 16-MHz clock source to operate properly.</p>	Value	Description	0	The system clock is the PLL output clock divided by the divisor specified by SYSDIV (page 316).	1	The system clock is derived from the OSC source and divided by the divisor specified by SYSDIV.
Value	Description									
0	The system clock is the PLL output clock divided by the divisor specified by SYSDIV (page 316).									
1	The system clock is derived from the OSC source and divided by the divisor specified by SYSDIV.									

Bit/Field	Name	Type	Reset	Description																																																																					
10:6	XTAL	RW	0x0B	<p>Crystal Value</p> <p>This field specifies the crystal value attached to the main oscillator. The encoding for this field is provided below.</p> <p>Frequencies that may be used with the USB interface are indicated in the table. To function within the clocking requirements of the USB specification, a crystal of 5, 6, 8, 10, 12, or 16 MHz must be used.</p> <table> <thead> <tr> <th>Value</th> <th>Crystal Frequency (MHz) Not Using the PLL</th> <th>Crystal Frequency (MHz) Using the PLL</th> </tr> </thead> <tbody> <tr> <td>0x00-0x05</td> <td>reserved</td> <td></td> </tr> <tr> <td>0x06</td> <td>4 MHz</td> <td>Reserved</td> </tr> <tr> <td>0x07</td> <td>4.096 MHz</td> <td>Reserved</td> </tr> <tr> <td>0x08</td> <td>4.9152 MHz</td> <td>Reserved</td> </tr> <tr> <td>0x09</td> <td></td> <td>5 MHz (USB)</td> </tr> <tr> <td>0x0A</td> <td></td> <td>5.12 MHz</td> </tr> <tr> <td>0x0B</td> <td></td> <td>6 MHz (USB)</td> </tr> <tr> <td>0x0C</td> <td></td> <td>6.144 MHz</td> </tr> <tr> <td>0x0D</td> <td></td> <td>7.3728 MHz</td> </tr> <tr> <td>0x0E</td> <td></td> <td>8 MHz (USB)</td> </tr> <tr> <td>0x0F</td> <td></td> <td>8.192 MHz</td> </tr> <tr> <td>0x10</td> <td></td> <td>10.0 MHz (USB)</td> </tr> <tr> <td>0x11</td> <td></td> <td>12.0 MHz (USB)</td> </tr> <tr> <td>0x12</td> <td></td> <td>12.288 MHz</td> </tr> <tr> <td>0x13</td> <td></td> <td>13.56 MHz</td> </tr> <tr> <td>0x14</td> <td></td> <td>14.31818 MHz</td> </tr> <tr> <td>0x15</td> <td></td> <td>16.0 MHz (USB)</td> </tr> <tr> <td>0x16</td> <td></td> <td>16.384 MHz</td> </tr> <tr> <td>0x17</td> <td></td> <td>18.0 MHz (USB)</td> </tr> <tr> <td>0x18</td> <td></td> <td>20.0 MHz (USB)</td> </tr> <tr> <td>0x19</td> <td></td> <td>24.0 MHz (USB)</td> </tr> <tr> <td>0x1A</td> <td></td> <td>25.0 MHz (USB)</td> </tr> </tbody> </table>	Value	Crystal Frequency (MHz) Not Using the PLL	Crystal Frequency (MHz) Using the PLL	0x00-0x05	reserved		0x06	4 MHz	Reserved	0x07	4.096 MHz	Reserved	0x08	4.9152 MHz	Reserved	0x09		5 MHz (USB)	0x0A		5.12 MHz	0x0B		6 MHz (USB)	0x0C		6.144 MHz	0x0D		7.3728 MHz	0x0E		8 MHz (USB)	0x0F		8.192 MHz	0x10		10.0 MHz (USB)	0x11		12.0 MHz (USB)	0x12		12.288 MHz	0x13		13.56 MHz	0x14		14.31818 MHz	0x15		16.0 MHz (USB)	0x16		16.384 MHz	0x17		18.0 MHz (USB)	0x18		20.0 MHz (USB)	0x19		24.0 MHz (USB)	0x1A		25.0 MHz (USB)
Value	Crystal Frequency (MHz) Not Using the PLL	Crystal Frequency (MHz) Using the PLL																																																																							
0x00-0x05	reserved																																																																								
0x06	4 MHz	Reserved																																																																							
0x07	4.096 MHz	Reserved																																																																							
0x08	4.9152 MHz	Reserved																																																																							
0x09		5 MHz (USB)																																																																							
0x0A		5.12 MHz																																																																							
0x0B		6 MHz (USB)																																																																							
0x0C		6.144 MHz																																																																							
0x0D		7.3728 MHz																																																																							
0x0E		8 MHz (USB)																																																																							
0x0F		8.192 MHz																																																																							
0x10		10.0 MHz (USB)																																																																							
0x11		12.0 MHz (USB)																																																																							
0x12		12.288 MHz																																																																							
0x13		13.56 MHz																																																																							
0x14		14.31818 MHz																																																																							
0x15		16.0 MHz (USB)																																																																							
0x16		16.384 MHz																																																																							
0x17		18.0 MHz (USB)																																																																							
0x18		20.0 MHz (USB)																																																																							
0x19		24.0 MHz (USB)																																																																							
0x1A		25.0 MHz (USB)																																																																							

Bit/Field	Name	Type	Reset	Description										
5:4	OSCSRC	RW	0x1	<p>Oscillator Source</p> <p>Selects the input source for the OSC. The values are:</p> <table> <thead> <tr> <th>Value</th> <th>Input Source</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>MOSC Main oscillator</td> </tr> <tr> <td>0x1</td> <td>PIOSC Precision internal oscillator (default)</td> </tr> <tr> <td>0x2</td> <td>PIOSC/4 Precision internal oscillator / 4</td> </tr> <tr> <td>0x3</td> <td>LFIOSC Low-frequency internal oscillator</td> </tr> </tbody> </table> <p>For additional oscillator sources, see the RCC2 register.</p>	Value	Input Source	0x0	MOSC Main oscillator	0x1	PIOSC Precision internal oscillator (default)	0x2	PIOSC/4 Precision internal oscillator / 4	0x3	LFIOSC Low-frequency internal oscillator
Value	Input Source													
0x0	MOSC Main oscillator													
0x1	PIOSC Precision internal oscillator (default)													
0x2	PIOSC/4 Precision internal oscillator / 4													
0x3	LFIOSC Low-frequency internal oscillator													
3:1	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.										
0	MOSCDIS	RW	1	<p>Main Oscillator Disable</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The main oscillator is enabled.</td> </tr> <tr> <td>1</td> <td>The main oscillator is disabled (default).</td> </tr> </tbody> </table>	Value	Description	0	The main oscillator is enabled.	1	The main oscillator is disabled (default).				
Value	Description													
0	The main oscillator is enabled.													
1	The main oscillator is disabled (default).													

23.4.1.1 Possible system clock frequencies using SYSDIV field

SYSDIV	Divisor	Frequency (BYPASS=0)	Frequency (BYPASS=1)	TivaWare Parameter
0x0	/1	Reserved	Clock source frequency/1	SYSCTL_SYSDIV_1
0x1	/2	Reserved	Clock source frequency/2	SYSCTL_SYSDIV_2
0x2	/3	66.67 MHz	Clock source frequency/3	SYSCTL_SYSDIV_3
0x3	/4	50 MHz	Clock source frequency/4	SYSCTL_SYSDIV_4
0x4	/5	40 MHz	Clock source frequency/5	SYSCTL_SYSDIV_5
0x5	/6	33.33 MHz	Clock source frequency/6	SYSCTL_SYSDIV_6
0x6	/7	28.57 MHz	Clock source frequency/7	SYSCTL_SYSDIV_7
0x7	/8	25 MHz	Clock source frequency/8	SYSCTL_SYSDIV_8
0x8	/9	22.22 MHz	Clock source frequency/9	SYSCTL_SYSDIV_9
0x9	/10	20 MHz	Clock source frequency/10	SYSCTL_SYSDIV_10
0xA	/11	18.18 MHz	Clock source frequency/11	SYSCTL_SYSDIV_11
0xB	/12	16.67 MHz	Clock source frequency/12	SYSCTL_SYSDIV_12
0xC	/13	15.38 MHz	Clock source frequency/13	SYSCTL_SYSDIV_13
0xD	/14	14.29 MHz	Clock source frequency/14	SYSCTL_SYSDIV_14
0xE	/15	13.33 MHz	Clock source frequency/15	SYSCTL_SYSDIV_15
0xF	/16	12.5 MHz (default)	Clock source frequency/16	SYSCTL_SYSDIV_16

23.4.2 Run Mode Clock Gating Control Register 0 (RCGC0)

- Base 0x400F.E000
- Offset 0x100
- Type RO, reset 0x0000.0040

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Type	reserved	WDT1	reserved	CAN1	CAN0	reserved	PWM0	reserved	ADC1	ADC0						
Reset	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	reserved	MAXADC1SPD	MAXADC0SPD	reserved	HIB	reserved	WDT0	reserved								
Reset	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 1	RO 0						

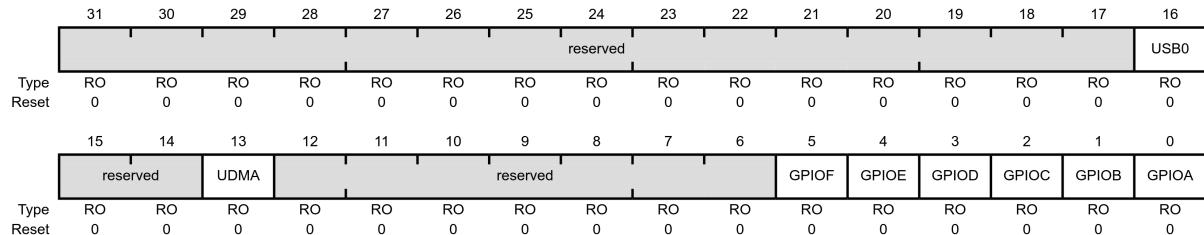
Bit/Field	Name	Type	Reset	Description
31:29	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
28	WDT1	RO	0x0	<p>WDT1 Clock Gating Control</p> <p>This bit controls the clock gating for the Watchdog Timer module 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>
27:26	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
25	CAN1	RO	0x0	<p>CAN1 Clock Gating Control</p> <p>This bit controls the clock gating for CAN module 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>
24	CAN0	RO	0x0	<p>CAN0 Clock Gating Control</p> <p>This bit controls the clock gating for CAN module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>
23:21	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description										
20	PWM0	RO	0x0	<p>PWM Clock Gating Control</p> <p>This bit controls the clock gating for the PWM module. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>										
19:18	reserved	RO	0	<p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p>										
17	ADC1	RO	0x0	<p>ADC1 Clock Gating Control</p> <p>This bit controls the clock gating for SAR ADC module 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>										
16	ADC0	RO	0x0	<p>ADC0 Clock Gating Control</p> <p>This bit controls the clock gating for ADC module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>										
15:12	reserved	RO	0	<p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p>										
11:10	MAXADC1SPD	RO	0x0	<p>ADC1 Sample Speed</p> <p>This field sets the rate at which ADC module 1 samples data. You cannot set the rate higher than the maximum rate. You can set the sample rate by setting the MAXADC1SPD bit as follows (all other encodings are reserved):</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>125K samples/second</td> </tr> <tr> <td>0x1</td> <td>250K samples/second</td> </tr> <tr> <td>0x2</td> <td>500K samples/second</td> </tr> <tr> <td>0x3</td> <td>1M samples/second</td> </tr> </tbody> </table>	Value	Description	0x0	125K samples/second	0x1	250K samples/second	0x2	500K samples/second	0x3	1M samples/second
Value	Description													
0x0	125K samples/second													
0x1	250K samples/second													
0x2	500K samples/second													
0x3	1M samples/second													

Bit/Field	Name	Type	Reset	Description										
9:8	MAXADC0SPD	RO	0x0	<p>ADC0 Sample Speed</p> <p>This field sets the rate at which ADC0 samples data. You cannot set the rate higher than the maximum rate. You can set the sample rate by setting the MAXADC0SPD bit as follows (all other encodings are reserved):</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>125K samples/second</td> </tr> <tr> <td>0x1</td> <td>250K samples/second</td> </tr> <tr> <td>0x2</td> <td>500K samples/second</td> </tr> <tr> <td>0x3</td> <td>1M samples/second</td> </tr> </tbody> </table>	Value	Description	0x0	125K samples/second	0x1	250K samples/second	0x2	500K samples/second	0x3	1M samples/second
Value	Description													
0x0	125K samples/second													
0x1	250K samples/second													
0x2	500K samples/second													
0x3	1M samples/second													
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.										
6	HIB	RO	0x1	<p>HIB Clock Gating Control</p> <p>This bit controls the clock gating for the Hibernation module. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>										
5:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.										
3	WDT0	RO	0x0	<p>WDT0 Clock Gating Control</p> <p>This bit controls the clock gating for the Watchdog Timer module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>										
2:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.										

23.4.3 Run Mode Clock Gating Control Register 2 (RCGC2)

- Base 0x400F.E000
- Offset 0x108
- Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:17	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	USB0	RO	0x0	USB0 Clock Gating Control This bit controls the clock gating for USB module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.
15:14	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13	UDMA	RO	0x0	Micro-DMA Clock Gating Control This bit controls the clock gating for micro-DMA. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.
12:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	GPIOF	RO	0x0	Port F Clock Gating Control This bit controls the clock gating for Port F. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.
4	GPIOE	RO	0x0	Port E Clock Gating Control This bit controls the clock gating for Port E. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.

Bit/Field	Name	Type	Reset	Description
3	GPIOD	RO	0x0	<p>Port D Clock Gating Control</p> <p>This bit controls the clock gating for Port D. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>
2	GPIOC	RO	0x0	<p>Port C Clock Gating Control</p> <p>This bit controls the clock gating for Port C. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>
1	GPIOB	RO	0x0	<p>Port B Clock Gating Control</p> <p>This bit controls the clock gating for Port B. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>
0	GPIOA	RO	0x0	<p>Port A Clock Gating Control</p> <p>This bit controls the clock gating for Port A. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>

23.4.4 PWMn Control (PWMnCTL)

These registers configure the PWM signal generation blocks ([PWM0CTL \(page 322\)](#) controls the PWM generator 0 block, and so on).

Documentation:

- PWM0 base: 0x4002.8000
- PWM1 base: 0x4002.9000
- Offset 0x040
- Type RW, reset 0x0000.0000

Registers:

- Register 12: PWM0 Control (PWM0CTL), offset 0x040
- Register 13: PWM1 Control (PWM1CTL), offset 0x080
- Register 14: PWM2 Control (PWM2CTL), offset 0x0C0
- Register 15: PWM3 Control (PWM3CTL), offset 0x100

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved												LATCH	MINFLTPER	FLTSRC	
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DBFALLUPD	DBRISEUPD	DBCTLUPD	GENBUPD	GENAUPD	CMPBUPD	CMPAUPD	LOADUPD	DEBUG	MODE	ENABLE					
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Bit/Field	Name	Type	Reset	Description							
31:19	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.							
18	LATCH	RW	0	<p>Latch Fault Input</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Fault Condition Not Latched A fault condition is in effect for as long as the generating source is asserting.</td> </tr> <tr> <td>1</td> <td>Fault Condition Latched A fault condition is set as the result of the assertion of the faulting source and is held (latched) while the PWMISETC (page 380) INTFAULTn bit is set. Clearing the INTFAULTn bit clears the fault condition.</td> </tr> </tbody> </table> <p>Note: When using an ADC digital comparator as a fault source, the LATCH and MINFLTPER bits in the PWMnCTL (page 322) register should be set to 1 to ensure trigger assertions are captured.</p>		Value	Description	0	Fault Condition Not Latched A fault condition is in effect for as long as the generating source is asserting.	1	Fault Condition Latched A fault condition is set as the result of the assertion of the faulting source and is held (latched) while the PWMISETC (page 380) INTFAULTn bit is set. Clearing the INTFAULTn bit clears the fault condition.
Value	Description										
0	Fault Condition Not Latched A fault condition is in effect for as long as the generating source is asserting.										
1	Fault Condition Latched A fault condition is set as the result of the assertion of the faulting source and is held (latched) while the PWMISETC (page 380) INTFAULTn bit is set. Clearing the INTFAULTn bit clears the fault condition.										

Bit/Field	Name	Type	Reset	Description						
17	MINFLTPER	RW	0	<p>Minimum Fault Period</p> <p>This bit specifies that the PWM generator enables a one-shot counter to provide a minimum fault condition period.</p> <p>The timer begins counting on the rising edge of the fault condition to extend the condition to the minimum duration of the count value. The timer ignores the state of the fault condition while counting.</p> <p>The minimum fault delay is in effect only when the MINFLTPER bit is set. If a detected fault is in the process of being extended when the MINFLTPER bit is cleared, the fault condition extension is aborted.</p> <p>The delay time is specified by the PWMnMINFLTPER register MFF field value. The effect of this is to pulse stretch the fault condition input.</p> <p>The delay value is defined by the PWM clock period. Because the fault input is not synchronized to the PWM clock, the period of the time is PWMnCLOCK (MFFP value + 1) or PWMnCLOCK (MFFP value + 2).</p> <p>The delay function makes sense only if the fault source is unlatched. A latched fault source makes the fault condition appear asserted until cleared by software and negates the utility of the extend feature. It applies to all fault condition sources as specified in the FLTSRC field.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The FAULT input deassertion is unaffected.</td></tr> <tr> <td>1</td><td>The PWMnMINFLTPER (page 349) one-shot counter is active and extends the period of the fault condition to a minimum period.</td></tr> </tbody> </table> <p>Note: When using an ADC digital comparator as a fault source, the LATCH and MINFLTPER bits in the PWMnCTL (page 322) register should be set to 1 to ensure trigger assertions are captured.</p>	Value	Description	0	The FAULT input deassertion is unaffected.	1	The PWMnMINFLTPER (page 349) one-shot counter is active and extends the period of the fault condition to a minimum period.
Value	Description									
0	The FAULT input deassertion is unaffected.									
1	The PWMnMINFLTPER (page 349) one-shot counter is active and extends the period of the fault condition to a minimum period.									
16	FLTSRC	RW	0	<p>Fault Condition Source</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The fault condition is determined by the Faultn input.</td></tr> <tr> <td>1</td><td>The fault condition is determined by the configuration of the PWMnFLTSRC0 and PWMnFLTSRC1 registers.</td></tr> </tbody> </table>	Value	Description	0	The fault condition is determined by the Faultn input.	1	The fault condition is determined by the configuration of the PWMnFLTSRC0 and PWMnFLTSRC1 registers.
Value	Description									
0	The fault condition is determined by the Faultn input.									
1	The fault condition is determined by the configuration of the PWMnFLTSRC0 and PWMnFLTSRC1 registers.									

Bit/Field	Name	Type	Reset	Description																
15:14	DBFALLUPD	RW	0x0	<p>PWMnDBFALL (page 338) Update Mode</p> <table> <thead> <tr> <th data-bbox="716 316 795 350">Value</th> <th data-bbox="795 316 954 350">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="716 361 795 395">0x0</td> <td data-bbox="795 361 954 395">Immediate</td> </tr> <tr> <td data-bbox="716 406 795 473"></td> <td data-bbox="795 406 954 473">PWMnDBFALL (page 338) register value is immediately updated on a write.</td> </tr> <tr> <td data-bbox="716 496 795 530">0x1</td> <td data-bbox="795 496 954 530">Reserved</td> </tr> <tr> <td data-bbox="716 541 795 574">0x2</td> <td data-bbox="795 541 954 574">Locally Synchronized</td> </tr> <tr> <td data-bbox="716 586 795 653"></td> <td data-bbox="795 586 954 653">Updates to the register are reflected to the generator the next time the counter is 0.</td> </tr> <tr> <td data-bbox="716 676 795 709">0x3</td> <td data-bbox="795 676 954 709">Globally Synchronized</td> </tr> <tr> <td data-bbox="716 720 795 878"></td> <td data-bbox="795 720 954 878">Updates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the PWMCTL (page 347) register.</td> </tr> </tbody> </table>	Value	Description	0x0	Immediate		PWMnDBFALL (page 338) register value is immediately updated on a write.	0x1	Reserved	0x2	Locally Synchronized		Updates to the register are reflected to the generator the next time the counter is 0.	0x3	Globally Synchronized		Updates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the PWMCTL (page 347) register.
Value	Description																			
0x0	Immediate																			
	PWMnDBFALL (page 338) register value is immediately updated on a write.																			
0x1	Reserved																			
0x2	Locally Synchronized																			
	Updates to the register are reflected to the generator the next time the counter is 0.																			
0x3	Globally Synchronized																			
	Updates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the PWMCTL (page 347) register.																			
13:12	DBRISEUPD	RW	0x0	<p>PWMnDBRISE (page 337) Update Mode</p> <table> <thead> <tr> <th data-bbox="716 983 795 1017">Value</th> <th data-bbox="795 983 954 1017">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="716 1028 795 1062">0x0</td> <td data-bbox="795 1028 954 1062">Immediate</td> </tr> <tr> <td data-bbox="716 1073 795 1140"></td> <td data-bbox="795 1073 954 1140">The PWMnDBRISE (page 337) register value is immediately updated on a write.</td> </tr> <tr> <td data-bbox="716 1163 795 1197">0x1</td> <td data-bbox="795 1163 954 1197">Reserved</td> </tr> <tr> <td data-bbox="716 1208 795 1242">0x2</td> <td data-bbox="795 1208 954 1242">Locally Synchronized</td> </tr> <tr> <td data-bbox="716 1253 795 1320"></td> <td data-bbox="795 1253 954 1320">Updates to the register are reflected to the generator the next time the counter is 0.</td> </tr> <tr> <td data-bbox="716 1343 795 1376">0x3</td> <td data-bbox="795 1343 954 1376">Globally Synchronized</td> </tr> <tr> <td data-bbox="716 1388 795 1545"></td> <td data-bbox="795 1388 954 1545">Updates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the PWMCTL (page 347) register.</td> </tr> </tbody> </table>	Value	Description	0x0	Immediate		The PWMnDBRISE (page 337) register value is immediately updated on a write.	0x1	Reserved	0x2	Locally Synchronized		Updates to the register are reflected to the generator the next time the counter is 0.	0x3	Globally Synchronized		Updates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the PWMCTL (page 347) register.
Value	Description																			
0x0	Immediate																			
	The PWMnDBRISE (page 337) register value is immediately updated on a write.																			
0x1	Reserved																			
0x2	Locally Synchronized																			
	Updates to the register are reflected to the generator the next time the counter is 0.																			
0x3	Globally Synchronized																			
	Updates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the PWMCTL (page 347) register.																			

Bit/Field	Name	Type	Reset	Description										
11:10	DBCTLUPD	RW	0x0	<p>PWMnDBCTL (page 345) Update Mode</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Immediate</td> </tr> <tr> <td>0x1</td> <td>Reserved</td> </tr> <tr> <td>0x2</td> <td>Locally Synchronized</td> </tr> <tr> <td>0x3</td> <td>Globally Synchronized</td> </tr> </tbody> </table> <p>The PWMnDBCTL (page 345) register value is immediately updated on a write.</p> <p>Updates to the register are reflected to the generator the next time the counter is 0.</p> <p>Updates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the PWMCTL (page 347) register.</p>	Value	Description	0x0	Immediate	0x1	Reserved	0x2	Locally Synchronized	0x3	Globally Synchronized
Value	Description													
0x0	Immediate													
0x1	Reserved													
0x2	Locally Synchronized													
0x3	Globally Synchronized													
9:8	GENBUPD	RW	0x0	<p>PWMnGENB (page 330) Update Mode</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Immediate</td> </tr> <tr> <td>0x1</td> <td>Reserved</td> </tr> <tr> <td>0x2</td> <td>Locally Synchronized</td> </tr> <tr> <td>0x3</td> <td>Globally Synchronized</td> </tr> </tbody> </table> <p>The PWMnGENB (page 330) register value is immediately updated on a write.</p> <p>Updates to the register are reflected to the generator the next time the counter is 0.</p> <p>Updates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the PWMCTL (page 347) register.</p>	Value	Description	0x0	Immediate	0x1	Reserved	0x2	Locally Synchronized	0x3	Globally Synchronized
Value	Description													
0x0	Immediate													
0x1	Reserved													
0x2	Locally Synchronized													
0x3	Globally Synchronized													

Bit/Field	Name	Type	Reset	Description										
7:6	GENAUPD	RW	0x0	<p>PWMnGENA (page 328) Update Mode</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Immediate</td> </tr> <tr> <td>0x1</td> <td>Reserved</td> </tr> <tr> <td>0x2</td> <td>Locally Synchronized</td> </tr> <tr> <td>0x3</td> <td>Updates to the register are reflected to the generator the next time the counter is 0.</td> </tr> </tbody> </table> <p>The PWMnGENA (page 328) register value is immediately updated on a write.</p> <p>Updates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the PWMCTL (page 347) register.</p>	Value	Description	0x0	Immediate	0x1	Reserved	0x2	Locally Synchronized	0x3	Updates to the register are reflected to the generator the next time the counter is 0.
Value	Description													
0x0	Immediate													
0x1	Reserved													
0x2	Locally Synchronized													
0x3	Updates to the register are reflected to the generator the next time the counter is 0.													
5	CMPBUPD	RW	0	<p>Comparator B Update Mode</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Locally Synchronized</td> </tr> <tr> <td>1</td> <td>Updates to the register are reflected to the generator the next time the counter is 0.</td> </tr> </tbody> </table> <p>Updates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the PWMCTL (page 347) register.</p>	Value	Description	0	Locally Synchronized	1	Updates to the register are reflected to the generator the next time the counter is 0.				
Value	Description													
0	Locally Synchronized													
1	Updates to the register are reflected to the generator the next time the counter is 0.													
4	CMPAUPD	RW	0	<p>Comparator A Update Mode</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Locally Synchronized</td> </tr> <tr> <td>1</td> <td>Updates to the register are reflected to the generator the next time the counter is 0.</td> </tr> </tbody> </table> <p>Updates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the PWMCTL (page 347) register.</p>	Value	Description	0	Locally Synchronized	1	Updates to the register are reflected to the generator the next time the counter is 0.				
Value	Description													
0	Locally Synchronized													
1	Updates to the register are reflected to the generator the next time the counter is 0.													

Bit/Field	Name	Type	Reset	Description						
3	LOADUPD	RW	0	<p>Load Register Update Mode</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Locally Synchronized Updates to the PWMnLOAD (page 332) register are reflected to the generator the next time the counter is 0.</td> </tr> <tr> <td>1</td> <td>Globally Synchronized Updates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the PWMCTL (page 347) register.</td> </tr> </tbody> </table>	Value	Description	0	Locally Synchronized Updates to the PWMnLOAD (page 332) register are reflected to the generator the next time the counter is 0.	1	Globally Synchronized Updates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the PWMCTL (page 347) register.
Value	Description									
0	Locally Synchronized Updates to the PWMnLOAD (page 332) register are reflected to the generator the next time the counter is 0.									
1	Globally Synchronized Updates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the PWMCTL (page 347) register.									
2	DEBUG	RW	0	<p>Debug Mode</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The counter stops running when it next reaches 0 and continues running again when no longer in Debug mode.</td> </tr> <tr> <td>1</td> <td>The counter always runs when in Debug mode.</td> </tr> </tbody> </table>	Value	Description	0	The counter stops running when it next reaches 0 and continues running again when no longer in Debug mode.	1	The counter always runs when in Debug mode.
Value	Description									
0	The counter stops running when it next reaches 0 and continues running again when no longer in Debug mode.									
1	The counter always runs when in Debug mode.									
1	MODE	RW	0	<p>Counter Mode</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The counter counts down from the load value to 0 and then wraps back to the load value (Count-Down mode).</td> </tr> <tr> <td>1</td> <td>The counter counts up from 0 to the load value, back down to 0, and then repeats (Count-Up/Down mode).</td> </tr> </tbody> </table>	Value	Description	0	The counter counts down from the load value to 0 and then wraps back to the load value (Count-Down mode).	1	The counter counts up from 0 to the load value, back down to 0, and then repeats (Count-Up/Down mode).
Value	Description									
0	The counter counts down from the load value to 0 and then wraps back to the load value (Count-Down mode).									
1	The counter counts up from 0 to the load value, back down to 0, and then repeats (Count-Up/Down mode).									
0	ENABLE	RW	0	<p>PWM Block Enable</p> <p>Note: Disabling the PWM by clearing the ENABLE bit does not clear the COUNT field of the PWMnCOUNT (page 376) register. Before re-enabling the PWM (ENABLE = 0x1), the COUNT field should be cleared by resetting the PWM registers through the SRPWM register in the System Control Module.</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The entire PWM generation block is disabled and not clocked.</td> </tr> <tr> <td>1</td> <td>The PWM generation block is enabled and produces PWM signals.</td> </tr> </tbody> </table>	Value	Description	0	The entire PWM generation block is disabled and not clocked.	1	The PWM generation block is enabled and produces PWM signals.
Value	Description									
0	The entire PWM generation block is disabled and not clocked.									
1	The PWM generation block is enabled and produces PWM signals.									

23.4.5 PWMn Generator A Control (PWMnGENA)

- PWM0 base: 0x4002.8000
- PWM1 base: 0x4002.9000
- Offset 0x060
- Type RW, reset 0x0000.0000

Registers:

- Register 44: PWM0 Generator A Control (PWM0GENA), offset 0x060
- Register 45: PWM1 Generator A Control (PWM1GENA), offset 0x0A0
- Register 46: PWM2 Generator A Control (PWM2GENA), offset 0x0E0
- Register 47: PWM3 Generator A Control (PWM3GENA), offset 0x120

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved																
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
reserved																
Type	RO	RO	RO	RO	RW											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description										
31:12	reserved	RO	0x000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.										
11:10	ACTCMPBD	RW	0x0	<p>Action for Comparator B Down</p> <p>This field specifies the action to be taken when the counter matches comparator B while counting down.</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Do nothing.</td> </tr> <tr> <td>0x1</td> <td>Invert pwmA.</td> </tr> <tr> <td>0x2</td> <td>Drive pwmA Low.</td> </tr> <tr> <td>0x3</td> <td>Drive pwmA High.</td> </tr> </tbody> </table>	Value	Description	0x0	Do nothing.	0x1	Invert pwmA.	0x2	Drive pwmA Low.	0x3	Drive pwmA High.
Value	Description													
0x0	Do nothing.													
0x1	Invert pwmA.													
0x2	Drive pwmA Low.													
0x3	Drive pwmA High.													
9:8	ACTMPBU	RW	0x0	<p>Action for Comparator B Up</p> <p>This field specifies the action to be taken when the counter matches comparator B while counting up. This action can only occur when the MODE bit in the PWMnCTL (page 322) register is set.</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Do nothing.</td> </tr> <tr> <td>0x1</td> <td>Invert pwmA.</td> </tr> <tr> <td>0x2</td> <td>Drive pwmA Low.</td> </tr> <tr> <td>0x3</td> <td>Drive pwmA High.</td> </tr> </tbody> </table>	Value	Description	0x0	Do nothing.	0x1	Invert pwmA.	0x2	Drive pwmA Low.	0x3	Drive pwmA High.
Value	Description													
0x0	Do nothing.													
0x1	Invert pwmA.													
0x2	Drive pwmA Low.													
0x3	Drive pwmA High.													

Bit/Field	Name	Type	Reset	Description										
7:6	ACTCMPAD	RW	0x0	<p>Action for Comparator A Down</p> <p>This field specifies the action to be taken when the counter matches comparator A while counting down.</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Do nothing.</td> </tr> <tr> <td>0x1</td> <td>Invert pwmA.</td> </tr> <tr> <td>0x2</td> <td>Drive pwmA Low.</td> </tr> <tr> <td>0x3</td> <td>Drive pwmA High.</td> </tr> </tbody> </table>	Value	Description	0x0	Do nothing.	0x1	Invert pwmA.	0x2	Drive pwmA Low.	0x3	Drive pwmA High.
Value	Description													
0x0	Do nothing.													
0x1	Invert pwmA.													
0x2	Drive pwmA Low.													
0x3	Drive pwmA High.													
5:4	ACTCMPAU	RW	0x0	<p>Action for Comparator A Up</p> <p>This field specifies the action to be taken when the counter matches comparator A while counting up. This action can only occur when the MODE bit in the PWMnCTL (page 322) register is set.</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Do nothing.</td> </tr> <tr> <td>0x1</td> <td>Invert pwmA.</td> </tr> <tr> <td>0x2</td> <td>Drive pwmA Low.</td> </tr> <tr> <td>0x3</td> <td>Drive pwmA High.</td> </tr> </tbody> </table>	Value	Description	0x0	Do nothing.	0x1	Invert pwmA.	0x2	Drive pwmA Low.	0x3	Drive pwmA High.
Value	Description													
0x0	Do nothing.													
0x1	Invert pwmA.													
0x2	Drive pwmA Low.													
0x3	Drive pwmA High.													
3:2	ACTLOAD	RW	0x0	<p>Action for Counter=LOAD</p> <p>This field specifies the action to be taken when the counter matches the value in the PWMnLOAD (page 332) register.</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Do nothing.</td> </tr> <tr> <td>0x1</td> <td>Invert pwmA.</td> </tr> <tr> <td>0x2</td> <td>Drive pwmA Low.</td> </tr> <tr> <td>0x3</td> <td>Drive pwmA High.</td> </tr> </tbody> </table>	Value	Description	0x0	Do nothing.	0x1	Invert pwmA.	0x2	Drive pwmA Low.	0x3	Drive pwmA High.
Value	Description													
0x0	Do nothing.													
0x1	Invert pwmA.													
0x2	Drive pwmA Low.													
0x3	Drive pwmA High.													
1:0	ACTZERO	RW	0x0	<p>Action for Counter=0</p> <p>This field specifies the action to be taken when the counter is zero.</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Do nothing.</td> </tr> <tr> <td>0x1</td> <td>Invert pwmA.</td> </tr> <tr> <td>0x2</td> <td>Drive pwmA Low.</td> </tr> <tr> <td>0x3</td> <td>Drive pwmA High.</td> </tr> </tbody> </table>	Value	Description	0x0	Do nothing.	0x1	Invert pwmA.	0x2	Drive pwmA Low.	0x3	Drive pwmA High.
Value	Description													
0x0	Do nothing.													
0x1	Invert pwmA.													
0x2	Drive pwmA Low.													
0x3	Drive pwmA High.													

23.4.6 PWMn Generator B Control (PWMnGENB)

- PWM0 base: 0x4002.8000
- PWM1 base: 0x4002.9000
- Offset 0x064
- Type RW, reset 0x0000.0000

Registers:

- Register 48: PWM0 Generator B Control (PWM0GENB), offset 0x064
- Register 49: PWM1 Generator B Control (PWM1GENB), offset 0x0A4
- Register 50: PWM2 Generator B Control (PWM2GENB), offset 0x0E4
- Register 51: PWM3 Generator B Control (PWM3GENB), offset 0x124

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved																
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
reserved																
Type	RO	RO	RO	RO	RW											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description										
31:12	reserved	RO	0x000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.										
11:10	ACTCMPBD	RW	0x0	<p>Action for Comparator B Down</p> <p>This field specifies the action to be taken when the counter matches comparator B while counting down.</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Do nothing.</td> </tr> <tr> <td>0x1</td> <td>Invert pwmB.</td> </tr> <tr> <td>0x2</td> <td>Drive pwmB Low.</td> </tr> <tr> <td>0x3</td> <td>Drive pwmB High.</td> </tr> </tbody> </table>	Value	Description	0x0	Do nothing.	0x1	Invert pwmB.	0x2	Drive pwmB Low.	0x3	Drive pwmB High.
Value	Description													
0x0	Do nothing.													
0x1	Invert pwmB.													
0x2	Drive pwmB Low.													
0x3	Drive pwmB High.													
9:8	ACTMPBU	RW	0x0	<p>Action for Comparator B Up</p> <p>This field specifies the action to be taken when the counter matches comparator B while counting up. This action can only occur when the MODE bit in the PWMnCTL (page 322) register is set.</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Do nothing.</td> </tr> <tr> <td>0x1</td> <td>Invert pwmB.</td> </tr> <tr> <td>0x2</td> <td>Drive pwmB Low.</td> </tr> <tr> <td>0x3</td> <td>Drive pwmB High.</td> </tr> </tbody> </table>	Value	Description	0x0	Do nothing.	0x1	Invert pwmB.	0x2	Drive pwmB Low.	0x3	Drive pwmB High.
Value	Description													
0x0	Do nothing.													
0x1	Invert pwmB.													
0x2	Drive pwmB Low.													
0x3	Drive pwmB High.													

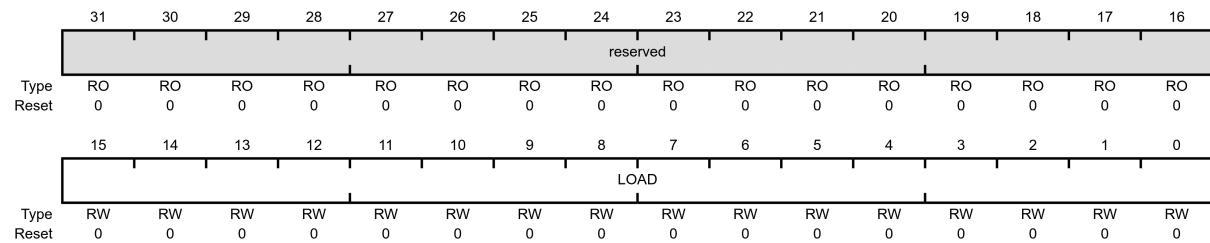
Bit/Field	Name	Type	Reset	Description										
7:6	ACTCMPAD	RW	0x0	<p>Action for Comparator A Down</p> <p>This field specifies the action to be taken when the counter matches comparator A while counting down.</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Do nothing.</td> </tr> <tr> <td>0x1</td> <td>Invert pwmB.</td> </tr> <tr> <td>0x2</td> <td>Drive pwmB Low.</td> </tr> <tr> <td>0x3</td> <td>Drive pwmB High.</td> </tr> </tbody> </table>	Value	Description	0x0	Do nothing.	0x1	Invert pwmB.	0x2	Drive pwmB Low.	0x3	Drive pwmB High.
Value	Description													
0x0	Do nothing.													
0x1	Invert pwmB.													
0x2	Drive pwmB Low.													
0x3	Drive pwmB High.													
5:4	ACTCMPAU	RW	0x0	<p>Action for Comparator A Up</p> <p>This field specifies the action to be taken when the counter matches comparator A while counting up. This action can only occur when the MODE bit in the PWMnCTL (page 322) register is set.</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Do nothing.</td> </tr> <tr> <td>0x1</td> <td>Invert pwmB.</td> </tr> <tr> <td>0x2</td> <td>Drive pwmB Low.</td> </tr> <tr> <td>0x3</td> <td>Drive pwmB High.</td> </tr> </tbody> </table>	Value	Description	0x0	Do nothing.	0x1	Invert pwmB.	0x2	Drive pwmB Low.	0x3	Drive pwmB High.
Value	Description													
0x0	Do nothing.													
0x1	Invert pwmB.													
0x2	Drive pwmB Low.													
0x3	Drive pwmB High.													
3:2	ACTLOAD	RW	0x0	<p>Action for Counter=LOAD</p> <p>This field specifies the action to be taken when the counter matches the load value.</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Do nothing.</td> </tr> <tr> <td>0x1</td> <td>Invert pwmB.</td> </tr> <tr> <td>0x2</td> <td>Drive pwmB Low.</td> </tr> <tr> <td>0x3</td> <td>Drive pwmB High.</td> </tr> </tbody> </table>	Value	Description	0x0	Do nothing.	0x1	Invert pwmB.	0x2	Drive pwmB Low.	0x3	Drive pwmB High.
Value	Description													
0x0	Do nothing.													
0x1	Invert pwmB.													
0x2	Drive pwmB Low.													
0x3	Drive pwmB High.													
1:0	ACTZERO	RW	0x0	<p>Action for Counter=0</p> <p>This field specifies the action to be taken when the counter is 0.</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Do nothing.</td> </tr> <tr> <td>0x1</td> <td>Invert pwmB.</td> </tr> <tr> <td>0x2</td> <td>Drive pwmB Low.</td> </tr> <tr> <td>0x3</td> <td>Drive pwmB High.</td> </tr> </tbody> </table>	Value	Description	0x0	Do nothing.	0x1	Invert pwmB.	0x2	Drive pwmB Low.	0x3	Drive pwmB High.
Value	Description													
0x0	Do nothing.													
0x1	Invert pwmB.													
0x2	Drive pwmB Low.													
0x3	Drive pwmB High.													

23.4.7 PWMn Load (PWMrLOAD)

- PWM0 base: 0x4002.8000
- PWM1 base: 0x4002.9000
- Offset 0x050
- Type RW, reset 0x0000.0000

Registers:

- Register 28: PWM0 Load (PWMrLOAD), offset 0x050
- Register 29: PWM1 Load (PWMrLOAD), offset 0x090
- Register 30: PWM2 Load (PWMrLOAD), offset 0x0D0
- Register 31: PWM3 Load (PWMrLOAD), offset 0x110



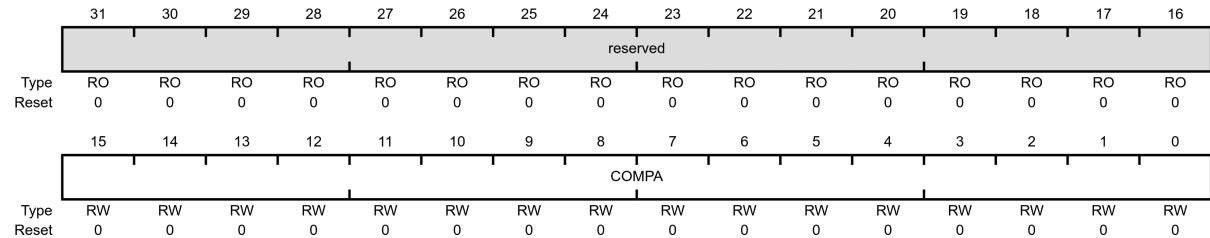
Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	LOAD	RW	0x0000	Counter Load Value The counter load value.

23.4.8 PWMn Compare A (PWMnCMPA)

- PWM0 base: 0x4002.8000
- PWM1 base: 0x4002.9000
- Offset 0x058
- Type RW, reset 0x0000.0000

Registers:

- Register 36: PWM0 Compare A (PWM0CMPA), offset 0x058
- Register 37: PWM1 Compare A (PWM1CMPA), offset 0x098
- Register 38: PWM2 Compare A (PWM2CMPA), offset 0x0D8
- Register 39: PWM3 Compare A (PWM3CMPA), offset 0x118

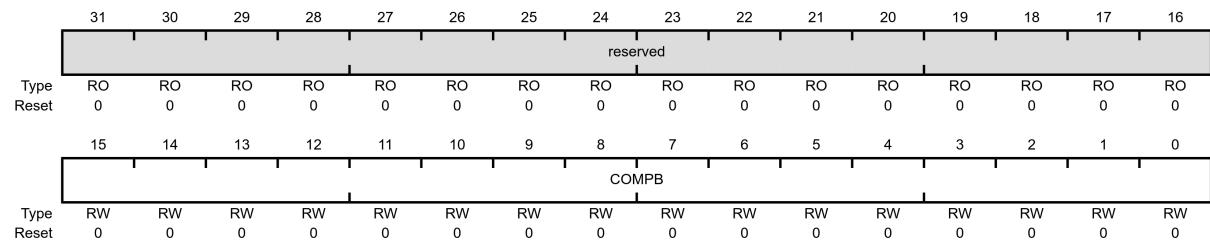


23.4.9 PWMn Compare B (PWMnCMPB)

- PWM0 base: 0x4002.8000
- PWM1 base: 0x4002.9000
- Offset 0x05C
- Type RW, reset 0x0000.0000

Registers:

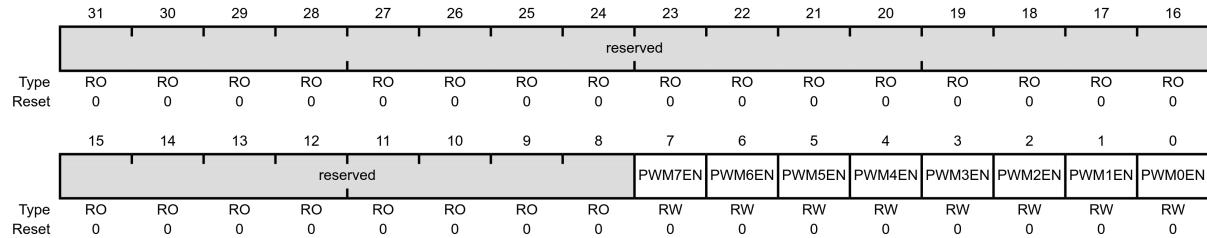
- Register 40: PWM0 Compare B (PWM0CMPB), offset 0x05C
- Register 41: PWM1 Compare B (PWM1CMPB), offset 0x09C
- Register 42: PWM2 Compare B (PWM2CMPB), offset 0x0DC
- Register 43: PWM3 Compare B (PWM3CMPB), offset 0x11C



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	COMPB	RW	0x0000	Comparator B Value The value to be compared against the counter.

23.4.10 PWM Output Enable (PWMDENABLE)

- PWM0 base: 0x4002.8000
- PWM1 base: 0x4002.9000
- Offset 0x008
- Type RW, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description						
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
7	PWM7EN	RW	0	<p>MnPWM7 Output Enable</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The MnPWM7 signal has a zero value.</td> </tr> <tr> <td>1</td> <td>The generated pwm3B' signal is passed to the MnPWM7 pin.</td> </tr> </tbody> </table>	Value	Description	0	The MnPWM7 signal has a zero value.	1	The generated pwm3B' signal is passed to the MnPWM7 pin.
Value	Description									
0	The MnPWM7 signal has a zero value.									
1	The generated pwm3B' signal is passed to the MnPWM7 pin.									
6	PWM6EN	RW	0	<p>MnPWM6 Output Enable</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The MnPWM6 signal has a zero value.</td> </tr> <tr> <td>1</td> <td>The generated pwm3A' signal is passed to the MnPWM6 pin.</td> </tr> </tbody> </table>	Value	Description	0	The MnPWM6 signal has a zero value.	1	The generated pwm3A' signal is passed to the MnPWM6 pin.
Value	Description									
0	The MnPWM6 signal has a zero value.									
1	The generated pwm3A' signal is passed to the MnPWM6 pin.									
5	PWM5EN	RW	0	<p>MnPWM5 Output Enable</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The MnPWM5 signal has a zero value.</td> </tr> <tr> <td>1</td> <td>The generated pwm2B' signal is passed to the MnPWM5 pin.</td> </tr> </tbody> </table>	Value	Description	0	The MnPWM5 signal has a zero value.	1	The generated pwm2B' signal is passed to the MnPWM5 pin.
Value	Description									
0	The MnPWM5 signal has a zero value.									
1	The generated pwm2B' signal is passed to the MnPWM5 pin.									
4	PWM4EN	RW	0	<p>MnPWM4 Output Enable</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The MnPWM4 signal has a zero value.</td> </tr> <tr> <td>1</td> <td>The generated pwm2A' signal is passed to the MnPWM4 pin.</td> </tr> </tbody> </table>	Value	Description	0	The MnPWM4 signal has a zero value.	1	The generated pwm2A' signal is passed to the MnPWM4 pin.
Value	Description									
0	The MnPWM4 signal has a zero value.									
1	The generated pwm2A' signal is passed to the MnPWM4 pin.									

Bit/Field	Name	Type	Reset	Description						
3	PWM3EN	RW	0	<p>MnPWM3 Output Enable</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The MnPWM3 signal has a zero value.</td> </tr> <tr> <td>1</td> <td>The generated pwm1B' signal is passed to the MnPWM3 pin.</td> </tr> </tbody> </table>	Value	Description	0	The MnPWM3 signal has a zero value.	1	The generated pwm1B' signal is passed to the MnPWM3 pin.
Value	Description									
0	The MnPWM3 signal has a zero value.									
1	The generated pwm1B' signal is passed to the MnPWM3 pin.									
2	PWM2EN	RW	0	<p>MnPWM2 Output Enable</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The MnPWM2 signal has a zero value.</td> </tr> <tr> <td>1</td> <td>The generated pwm1A' signal is passed to the MnPWM2 pin.</td> </tr> </tbody> </table>	Value	Description	0	The MnPWM2 signal has a zero value.	1	The generated pwm1A' signal is passed to the MnPWM2 pin.
Value	Description									
0	The MnPWM2 signal has a zero value.									
1	The generated pwm1A' signal is passed to the MnPWM2 pin.									
1	PWM1EN	RW	0	<p>MnPWM1 Output Enable</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The MnPWM1 signal has a zero value.</td> </tr> <tr> <td>1</td> <td>The generated pwm0B' signal is passed to the MnPWM1 pin.</td> </tr> </tbody> </table>	Value	Description	0	The MnPWM1 signal has a zero value.	1	The generated pwm0B' signal is passed to the MnPWM1 pin.
Value	Description									
0	The MnPWM1 signal has a zero value.									
1	The generated pwm0B' signal is passed to the MnPWM1 pin.									
0	PWM0EN	RW	0	<p>MnPWM0 Output Enable</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The MnPWM0 signal has a zero value.</td> </tr> <tr> <td>1</td> <td>The generated pwm0A' signal is passed to the MnPWM0 pin.</td> </tr> </tbody> </table>	Value	Description	0	The MnPWM0 signal has a zero value.	1	The generated pwm0A' signal is passed to the MnPWM0 pin.
Value	Description									
0	The MnPWM0 signal has a zero value.									
1	The generated pwm0A' signal is passed to the MnPWM0 pin.									

23.4.11 PWMn Dead-Band Rising-Edge Delay (PWMnDBRISE)

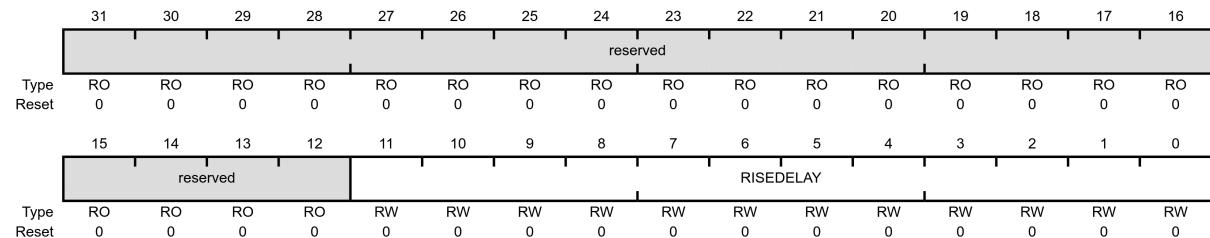
This register sets the dead-band rising delay.

Documentation:

- PWM0 base: 0x4002.8000
- PWM1 base: 0x4002.9000
- Offset 0x06C
- Type RW, reset 0x0000.0000

Registers:

- Register 56: PWM0 Dead-Band Rising-Edge Delay (PWM0DBRISE), offset 0x06C
- Register 57: PWM1 Dead-Band Rising-Edge Delay (PWM1DBRISE), offset 0x0AC
- Register 58: PWM2 Dead-Band Rising-Edge Delay (PWM2DBRISE), offset 0x0EC
- Register 59: PWM3 Dead-Band Rising-Edge Delay (PWM3DBRISE), offset 0x12C



Bit/Field	Name	Type	Reset	Description
31:12	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11:0	RISEDELAY	RW	0x000	Dead-Band Rise Delay The number of clock cycles to delay the rising edge of pwmA' after the rising edge of pwmA.

23.4.12 PWMn Dead-Band Falling-Edge-Delay (PWMnDBFALL)

This register sets the dead-band falling delay.

Documentation:

- PWM0 base: 0x4002.8000
- PWM1 base: 0x4002.9000
- Offset 0x070
- Type RW, reset 0x0000.0000

Registers:

- Register 60: PWM0 Dead-Band Falling-Edge-Delay (PWM0DBFALL), offset 0x070
- Register 61: PWM1 Dead-Band Falling-Edge-Delay (PWM1DBFALL), offset 0x0B0
- Register 62: PWM2 Dead-Band Falling-Edge-Delay (PWM2DBFALL), offset 0x0F0
- Register 63: PWM3 Dead-Band Falling-Edge-Delay (PWM3DBFALL), offset 0x130

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved															
Type	RO														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	RO	RO	RO	RO	RW										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FALLDELAY															

Bit/Field	Name	Type	Reset	Description
31:12	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11:0	FALLDELAY	RW	0x000	Dead-Band Fall Delay The number of clock cycles to delay the falling edge of pwmB' from the rising edge of pwmA.

23.4.13 PWMn Fault Source 0 (PWMnFLTSRC0)

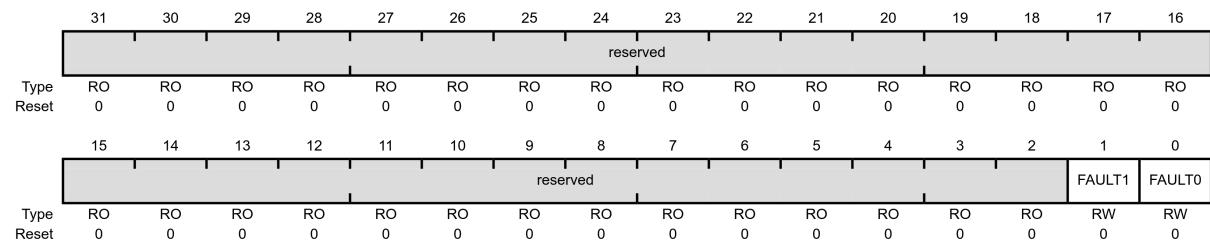
This register enables the PWM's fault source 0, which has 2 fault input pins. It specifies which fault pin inputs are used to generate a fault condition. Each bit in this register indicates whether the corresponding fault pin is included in the fault condition.

Documentation:

- PWM0 base: 0x4002.8000
- PWM1 base: 0x4002.9000
- Offset 0x074
- Type RW, reset 0x0000.0000

Registers:

- Register 64: PWM0 Fault Source 0 (PWM0FLTSRC0), offset 0x074
- Register 65: PWM1 Fault Source 0 (PWM1FLTSRC0), offset 0x0B4
- Register 66: PWM2 Fault Source 0 (PWM2FLTSRC0), offset 0x0F4
- Register 67: PWM3 Fault Source 0 (PWM3FLTSRC0), offset 0x134



Bit/Field	Name	Type	Reset	Description						
31:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
1	FAULT1	RW	0	<p>Fault1 Input</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The Fault1 signal is suppressed and cannot generate a fault condition.</td> </tr> <tr> <td>1</td> <td>The Fault1 signal value is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).</td> </tr> </tbody> </table> <p>Note: The FLTSRC bit in the PWMnCTL (page 322) register must be set for this bit to affect fault condition generation.</p>	Value	Description	0	The Fault1 signal is suppressed and cannot generate a fault condition.	1	The Fault1 signal value is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).
Value	Description									
0	The Fault1 signal is suppressed and cannot generate a fault condition.									
1	The Fault1 signal value is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).									

Bit/Field	Name	Type	Reset	Description						
0	FAULT0	RW	0	<p>Fault0 Input</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The Fault0 signal is suppressed and cannot generate a fault condition.</td> </tr> <tr> <td>1</td> <td>The Fault0 signal value is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).</td> </tr> </tbody> </table> <p>Note: The FLTSRC bit in the PWMnCTL (page 322) register must be set for this bit to affect fault condition generation.</p>	Value	Description	0	The Fault0 signal is suppressed and cannot generate a fault condition.	1	The Fault0 signal value is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).
Value	Description									
0	The Fault0 signal is suppressed and cannot generate a fault condition.									
1	The Fault0 signal value is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).									

23.4.14 PWMn Fault Source 1 (PWMnFLTSRC1)

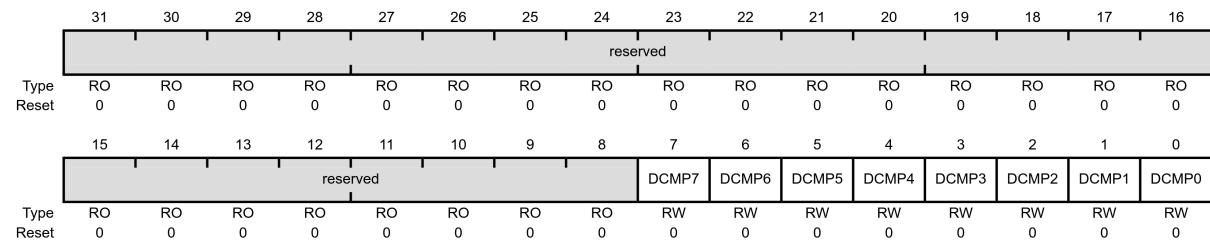
This register enables the PWM's fault source 1, which contains ADC's 8 digital comparators. It also specifies which digital comparator triggers from the ADC are used to generate a fault condition. Each bit in this register indicates whether the corresponding digital comparator trigger is included in the fault condition.

Documentation:

- PWM0 base: 0x4002.8000
- PWM1 base: 0x4002.9000
- Offset 0x078
- Type RW, reset 0x0000.0000

Registers:

- Register 68: PWM0 Fault Source 1 (PWM0FLTSRC1), offset 0x078
- Register 69: PWM1 Fault Source 1 (PWM1FLTSRC1), offset 0x0B8
- Register 70: PWM2 Fault Source 1 (PWM2FLTSRC1), offset 0x0F8
- Register 71: PWM3 Fault Source 1 (PWM3FLTSRC1), offset 0x138



Bit/Field	Name	Type	Reset	Description						
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
7	DCMP7	RW	0	<p>Digital Comparator 7</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The trigger from digital comparator 7 is suppressed and cannot generate a fault condition.</td> </tr> <tr> <td>1</td> <td>The trigger from digital comparator 7 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).</td> </tr> </tbody> </table> <p>Note: The FLTSRC bit in the PWMnCTL (page 322) register must be set for this bit to affect fault condition generation.</p>	Value	Description	0	The trigger from digital comparator 7 is suppressed and cannot generate a fault condition.	1	The trigger from digital comparator 7 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).
Value	Description									
0	The trigger from digital comparator 7 is suppressed and cannot generate a fault condition.									
1	The trigger from digital comparator 7 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).									

Bit/Field	Name	Type	Reset	Description						
6	DCMP6	RW	0	<p>Digital Comparator 6</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The trigger from digital comparator 6 is suppressed and cannot generate a fault condition.</td> </tr> <tr> <td>1</td> <td>The trigger from digital comparator 6 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).</td> </tr> </tbody> </table> <p>Note: The FLTSRC bit in the PWMMnCTL (page 322) register must be set for this bit to affect fault condition generation.</p>	Value	Description	0	The trigger from digital comparator 6 is suppressed and cannot generate a fault condition.	1	The trigger from digital comparator 6 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).
Value	Description									
0	The trigger from digital comparator 6 is suppressed and cannot generate a fault condition.									
1	The trigger from digital comparator 6 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).									
5	DCMP5	RW	0	<p>Digital Comparator 5</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The trigger from digital comparator 5 is suppressed and cannot generate a fault condition.</td> </tr> <tr> <td>1</td> <td>The trigger from digital comparator 5 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).</td> </tr> </tbody> </table> <p>Note: The FLTSRC bit in the PWMMnCTL (page 322) register must be set for this bit to affect fault condition generation.</p>	Value	Description	0	The trigger from digital comparator 5 is suppressed and cannot generate a fault condition.	1	The trigger from digital comparator 5 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).
Value	Description									
0	The trigger from digital comparator 5 is suppressed and cannot generate a fault condition.									
1	The trigger from digital comparator 5 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).									
4	DCMP4	RW	0	<p>Digital Comparator 4</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The trigger from digital comparator 4 is suppressed and cannot generate a fault condition.</td> </tr> <tr> <td>1</td> <td>The trigger from digital comparator 4 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).</td> </tr> </tbody> </table> <p>Note: The FLTSRC bit in the PWMMnCTL (page 322) register must be set for this bit to affect fault condition generation.</p>	Value	Description	0	The trigger from digital comparator 4 is suppressed and cannot generate a fault condition.	1	The trigger from digital comparator 4 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).
Value	Description									
0	The trigger from digital comparator 4 is suppressed and cannot generate a fault condition.									
1	The trigger from digital comparator 4 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).									

Bit/Field	Name	Type	Reset	Description						
3	DCMP3	RW	0	<p>Digital Comparator 3</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The trigger from digital comparator 3 is suppressed and cannot generate a fault condition.</td> </tr> <tr> <td>1</td> <td>The trigger from digital comparator 3 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).</td> </tr> </tbody> </table> <p>Note: The FLTSRC bit in the PWMMnCTL (page 322) register must be set for this bit to affect fault condition generation.</p>	Value	Description	0	The trigger from digital comparator 3 is suppressed and cannot generate a fault condition.	1	The trigger from digital comparator 3 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).
Value	Description									
0	The trigger from digital comparator 3 is suppressed and cannot generate a fault condition.									
1	The trigger from digital comparator 3 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).									
2	DCMP2	RW	0	<p>Digital Comparator 2</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The trigger from digital comparator 2 is suppressed and cannot generate a fault condition.</td> </tr> <tr> <td>1</td> <td>The trigger from digital comparator 2 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).</td> </tr> </tbody> </table> <p>Note: The FLTSRC bit in the PWMMnCTL (page 322) register must be set for this bit to affect fault condition generation.</p>	Value	Description	0	The trigger from digital comparator 2 is suppressed and cannot generate a fault condition.	1	The trigger from digital comparator 2 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).
Value	Description									
0	The trigger from digital comparator 2 is suppressed and cannot generate a fault condition.									
1	The trigger from digital comparator 2 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).									
1	DCMP1	RW	0	<p>Digital Comparator 1</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The trigger from digital comparator 1 is suppressed and cannot generate a fault condition.</td> </tr> <tr> <td>1</td> <td>The trigger from digital comparator 1 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).</td> </tr> </tbody> </table> <p>Note: The FLTSRC bit in the PWMMnCTL (page 322) register must be set for this bit to affect fault condition generation.</p>	Value	Description	0	The trigger from digital comparator 1 is suppressed and cannot generate a fault condition.	1	The trigger from digital comparator 1 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).
Value	Description									
0	The trigger from digital comparator 1 is suppressed and cannot generate a fault condition.									
1	The trigger from digital comparator 1 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).									

Bit/Field	Name	Type	Reset	Description						
0	DCMP0	RW	0	<p>Digital Comparator 0</p> <table> <thead> <tr> <th data-bbox="719 316 814 350">Value</th> <th data-bbox="814 316 973 350">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="719 361 814 473">0</td> <td data-bbox="814 361 1370 473">The trigger from digital comparator 0 is suppressed and cannot generate a fault condition.</td> </tr> <tr> <td data-bbox="719 485 814 631">1</td> <td data-bbox="814 485 1370 631">The trigger from digital comparator 0 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).</td> </tr> </tbody> </table> <p>Note: The FLTSRC bit in the PWMnCTL (page 322) register must be set for this bit to affect fault condition generation.</p>	Value	Description	0	The trigger from digital comparator 0 is suppressed and cannot generate a fault condition.	1	The trigger from digital comparator 0 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).
Value	Description									
0	The trigger from digital comparator 0 is suppressed and cannot generate a fault condition.									
1	The trigger from digital comparator 0 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).									

23.4.15 PWMn Dead-Band Control (PWMnDBCTL)

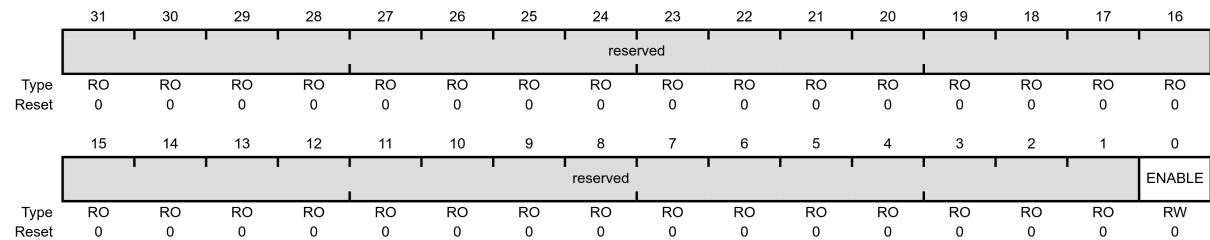
This register enables the use of the PWM's dead-band between pulses A' and B'.

Documentation:

- PWM0 base: 0x4002.8000
- PWM1 base: 0x4002.9000
- Offset 0x068
- Type RW, reset 0x0000.0000

Registers:

- Register 52: PWM0 Dead-Band Control (PWM0DBCTL), offset 0x068
- Register 53: PWM1 Dead-Band Control (PWM1DBCTL), offset 0x0A8
- Register 54: PWM2 Dead-Band Control (PWM2DBCTL), offset 0x0E8
- Register 55: PWM3 Dead-Band Control (PWM3DBCTL), offset 0x128



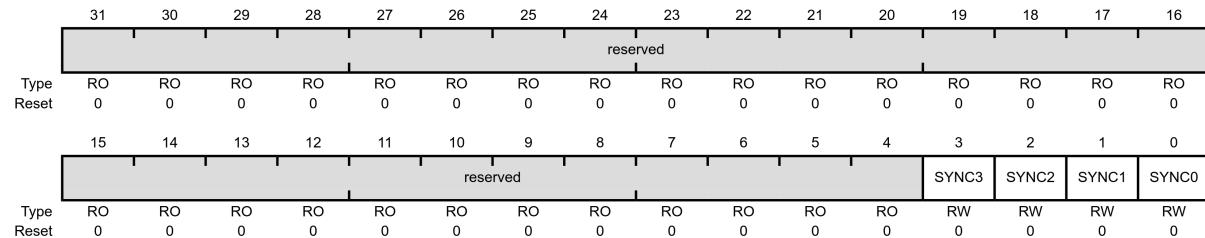
Bit/Field	Name	Type	Reset	Description						
31:1	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
0	ENABLE	RW	0	<p>Dead-Band Generator Enable</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The pwmA and pwmB signals pass through to the pwmA' and pwmB' signals unmodified.</td> </tr> <tr> <td>1</td> <td>The dead-band generator modifies the pwmA signal by inserting dead bands into the pwmA' and pwmB' signals.</td> </tr> </tbody> </table>	Value	Description	0	The pwmA and pwmB signals pass through to the pwmA' and pwmB' signals unmodified.	1	The dead-band generator modifies the pwmA signal by inserting dead bands into the pwmA' and pwmB' signals.
Value	Description									
0	The pwmA and pwmB signals pass through to the pwmA' and pwmB' signals unmodified.									
1	The dead-band generator modifies the pwmA signal by inserting dead bands into the pwmA' and pwmB' signals.									

23.4.16 PWM Time Base Sync (PWMSYNC)

This register provides a method to perform synchronisation of the counters in the PWM generation blocks. Setting a bit in this register causes the specified counter to reset back to 0.

Documentation:

- PWM0 base: 0x4002.8000
- PWM1 base: 0x4002.9000
- Offset 0x004
- Type RW, reset 0x0000.0000



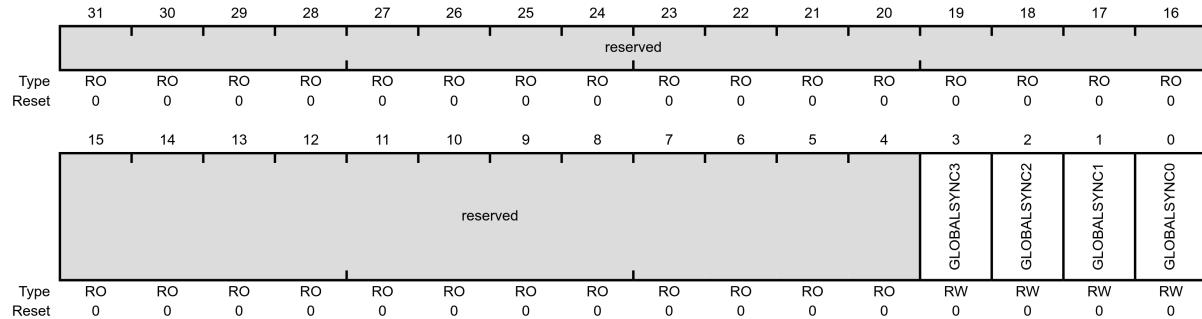
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	SYNC3	RW	0	Reset Generator 3 Counter Value Description 0 No effect. 1 Resets the PWM generator 3 counter.
2	SYNC2	RW	0	Reset Generator 2 Counter Value Description 0 No effect. 1 Resets the PWM generator 2 counter.
1	SYNC1	RW	0	Reset Generator 1 Counter Value Description 0 No effect. 1 Resets the PWM generator 1 counter.
0	SYNC0	RW	0	Reset Generator 0 Counter Value Description 0 No effect. 1 Resets the PWM generator 0 counter.

23.4.17 PWM Master Control (PWMCTL)

This register provides master control over the PWM generation blocks.

Documentation:

- PWM0 base: 0x4002.8000
- PWM1 base: 0x4002.9000
- Offset 0x000
- Type RW, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description						
31:4	reserved	RO	0x0000	<p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p>						
3	GLOBALSYNC3	RW	0	<p>Update PWM Generator 3</p> <table> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>No effect.</td> </tr> <tr> <td>1</td> <td>Any queued update to a load or comparator register in PWM generator 3 is applied the next time the corresponding counter becomes zero.</td> </tr> </table> <p>This bit automatically clears when the updates have completed; it cannot be cleared by software.</p>	Value	Description	0	No effect.	1	Any queued update to a load or comparator register in PWM generator 3 is applied the next time the corresponding counter becomes zero.
Value	Description									
0	No effect.									
1	Any queued update to a load or comparator register in PWM generator 3 is applied the next time the corresponding counter becomes zero.									
2	GLOBALSYNC2	RW	0	<p>Update PWM Generator 2</p> <table> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>No effect.</td> </tr> <tr> <td>1</td> <td>Any queued update to a load or comparator register in PWM generator 2 is applied the next time the corresponding counter becomes zero.</td> </tr> </table> <p>This bit automatically clears when the updates have completed; it cannot be cleared by software.</p>	Value	Description	0	No effect.	1	Any queued update to a load or comparator register in PWM generator 2 is applied the next time the corresponding counter becomes zero.
Value	Description									
0	No effect.									
1	Any queued update to a load or comparator register in PWM generator 2 is applied the next time the corresponding counter becomes zero.									

Bit/Field	Name	Type	Reset	Description						
1	GLOBALSYNC1	RW	0	<p>Update PWM Generator 1</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No effect.</td> </tr> <tr> <td>1</td> <td>Any queued update to a load or comparator register in PWM generator 1 is applied the next time the corresponding counter becomes zero.</td> </tr> </tbody> </table> <p>This bit automatically clears when the updates have completed; it cannot be cleared by software.</p>	Value	Description	0	No effect.	1	Any queued update to a load or comparator register in PWM generator 1 is applied the next time the corresponding counter becomes zero.
Value	Description									
0	No effect.									
1	Any queued update to a load or comparator register in PWM generator 1 is applied the next time the corresponding counter becomes zero.									
0	GLOBALSYNC0	RW	0	<p>Update PWM Generator 0</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No effect.</td> </tr> <tr> <td>1</td> <td>Any queued update to a load or comparator register in PWM generator 0 is applied the next time the corresponding counter becomes zero.</td> </tr> </tbody> </table> <p>This bit automatically clears when the updates have completed; it cannot be cleared by software.</p>	Value	Description	0	No effect.	1	Any queued update to a load or comparator register in PWM generator 0 is applied the next time the corresponding counter becomes zero.
Value	Description									
0	No effect.									
1	Any queued update to a load or comparator register in PWM generator 0 is applied the next time the corresponding counter becomes zero.									

23.4.18 PWMn Minimum Fault Period (PWMrMINFLTPER)

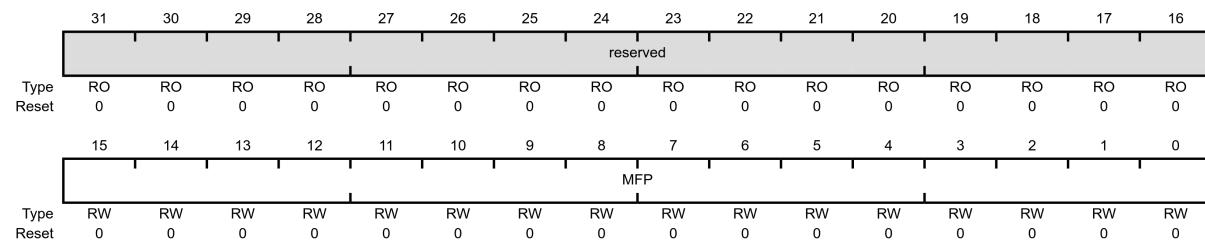
This register sets the minimum delay period before releasing the fault condition. It specifies the 16-bit time extension value to be used in extending the fault condition.

Documentation:

- PWM0 base: 0x4002.8000
- PWM1 base: 0x4002.9000
- Offset 0x07C
- Type RW, reset 0x0000.0000

Registers:

- Register 72: PWM0 Minimum Fault Period (PWMrMINFLTPER), offset 0x07C
- Register 73: PWM1 Minimum Fault Period (PWMrMINFLTPER), offset 0x0BC
- Register 74: PWM2 Minimum Fault Period (PWMrMINFLTPER), offset 0x0FC
- Register 75: PWM3 Minimum Fault Period (PWMrMINFLTPER), offset 0x13C



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	MFP	RW	0x0000	Minimum Fault Period The number of PWM clocks by which a fault condition is extended when the delay is enabled by PWMrCTL (page 322) MINFLIPER.

23.4.19 PWMn Fault Pin Logic Sense (PWMnFLTSEN)

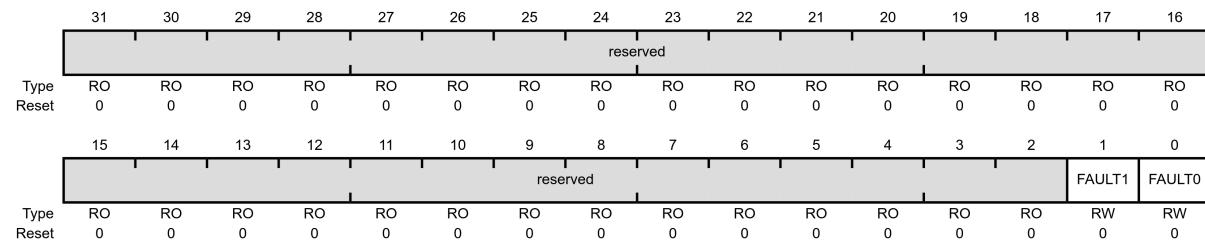
This register defines the PWM fault pin logic sense.

Documentation:

- PWM0 base: 0x4002.8000
- PWM1 base: 0x4002.9000
- Offset 0x800
- Type RW, reset 0x0000.0000

Registers:

- Register 76: PWM0 Fault Pin Logic Sense (PWM0FLTSEN), offset 0x800
- Register 77: PWM1 Fault Pin Logic Sense (PWM1FLTSEN), offset 0x880



Bit/Field	Name	Type	Reset	Description						
31:2	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
1	FAULT1	RW	0	<p>Fault1 Sense</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>An error is indicated if the Fault1 signal is High.</td> </tr> <tr> <td>1</td> <td>An error is indicated if the Fault1 signal is Low.</td> </tr> </tbody> </table>	Value	Description	0	An error is indicated if the Fault1 signal is High.	1	An error is indicated if the Fault1 signal is Low.
Value	Description									
0	An error is indicated if the Fault1 signal is High.									
1	An error is indicated if the Fault1 signal is Low.									
0	FAULT0	RW	0	<p>Fault0 Sense</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>An error is indicated if the Fault0 signal is High.</td> </tr> <tr> <td>1</td> <td>An error is indicated if the Fault0 signal is Low.</td> </tr> </tbody> </table>	Value	Description	0	An error is indicated if the Fault0 signal is High.	1	An error is indicated if the Fault0 signal is Low.
Value	Description									
0	An error is indicated if the Fault0 signal is High.									
1	An error is indicated if the Fault0 signal is Low.									

23.4.20 PWM Enable Update (PWMENUPD)

This register specifies when updates to the **PWMnEN** bit in **PWMENABLE** (page 335) register are performed.

Documentation:

- PWM0 base: 0x4002.8000
- PWM1 base: 0x4002.9000
- Offset 0x028
- Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	ENUPD7	ENUPD6	ENUPD5	ENUPD4	ENUPD3	ENUPD2	ENUPD1	ENUPD0								
Reset	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit/Field	Name	Type	Reset	Description										
31:16	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.										
15:14	ENUPD7	RW	0	<p>MnPWM7 Enable Update Mode</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Immediate Writes to the PWM7EN bit in the PWMENABLE (page 335) register are used by the PWM generator immediately.</td> </tr> <tr> <td>0x1</td> <td>Reserved</td> </tr> <tr> <td>0x2</td> <td>Locally Synchronized Writes to the PWM7EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0.</td> </tr> <tr> <td>0x3</td> <td>Globally Synchronized Writes to the PWM7EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL (page 347)) register.</td> </tr> </tbody> </table>	Value	Description	0x0	Immediate Writes to the PWM7EN bit in the PWMENABLE (page 335) register are used by the PWM generator immediately.	0x1	Reserved	0x2	Locally Synchronized Writes to the PWM7EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0.	0x3	Globally Synchronized Writes to the PWM7EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL (page 347)) register.
Value	Description													
0x0	Immediate Writes to the PWM7EN bit in the PWMENABLE (page 335) register are used by the PWM generator immediately.													
0x1	Reserved													
0x2	Locally Synchronized Writes to the PWM7EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0.													
0x3	Globally Synchronized Writes to the PWM7EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL (page 347)) register.													

Bit/Field	Name	Type	Reset	Description																
13:12	ENUPD6	RW	0	<p>MnPWM6 Enable Update Mode</p> <table> <thead> <tr> <th data-bbox="668 316 755 350">Value</th> <th data-bbox="755 316 922 350">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="668 361 755 395">0x0</td> <td data-bbox="755 361 906 395">Immediate</td> </tr> <tr> <td data-bbox="668 406 755 518">0x1</td> <td data-bbox="755 406 890 518">Writes to the PWM6EN bit in the PWMENABLE (page 335) register are used by the PWM generator immediately.</td> </tr> <tr> <td data-bbox="668 530 755 563">0x2</td> <td data-bbox="755 530 1033 563">Reserved</td> </tr> <tr> <td data-bbox="668 574 755 687">0x3</td> <td data-bbox="755 574 1033 687">Locally Synchronized</td> </tr> <tr> <td data-bbox="668 698 755 1017"></td> <td data-bbox="755 698 1367 1017">Writes to the PWM6EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0.</td> </tr> <tr> <td data-bbox="668 1028 755 1051"></td> <td data-bbox="755 1028 1367 1051">Globally Synchronized</td> </tr> <tr> <td data-bbox="668 1062 755 1096"></td> <td data-bbox="755 1062 1367 1096">Writes to the PWM6EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL (page 347)) register.</td> </tr> </tbody> </table>	Value	Description	0x0	Immediate	0x1	Writes to the PWM6EN bit in the PWMENABLE (page 335) register are used by the PWM generator immediately.	0x2	Reserved	0x3	Locally Synchronized		Writes to the PWM6EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0.		Globally Synchronized		Writes to the PWM6EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL (page 347)) register.
Value	Description																			
0x0	Immediate																			
0x1	Writes to the PWM6EN bit in the PWMENABLE (page 335) register are used by the PWM generator immediately.																			
0x2	Reserved																			
0x3	Locally Synchronized																			
	Writes to the PWM6EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0.																			
	Globally Synchronized																			
	Writes to the PWM6EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL (page 347)) register.																			
11:10	ENUPD5	RW	0	<p>MnPWM5 Enable Update Mode</p> <table> <thead> <tr> <th data-bbox="668 1131 755 1165">Value</th> <th data-bbox="755 1131 922 1165">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="668 1176 755 1210">0x0</td> <td data-bbox="755 1176 906 1210">Immediate</td> </tr> <tr> <td data-bbox="668 1221 755 1334">0x1</td> <td data-bbox="755 1221 890 1334">Writes to the PWM5EN bit in the PWMENABLE (page 335) register are used by the PWM generator immediately.</td> </tr> <tr> <td data-bbox="668 1367 755 1401">0x2</td> <td data-bbox="755 1367 1033 1401">Reserved</td> </tr> <tr> <td data-bbox="668 1412 755 1525">0x3</td> <td data-bbox="755 1412 1033 1525">Locally Synchronized</td> </tr> <tr> <td data-bbox="668 1536 755 1648"></td> <td data-bbox="755 1536 1367 1648">Writes to the PWM5EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0.</td> </tr> <tr> <td data-bbox="668 1659 755 1843"></td> <td data-bbox="755 1659 1367 1843">Globally Synchronized</td> </tr> <tr> <td data-bbox="668 1855 755 1877"></td> <td data-bbox="755 1855 1367 1877">Writes to the PWM5EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL (page 347)) register.</td> </tr> </tbody> </table>	Value	Description	0x0	Immediate	0x1	Writes to the PWM5EN bit in the PWMENABLE (page 335) register are used by the PWM generator immediately.	0x2	Reserved	0x3	Locally Synchronized		Writes to the PWM5EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0.		Globally Synchronized		Writes to the PWM5EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL (page 347)) register.
Value	Description																			
0x0	Immediate																			
0x1	Writes to the PWM5EN bit in the PWMENABLE (page 335) register are used by the PWM generator immediately.																			
0x2	Reserved																			
0x3	Locally Synchronized																			
	Writes to the PWM5EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0.																			
	Globally Synchronized																			
	Writes to the PWM5EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL (page 347)) register.																			

Bit/Field	Name	Type	Reset	Description																
9:8	ENUPD4	RW	0	<p>MnPWM4 Enable Update Mode</p> <table> <thead> <tr> <th data-bbox="668 316 732 350">Value</th> <th data-bbox="732 316 922 350">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="668 361 732 395">0x0</td> <td data-bbox="732 361 906 395">Immediate</td> </tr> <tr> <td data-bbox="668 406 732 518">0x1</td> <td data-bbox="732 406 890 518">Writes to the PWM4EN bit in the PWMENABLE (page 335) register are used by the PWM generator immediately.</td> </tr> <tr> <td data-bbox="668 530 732 563">0x2</td> <td data-bbox="732 530 1033 563">Reserved</td> </tr> <tr> <td data-bbox="668 574 732 687">0x3</td> <td data-bbox="732 574 1033 687">Locally Synchronized</td> </tr> <tr> <td data-bbox="668 698 732 1035"></td> <td data-bbox="732 698 1367 1035">Writes to the PWM4EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0.</td> </tr> <tr> <td data-bbox="668 1046 732 1080"></td> <td data-bbox="732 1046 1367 1080">Globally Synchronized</td> </tr> <tr> <td data-bbox="668 1091 732 1102"></td> <td data-bbox="732 1091 1367 1102">Writes to the PWM4EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL (page 347)) register.</td> </tr> </tbody> </table>	Value	Description	0x0	Immediate	0x1	Writes to the PWM4EN bit in the PWMENABLE (page 335) register are used by the PWM generator immediately.	0x2	Reserved	0x3	Locally Synchronized		Writes to the PWM4EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0.		Globally Synchronized		Writes to the PWM4EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL (page 347)) register.
Value	Description																			
0x0	Immediate																			
0x1	Writes to the PWM4EN bit in the PWMENABLE (page 335) register are used by the PWM generator immediately.																			
0x2	Reserved																			
0x3	Locally Synchronized																			
	Writes to the PWM4EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0.																			
	Globally Synchronized																			
	Writes to the PWM4EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL (page 347)) register.																			
7:6	ENUPD3	RW	0	<p>MnPWM3 Enable Update Mode</p> <table> <thead> <tr> <th data-bbox="668 1131 732 1165">Value</th> <th data-bbox="732 1131 922 1165">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="668 1176 732 1210">0x0</td> <td data-bbox="732 1176 906 1210">Immediate</td> </tr> <tr> <td data-bbox="668 1221 732 1356">0x1</td> <td data-bbox="732 1221 890 1356">Writes to the PWM3EN bit in the PWMENABLE (page 335) register are used by the PWM generator immediately.</td> </tr> <tr> <td data-bbox="668 1367 732 1401">0x2</td> <td data-bbox="732 1367 1033 1401">Reserved</td> </tr> <tr> <td data-bbox="668 1412 732 1525">0x3</td> <td data-bbox="732 1412 1033 1525">Locally Synchronized</td> </tr> <tr> <td data-bbox="668 1536 732 1648"></td> <td data-bbox="732 1536 1367 1648">Writes to the PWM3EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0.</td> </tr> <tr> <td data-bbox="668 1659 732 1861"></td> <td data-bbox="732 1659 1367 1861">Globally Synchronized</td> </tr> <tr> <td data-bbox="668 1873 732 1906"></td> <td data-bbox="732 1873 1367 1906">Writes to the PWM3EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL (page 347)) register.</td> </tr> </tbody> </table>	Value	Description	0x0	Immediate	0x1	Writes to the PWM3EN bit in the PWMENABLE (page 335) register are used by the PWM generator immediately.	0x2	Reserved	0x3	Locally Synchronized		Writes to the PWM3EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0.		Globally Synchronized		Writes to the PWM3EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL (page 347)) register.
Value	Description																			
0x0	Immediate																			
0x1	Writes to the PWM3EN bit in the PWMENABLE (page 335) register are used by the PWM generator immediately.																			
0x2	Reserved																			
0x3	Locally Synchronized																			
	Writes to the PWM3EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0.																			
	Globally Synchronized																			
	Writes to the PWM3EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL (page 347)) register.																			

Bit/Field	Name	Type	Reset	Description																
5:4	ENUPD2	RW	0	<p>MnPWM2 Enable Update Mode</p> <table> <thead> <tr> <th data-bbox="671 316 751 350">Value</th> <th data-bbox="751 316 925 350">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="671 361 751 395">0x0</td> <td data-bbox="751 361 925 395">Immediate</td> </tr> <tr> <td data-bbox="671 406 751 507">0x1</td> <td data-bbox="751 406 925 507">Writes to the PWM2EN bit in the PWMENABLE (page 335) register are used by the PWM generator immediately.</td> </tr> <tr> <td data-bbox="671 518 751 552">0x2</td> <td data-bbox="751 518 925 552">Reserved</td> </tr> <tr> <td data-bbox="671 563 751 597">0x3</td> <td data-bbox="751 563 925 597">Locally Synchronized</td> </tr> <tr> <td data-bbox="671 608 751 1017"></td> <td data-bbox="751 608 1391 1017">Writes to the PWM2EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0.</td> </tr> <tr> <td data-bbox="671 743 751 934"></td> <td data-bbox="751 743 1391 934">Globally Synchronized</td> </tr> <tr> <td data-bbox="671 788 751 1017"></td> <td data-bbox="751 788 1391 1017">Writes to the PWM2EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL (page 347)) register.</td> </tr> </tbody> </table>	Value	Description	0x0	Immediate	0x1	Writes to the PWM2EN bit in the PWMENABLE (page 335) register are used by the PWM generator immediately.	0x2	Reserved	0x3	Locally Synchronized		Writes to the PWM2EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0.		Globally Synchronized		Writes to the PWM2EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL (page 347)) register.
Value	Description																			
0x0	Immediate																			
0x1	Writes to the PWM2EN bit in the PWMENABLE (page 335) register are used by the PWM generator immediately.																			
0x2	Reserved																			
0x3	Locally Synchronized																			
	Writes to the PWM2EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0.																			
	Globally Synchronized																			
	Writes to the PWM2EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL (page 347)) register.																			
3:2	ENUPD1	RW	0	<p>MnPWM1 Enable Update Mode</p> <table> <thead> <tr> <th data-bbox="671 1131 751 1165">Value</th> <th data-bbox="751 1131 925 1165">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="671 1176 751 1210">0x0</td> <td data-bbox="751 1176 925 1210">Immediate</td> </tr> <tr> <td data-bbox="671 1221 751 1322">0x1</td> <td data-bbox="751 1221 925 1322">Writes to the PWM1EN bit in the PWMENABLE (page 335) register are used by the PWM generator immediately.</td> </tr> <tr> <td data-bbox="671 1334 751 1367">0x2</td> <td data-bbox="751 1334 925 1367">Reserved</td> </tr> <tr> <td data-bbox="671 1379 751 1563">0x3</td> <td data-bbox="751 1379 925 1563">Locally Synchronized</td> </tr> <tr> <td data-bbox="671 1439 751 1585"></td> <td data-bbox="751 1439 1391 1585">Writes to the PWM1EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0.</td> </tr> <tr> <td data-bbox="671 1596 751 1781"></td> <td data-bbox="751 1596 1391 1781">Globally Synchronized</td> </tr> <tr> <td data-bbox="671 1619 751 1848"></td> <td data-bbox="751 1619 1391 1848">Writes to the PWM1EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL (page 347)) register.</td> </tr> </tbody> </table>	Value	Description	0x0	Immediate	0x1	Writes to the PWM1EN bit in the PWMENABLE (page 335) register are used by the PWM generator immediately.	0x2	Reserved	0x3	Locally Synchronized		Writes to the PWM1EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0.		Globally Synchronized		Writes to the PWM1EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL (page 347)) register.
Value	Description																			
0x0	Immediate																			
0x1	Writes to the PWM1EN bit in the PWMENABLE (page 335) register are used by the PWM generator immediately.																			
0x2	Reserved																			
0x3	Locally Synchronized																			
	Writes to the PWM1EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0.																			
	Globally Synchronized																			
	Writes to the PWM1EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL (page 347)) register.																			

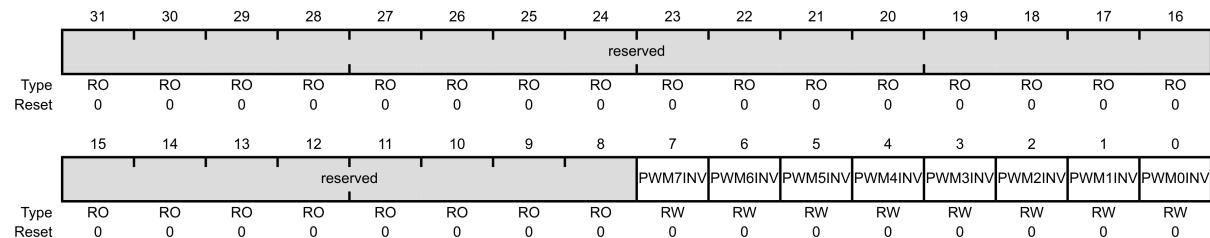
Bit/Field	Name	Type	Reset	Description																
1:0	ENUPD0	RW	0	<p>MnPWM0 Enable Update Mode</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Immediate</td> </tr> <tr> <td></td> <td>Writes to the PWM0EN bit in the PWMENABLE (page 335) register are used by the PWM generator immediately.</td> </tr> <tr> <td>0x1</td> <td>Reserved</td> </tr> <tr> <td>0x2</td> <td>Locally Synchronized</td> </tr> <tr> <td></td> <td>Writes to the PWM0EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0.</td> </tr> <tr> <td>0x3</td> <td>Globally Synchronized</td> </tr> <tr> <td></td> <td>Writes to the PWM0EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL (page 347)) register.</td> </tr> </tbody> </table>	Value	Description	0x0	Immediate		Writes to the PWM0EN bit in the PWMENABLE (page 335) register are used by the PWM generator immediately.	0x1	Reserved	0x2	Locally Synchronized		Writes to the PWM0EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0.	0x3	Globally Synchronized		Writes to the PWM0EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL (page 347)) register.
Value	Description																			
0x0	Immediate																			
	Writes to the PWM0EN bit in the PWMENABLE (page 335) register are used by the PWM generator immediately.																			
0x1	Reserved																			
0x2	Locally Synchronized																			
	Writes to the PWM0EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0.																			
0x3	Globally Synchronized																			
	Writes to the PWM0EN bit in the PWMENABLE (page 335) register are used by the PWM generator the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL (page 347)) register.																			

23.4.21 PWM Output Inversion (PWMINVERT)

This register provides a master control of the polarity of the MnPWMn signals on the device pins.

Documentation:

- PWM0 base: 0x4002.8000
- PWM1 base: 0x4002.9000
- Offset 0x00C
- Type RW, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description						
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
7	PWM7INV	RW	0	<p>Invert MnPWM7 Signal</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The MnPWM7 signal is not inverted.</td> </tr> <tr> <td>1</td> <td>The MnPWM7 signal is inverted.</td> </tr> </tbody> </table>	Value	Description	0	The MnPWM7 signal is not inverted.	1	The MnPWM7 signal is inverted.
Value	Description									
0	The MnPWM7 signal is not inverted.									
1	The MnPWM7 signal is inverted.									
6	PWM6INV	RW	0	<p>Invert MnPWM6 Signal</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The MnPWM6 signal is not inverted.</td> </tr> <tr> <td>1</td> <td>The MnPWM6 signal is inverted.</td> </tr> </tbody> </table>	Value	Description	0	The MnPWM6 signal is not inverted.	1	The MnPWM6 signal is inverted.
Value	Description									
0	The MnPWM6 signal is not inverted.									
1	The MnPWM6 signal is inverted.									
5	PWM5INV	RW	0	<p>Invert MnPWM5 Signal</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The MnPWM5 signal is not inverted.</td> </tr> <tr> <td>1</td> <td>The MnPWM5 signal is inverted.</td> </tr> </tbody> </table>	Value	Description	0	The MnPWM5 signal is not inverted.	1	The MnPWM5 signal is inverted.
Value	Description									
0	The MnPWM5 signal is not inverted.									
1	The MnPWM5 signal is inverted.									
4	PWM4INV	RW	0	<p>Invert MnPWM4 Signal</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The MnPWM4 signal is not inverted.</td> </tr> <tr> <td>1</td> <td>The MnPWM4 signal is inverted.</td> </tr> </tbody> </table>	Value	Description	0	The MnPWM4 signal is not inverted.	1	The MnPWM4 signal is inverted.
Value	Description									
0	The MnPWM4 signal is not inverted.									
1	The MnPWM4 signal is inverted.									

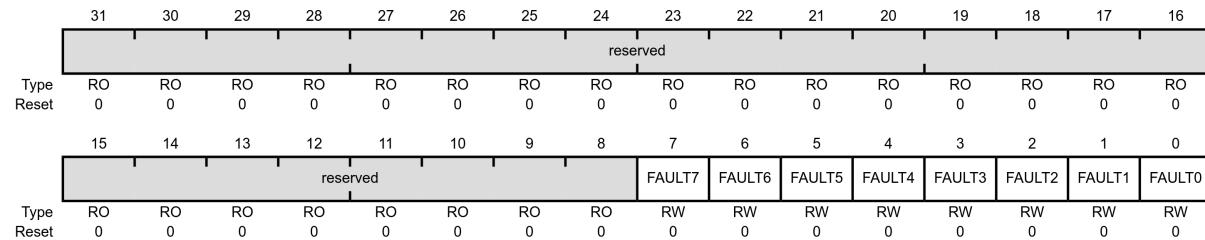
Bit/Field	Name	Type	Reset	Description						
3	PWM3INV	RW	0	<p>Invert MnPWM3 Signal</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The MnPWM3 signal is not inverted.</td> </tr> <tr> <td>1</td> <td>The MnPWM3 signal is inverted.</td> </tr> </tbody> </table>	Value	Description	0	The MnPWM3 signal is not inverted.	1	The MnPWM3 signal is inverted.
Value	Description									
0	The MnPWM3 signal is not inverted.									
1	The MnPWM3 signal is inverted.									
2	PWM2INV	RW	0	<p>Invert MnPWM2 Signal</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The MnPWM2 signal is not inverted.</td> </tr> <tr> <td>1</td> <td>The MnPWM2 signal is inverted.</td> </tr> </tbody> </table>	Value	Description	0	The MnPWM2 signal is not inverted.	1	The MnPWM2 signal is inverted.
Value	Description									
0	The MnPWM2 signal is not inverted.									
1	The MnPWM2 signal is inverted.									
1	PWM1INV	RW	0	<p>Invert MnPWM1 Signal</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The MnPWM1 signal is not inverted.</td> </tr> <tr> <td>1</td> <td>The MnPWM1 signal is inverted.</td> </tr> </tbody> </table>	Value	Description	0	The MnPWM1 signal is not inverted.	1	The MnPWM1 signal is inverted.
Value	Description									
0	The MnPWM1 signal is not inverted.									
1	The MnPWM1 signal is inverted.									
0	PWM0INV	RW	0	<p>Invert MnPWM0 Signal</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The MnPWM0 signal is not inverted.</td> </tr> <tr> <td>1</td> <td>The MnPWM0 signal is inverted.</td> </tr> </tbody> </table>	Value	Description	0	The MnPWM0 signal is not inverted.	1	The MnPWM0 signal is inverted.
Value	Description									
0	The MnPWM0 signal is not inverted.									
1	The MnPWM0 signal is inverted.									

23.4.22 PWM Output Fault (PWMFAULT)

This register controls the behaviour of the MnPWMn outputs in the presence of fault conditions.

Documentation:

- PWM0 base: 0x4002.8000
- PWM1 base: 0x4002.9000
- Offset 0x010
- Type RW, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description						
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
7	FAULT7	RW	0	<p>MnPWM7 Fault</p> <table> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>The generated pwm3B' signal is passed to the MnPWM7 pin.</td> </tr> <tr> <td>1</td> <td>The MnPWM7 output signal is driven to the value specified by the PWM7 bit in the PWMFAULTVAL (page 360) register.</td> </tr> </table>	Value	Description	0	The generated pwm3B' signal is passed to the MnPWM7 pin.	1	The MnPWM7 output signal is driven to the value specified by the PWM7 bit in the PWMFAULTVAL (page 360) register.
Value	Description									
0	The generated pwm3B' signal is passed to the MnPWM7 pin.									
1	The MnPWM7 output signal is driven to the value specified by the PWM7 bit in the PWMFAULTVAL (page 360) register.									
6	FAULT6	RW	0	<p>MnPWM6 Fault</p> <table> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>The generated pwm3A' signal is passed to the MnPWM6 pin.</td> </tr> <tr> <td>1</td> <td>The MnPWM6 output signal is driven to the value specified by the PWM6 bit in the PWMFAULTVAL (page 360) register.</td> </tr> </table>	Value	Description	0	The generated pwm3A' signal is passed to the MnPWM6 pin.	1	The MnPWM6 output signal is driven to the value specified by the PWM6 bit in the PWMFAULTVAL (page 360) register.
Value	Description									
0	The generated pwm3A' signal is passed to the MnPWM6 pin.									
1	The MnPWM6 output signal is driven to the value specified by the PWM6 bit in the PWMFAULTVAL (page 360) register.									
5	FAULT5	RW	0	<p>MnPWM5 Fault</p> <table> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>The generated pwm2B' signal is passed to the MnPWM5 pin.</td> </tr> <tr> <td>1</td> <td>The MnPWM5 output signal is driven to the value specified by the PWM5 bit in the PWMFAULTVAL (page 360) register.</td> </tr> </table>	Value	Description	0	The generated pwm2B' signal is passed to the MnPWM5 pin.	1	The MnPWM5 output signal is driven to the value specified by the PWM5 bit in the PWMFAULTVAL (page 360) register.
Value	Description									
0	The generated pwm2B' signal is passed to the MnPWM5 pin.									
1	The MnPWM5 output signal is driven to the value specified by the PWM5 bit in the PWMFAULTVAL (page 360) register.									

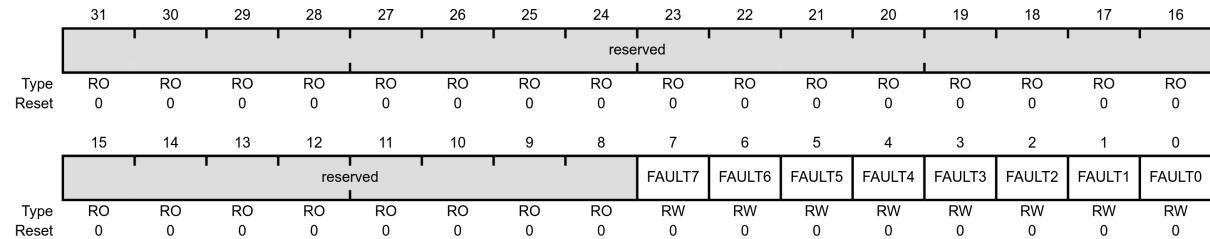
Bit/Field	Name	Type	Reset	Description
4	FAULT4	RW	0	MnPWM4 Fault Value Description 0 The generated pwm2A' signal is passed to the MnPWM4 pin. 1 The MnPWM4 output signal is driven to the value specified by the PWM4 bit in the PWMFAULTVAL (page 360) register.
3	FAULT3	RW	0	MnPWM3 Fault Value Description 0 The generated pwm1B' signal is passed to the MnPWM3 pin. 1 The MnPWM3 output signal is driven to the value specified by the PWM3 bit in the PWMFAULTVAL (page 360) register.
2	FAULT2	RW	0	MnPWM2 Fault Value Description 0 The generated pwm1A' signal is passed to the MnPWM2 pin. 1 The MnPWM2 output signal is driven to the value specified by the PWM2 bit in the PWMFAULTVAL (page 360) register.
1	FAULT1	RW	0	MnPWM1 Fault Value Description 0 The generated pwm0B' signal is passed to the MnPWM1 pin. 1 The MnPWM1 output signal is driven to the value specified by the PWM1 bit in the PWMFAULTVAL (page 360) register.
0	FAULT0	RW	0	MnPWM0 Fault Value Description 0 The generated pwm0A' signal is passed to the MnPWM0 pin. 1 The MnPWM0 output signal is driven to the value specified by the PWM0 bit in the PWMFAULTVAL (page 360) register.

23.4.23 PWM Fault Condition Value (PWMFAULTVAL)

This register sets the output value (high or low) of fault conditions. It specifies the output value driven on the MnPWM_n signals during a fault condition if enabled by the corresponding bit in the **PWMFAULT** (page 358) register.

Documentation:

- PWM0 base: 0x4002.8000
- PWM1 base: 0x4002.9000
- Offset 0x024
- Type RW, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description						
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
7	PWM7	RW	0	<p>MnPWM7 Fault Value</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The MnPWM7 output signal is driven Low during fault conditions if the FAULT7 bit in the PWMFAULT (page 358) register is set.</td> </tr> <tr> <td>1</td> <td>The MnPWM7 output signal is driven High during fault conditions if the FAULT7 bit in the PWMFAULT (page 358) register is set.</td> </tr> </tbody> </table>	Value	Description	0	The MnPWM7 output signal is driven Low during fault conditions if the FAULT7 bit in the PWMFAULT (page 358) register is set.	1	The MnPWM7 output signal is driven High during fault conditions if the FAULT7 bit in the PWMFAULT (page 358) register is set.
Value	Description									
0	The MnPWM7 output signal is driven Low during fault conditions if the FAULT7 bit in the PWMFAULT (page 358) register is set.									
1	The MnPWM7 output signal is driven High during fault conditions if the FAULT7 bit in the PWMFAULT (page 358) register is set.									
6	PWM6	RW	0	<p>MnPWM6 Fault Value</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The MnPWM6 output signal is driven Low during fault conditions if the FAULT6 bit in the PWMFAULT (page 358) register is set.</td> </tr> <tr> <td>1</td> <td>The MnPWM6 output signal is driven High during fault conditions if the FAULT6 bit in the PWMFAULT (page 358) register is set.</td> </tr> </tbody> </table>	Value	Description	0	The MnPWM6 output signal is driven Low during fault conditions if the FAULT6 bit in the PWMFAULT (page 358) register is set.	1	The MnPWM6 output signal is driven High during fault conditions if the FAULT6 bit in the PWMFAULT (page 358) register is set.
Value	Description									
0	The MnPWM6 output signal is driven Low during fault conditions if the FAULT6 bit in the PWMFAULT (page 358) register is set.									
1	The MnPWM6 output signal is driven High during fault conditions if the FAULT6 bit in the PWMFAULT (page 358) register is set.									

Bit/Field	Name	Type	Reset	Description						
5	PWM5	RW	0	<p>MnPWM5 Fault Value</p> <table> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>The MnPWM5 output signal is driven Low during fault conditions if the FAULT5 bit in the PWMFAULT (page 358) register is set.</td> </tr> <tr> <td>1</td> <td>The MnPWM5 output signal is driven High during fault conditions if the FAULT5 bit in the PWMFAULT (page 358) register is set.</td> </tr> </table>	Value	Description	0	The MnPWM5 output signal is driven Low during fault conditions if the FAULT5 bit in the PWMFAULT (page 358) register is set.	1	The MnPWM5 output signal is driven High during fault conditions if the FAULT5 bit in the PWMFAULT (page 358) register is set.
Value	Description									
0	The MnPWM5 output signal is driven Low during fault conditions if the FAULT5 bit in the PWMFAULT (page 358) register is set.									
1	The MnPWM5 output signal is driven High during fault conditions if the FAULT5 bit in the PWMFAULT (page 358) register is set.									
4	PWM4	RW	0	<p>MnPWM4 Fault Value</p> <table> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>The MnPWM4 output signal is driven Low during fault conditions if the FAULT4 bit in the PWMFAULT (page 358) register is set.</td> </tr> <tr> <td>1</td> <td>The MnPWM4 output signal is driven High during fault conditions if the FAULT4 bit in the PWMFAULT (page 358) register is set.</td> </tr> </table>	Value	Description	0	The MnPWM4 output signal is driven Low during fault conditions if the FAULT4 bit in the PWMFAULT (page 358) register is set.	1	The MnPWM4 output signal is driven High during fault conditions if the FAULT4 bit in the PWMFAULT (page 358) register is set.
Value	Description									
0	The MnPWM4 output signal is driven Low during fault conditions if the FAULT4 bit in the PWMFAULT (page 358) register is set.									
1	The MnPWM4 output signal is driven High during fault conditions if the FAULT4 bit in the PWMFAULT (page 358) register is set.									
3	PWM3	RW	0	<p>MnPWM3 Fault Value</p> <table> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>The MnPWM3 output signal is driven Low during fault conditions if the FAULT3 bit in the PWMFAULT (page 358) register is set.</td> </tr> <tr> <td>1</td> <td>The MnPWM3 output signal is driven High during fault conditions if the FAULT3 bit in the PWMFAULT (page 358) register is set.</td> </tr> </table>	Value	Description	0	The MnPWM3 output signal is driven Low during fault conditions if the FAULT3 bit in the PWMFAULT (page 358) register is set.	1	The MnPWM3 output signal is driven High during fault conditions if the FAULT3 bit in the PWMFAULT (page 358) register is set.
Value	Description									
0	The MnPWM3 output signal is driven Low during fault conditions if the FAULT3 bit in the PWMFAULT (page 358) register is set.									
1	The MnPWM3 output signal is driven High during fault conditions if the FAULT3 bit in the PWMFAULT (page 358) register is set.									
2	PWM2	RW	0	<p>MnPWM2 Fault Value</p> <table> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>The MnPWM2 output signal is driven Low during fault conditions if the FAULT2 bit in the PWMFAULT (page 358) register is set.</td> </tr> <tr> <td>1</td> <td>The MnPWM2 output signal is driven High during fault conditions if the FAULT2 bit in the PWMFAULT (page 358) register is set.</td> </tr> </table>	Value	Description	0	The MnPWM2 output signal is driven Low during fault conditions if the FAULT2 bit in the PWMFAULT (page 358) register is set.	1	The MnPWM2 output signal is driven High during fault conditions if the FAULT2 bit in the PWMFAULT (page 358) register is set.
Value	Description									
0	The MnPWM2 output signal is driven Low during fault conditions if the FAULT2 bit in the PWMFAULT (page 358) register is set.									
1	The MnPWM2 output signal is driven High during fault conditions if the FAULT2 bit in the PWMFAULT (page 358) register is set.									

Bit/Field	Name	Type	Reset	Description						
1	PWM1	RW	0	<p>MnPWM1 Fault Value</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The MnPWM1 output signal is driven Low during fault conditions if the FAULT1 bit in the PWMFAULT (page 358) register is set.</td> </tr> <tr> <td>1</td> <td>The MnPWM1 output signal is driven High during fault conditions if the FAULT1 bit in the PWMFAULT (page 358) register is set.</td> </tr> </tbody> </table>	Value	Description	0	The MnPWM1 output signal is driven Low during fault conditions if the FAULT1 bit in the PWMFAULT (page 358) register is set.	1	The MnPWM1 output signal is driven High during fault conditions if the FAULT1 bit in the PWMFAULT (page 358) register is set.
Value	Description									
0	The MnPWM1 output signal is driven Low during fault conditions if the FAULT1 bit in the PWMFAULT (page 358) register is set.									
1	The MnPWM1 output signal is driven High during fault conditions if the FAULT1 bit in the PWMFAULT (page 358) register is set.									
0	PWM0	RW	0	<p>MnPWM0 Fault Value</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The MnPWM0 output signal is driven Low during fault conditions if the FAULT0 bit in the PWMFAULT (page 358) register is set.</td> </tr> <tr> <td>1</td> <td>The MnPWM0 output signal is driven High during fault conditions if the FAULT0 bit in the PWMFAULT (page 358) register is set.</td> </tr> </tbody> </table>	Value	Description	0	The MnPWM0 output signal is driven Low during fault conditions if the FAULT0 bit in the PWMFAULT (page 358) register is set.	1	The MnPWM0 output signal is driven High during fault conditions if the FAULT0 bit in the PWMFAULT (page 358) register is set.
Value	Description									
0	The MnPWM0 output signal is driven Low during fault conditions if the FAULT0 bit in the PWMFAULT (page 358) register is set.									
1	The MnPWM0 output signal is driven High during fault conditions if the FAULT0 bit in the PWMFAULT (page 358) register is set.									

23.4.24 PWM Interrupt Enable (PWMINTEN)

This register controls the global interrupt generation capabilities of the PWM module.

Documentation:

- PWM0 base: 0x4002.8000
- PWM1 base: 0x4002.9000
- Offset 0x014
- Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
	reserved														INTFAULT1	INTFAULT0		
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
	reserved														INTPWM3	INTPWM2	INTPWM1	INTPWM0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Bit/Field	Name	Type	Reset	Description						
31:18	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
17	INTFAULT1	RW	0	<p>Interrupt Fault 1</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The fault condition for PWM generator 1 is suppressed and not sent to the interrupt controller.</td> </tr> <tr> <td>1</td> <td>An interrupt is sent to the interrupt controller when the fault condition for PWM generator 1 is asserted.</td> </tr> </tbody> </table>	Value	Description	0	The fault condition for PWM generator 1 is suppressed and not sent to the interrupt controller.	1	An interrupt is sent to the interrupt controller when the fault condition for PWM generator 1 is asserted.
Value	Description									
0	The fault condition for PWM generator 1 is suppressed and not sent to the interrupt controller.									
1	An interrupt is sent to the interrupt controller when the fault condition for PWM generator 1 is asserted.									
16	INTFAULT0	RW	0	<p>Interrupt Fault 0</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The fault condition for PWM generator 0 is suppressed and not sent to the interrupt controller.</td> </tr> <tr> <td>1</td> <td>An interrupt is sent to the interrupt controller when the fault condition for PWM generator 0 is asserted.</td> </tr> </tbody> </table>	Value	Description	0	The fault condition for PWM generator 0 is suppressed and not sent to the interrupt controller.	1	An interrupt is sent to the interrupt controller when the fault condition for PWM generator 0 is asserted.
Value	Description									
0	The fault condition for PWM generator 0 is suppressed and not sent to the interrupt controller.									
1	An interrupt is sent to the interrupt controller when the fault condition for PWM generator 0 is asserted.									
15:4	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						

Bit/Field	Name	Type	Reset	Description						
3	INTPWM3	RW	0	<p>PWM3 Interrupt Enable</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The PWM generator 3 interrupt is suppressed and not sent to the interrupt controller.</td> </tr> <tr> <td>1</td> <td>An interrupt is sent to the interrupt controller when the PWM generator 3 block asserts an interrupt.</td> </tr> </tbody> </table>	Value	Description	0	The PWM generator 3 interrupt is suppressed and not sent to the interrupt controller.	1	An interrupt is sent to the interrupt controller when the PWM generator 3 block asserts an interrupt.
Value	Description									
0	The PWM generator 3 interrupt is suppressed and not sent to the interrupt controller.									
1	An interrupt is sent to the interrupt controller when the PWM generator 3 block asserts an interrupt.									
2	INTPWM2	RW	0	<p>PWM2 Interrupt Enable</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The PWM generator 2 interrupt is suppressed and not sent to the interrupt controller.</td> </tr> <tr> <td>1</td> <td>An interrupt is sent to the interrupt controller when the PWM generator 2 block asserts an interrupt.</td> </tr> </tbody> </table>	Value	Description	0	The PWM generator 2 interrupt is suppressed and not sent to the interrupt controller.	1	An interrupt is sent to the interrupt controller when the PWM generator 2 block asserts an interrupt.
Value	Description									
0	The PWM generator 2 interrupt is suppressed and not sent to the interrupt controller.									
1	An interrupt is sent to the interrupt controller when the PWM generator 2 block asserts an interrupt.									
1	INTPWM1	RW	0	<p>PWM1 Interrupt Enable</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The PWM generator 1 interrupt is suppressed and not sent to the interrupt controller.</td> </tr> <tr> <td>1</td> <td>An interrupt is sent to the interrupt controller when the PWM generator 1 block asserts an interrupt.</td> </tr> </tbody> </table>	Value	Description	0	The PWM generator 1 interrupt is suppressed and not sent to the interrupt controller.	1	An interrupt is sent to the interrupt controller when the PWM generator 1 block asserts an interrupt.
Value	Description									
0	The PWM generator 1 interrupt is suppressed and not sent to the interrupt controller.									
1	An interrupt is sent to the interrupt controller when the PWM generator 1 block asserts an interrupt.									
0	INTPWM0	RW	0	<p>PWM0 Interrupt Enable</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The PWM generator 0 interrupt is suppressed and not sent to the interrupt controller.</td> </tr> <tr> <td>1</td> <td>An interrupt is sent to the interrupt controller when the PWM generator 0 block asserts an interrupt.</td> </tr> </tbody> </table>	Value	Description	0	The PWM generator 0 interrupt is suppressed and not sent to the interrupt controller.	1	An interrupt is sent to the interrupt controller when the PWM generator 0 block asserts an interrupt.
Value	Description									
0	The PWM generator 0 interrupt is suppressed and not sent to the interrupt controller.									
1	An interrupt is sent to the interrupt controller when the PWM generator 0 block asserts an interrupt.									

23.4.25 PWMn Interrupt and Trigger Enable (PWMnINTEN)

This register enables the PWM counter's interrupts and triggers.

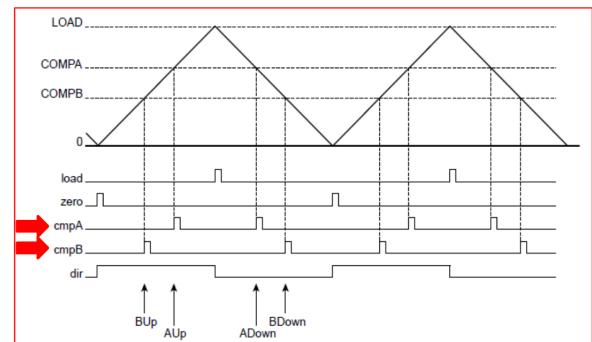
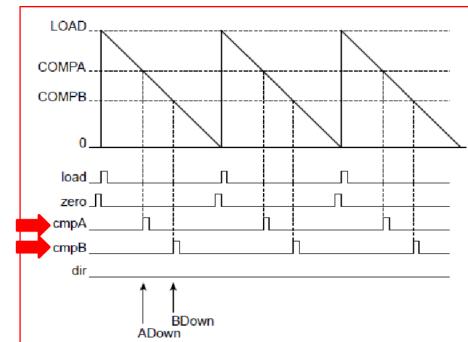
Documentation:

- PWM0 base: 0x4002.8000
- PWM1 base: 0x4002.9000
- Offset 0x044
- Type RW, reset 0x0000.0000

Registers:

- Register 16: PWM0 Interrupt and Trigger Enable (PWM0INTEN), offset 0x044
- Register 17: PWM1 Interrupt and Trigger Enable (PWM1INTEN), offset 0x084
- Register 18: PWM2 Interrupt and Trigger Enable (PWM2INTEN), offset 0x0C4
- Register 19: PWM3 Interrupt and Trigger Enable (PWM3INTEN), offset 0x104

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	reserved															
Type	RO	RO	RW	RW	RW	RW	RW	RW	RO	RO	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



Bit/Field	Name	Type	Reset	Description						
31:14	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
13	TRCMPBD	RW	0	<p>Trigger for Counter=PWMnCMPB (page 334) Down</p> <table border="0"> <tr> <td style="vertical-align: top; padding-right: 10px;">Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>No ADC trigger is output.</td> </tr> <tr> <td>1</td> <td>An ADC trigger pulse is output when the counter matches the value in the PWMnCMPB (page 334) register value while counting down.</td> </tr> </table>	Value	Description	0	No ADC trigger is output.	1	An ADC trigger pulse is output when the counter matches the value in the PWMnCMPB (page 334) register value while counting down.
Value	Description									
0	No ADC trigger is output.									
1	An ADC trigger pulse is output when the counter matches the value in the PWMnCMPB (page 334) register value while counting down.									

Bit/Field	Name	Type	Reset	Description						
12	TRCMPBU	RW	0	<p>Trigger for Counter=PWMnCMPB (page 334) Up</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No ADC trigger is output.</td> </tr> <tr> <td>1</td> <td>An ADC trigger pulse is output when the counter matches the value in the PWMnCMPB (page 334) register value while counting up.</td> </tr> </tbody> </table>	Value	Description	0	No ADC trigger is output.	1	An ADC trigger pulse is output when the counter matches the value in the PWMnCMPB (page 334) register value while counting up.
Value	Description									
0	No ADC trigger is output.									
1	An ADC trigger pulse is output when the counter matches the value in the PWMnCMPB (page 334) register value while counting up.									
11	TRCMPAD	RW	0	<p>Trigger for Counter=PWMnCMPA (page 333) Down</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No ADC trigger is output.</td> </tr> <tr> <td>1</td> <td>An ADC trigger pulse is output when the counter matches the value in the PWMnCMPA (page 333) register value while counting down.</td> </tr> </tbody> </table>	Value	Description	0	No ADC trigger is output.	1	An ADC trigger pulse is output when the counter matches the value in the PWMnCMPA (page 333) register value while counting down.
Value	Description									
0	No ADC trigger is output.									
1	An ADC trigger pulse is output when the counter matches the value in the PWMnCMPA (page 333) register value while counting down.									
10	TRCMPAU	RW	0	<p>Trigger for Counter=PWMnCMPA (page 333) Up</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No ADC trigger is output.</td> </tr> <tr> <td>1</td> <td>An ADC trigger pulse is output when the counter matches the value in the PWMnCMPA (page 333) register value while counting up.</td> </tr> </tbody> </table>	Value	Description	0	No ADC trigger is output.	1	An ADC trigger pulse is output when the counter matches the value in the PWMnCMPA (page 333) register value while counting up.
Value	Description									
0	No ADC trigger is output.									
1	An ADC trigger pulse is output when the counter matches the value in the PWMnCMPA (page 333) register value while counting up.									
9	TRCNTLOAD	RW	0	<p>Trigger for Counter=PWMnLOAD (page 332)</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No ADC trigger is output.</td> </tr> <tr> <td>1</td> <td>An ADC trigger pulse is output when the counter matches the PWMnLOAD (page 332) register.</td> </tr> </tbody> </table>	Value	Description	0	No ADC trigger is output.	1	An ADC trigger pulse is output when the counter matches the PWMnLOAD (page 332) register.
Value	Description									
0	No ADC trigger is output.									
1	An ADC trigger pulse is output when the counter matches the PWMnLOAD (page 332) register.									
8	TRCNTZERO	RW	0	<p>Trigger for Counter=0</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No ADC trigger is output.</td> </tr> <tr> <td>1</td> <td>An ADC trigger pulse is output when the counter is 0.</td> </tr> </tbody> </table>	Value	Description	0	No ADC trigger is output.	1	An ADC trigger pulse is output when the counter is 0.
Value	Description									
0	No ADC trigger is output.									
1	An ADC trigger pulse is output when the counter is 0.									
7:6	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						

Bit/Field	Name	Type	Reset	Description						
5	INTCMPBD	RW	0	<p>Interrupt for Counter=PWMnCMPB (page 334) Down</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>No interrupt.</td></tr> <tr> <td>1</td><td>A raw interrupt occurs when the counter matches the value in the PWMnCMPB (page 334) register value while counting down.</td></tr> </tbody> </table>	Value	Description	0	No interrupt.	1	A raw interrupt occurs when the counter matches the value in the PWMnCMPB (page 334) register value while counting down.
Value	Description									
0	No interrupt.									
1	A raw interrupt occurs when the counter matches the value in the PWMnCMPB (page 334) register value while counting down.									
4	INTCMPBU	RW	0	<p>Interrupt for Counter=PWMnCMPB (page 334) Up</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>No interrupt.</td></tr> <tr> <td>1</td><td>A raw interrupt occurs when the counter matches the value in the PWMnCMPB (page 334) register value while counting up.</td></tr> </tbody> </table>	Value	Description	0	No interrupt.	1	A raw interrupt occurs when the counter matches the value in the PWMnCMPB (page 334) register value while counting up.
Value	Description									
0	No interrupt.									
1	A raw interrupt occurs when the counter matches the value in the PWMnCMPB (page 334) register value while counting up.									
3	INTCMPPAD	RW	0	<p>Interrupt for Counter=PWMnCMPA (page 333) Down</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>No interrupt.</td></tr> <tr> <td>1</td><td>A raw interrupt occurs when the counter matches the value in the PWMnCMPA (page 333) register value while counting down.</td></tr> </tbody> </table>	Value	Description	0	No interrupt.	1	A raw interrupt occurs when the counter matches the value in the PWMnCMPA (page 333) register value while counting down.
Value	Description									
0	No interrupt.									
1	A raw interrupt occurs when the counter matches the value in the PWMnCMPA (page 333) register value while counting down.									
2	INTCMPPAU	RW	0	<p>Interrupt for Counter=PWMnCMPA (page 333) Up</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>No interrupt.</td></tr> <tr> <td>1</td><td>A raw interrupt occurs when the counter matches the value in the PWMnCMPA (page 333) register value while counting up.</td></tr> </tbody> </table>	Value	Description	0	No interrupt.	1	A raw interrupt occurs when the counter matches the value in the PWMnCMPA (page 333) register value while counting up.
Value	Description									
0	No interrupt.									
1	A raw interrupt occurs when the counter matches the value in the PWMnCMPA (page 333) register value while counting up.									
1	INTCNTLOAD	RW	0	<p>Interrupt for Counter=PWMnLOAD (page 332)</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>No interrupt.</td></tr> <tr> <td>1</td><td>A raw interrupt occurs when the counter matches the value in the PWMnLOAD (page 332) register value.</td></tr> </tbody> </table>	Value	Description	0	No interrupt.	1	A raw interrupt occurs when the counter matches the value in the PWMnLOAD (page 332) register value.
Value	Description									
0	No interrupt.									
1	A raw interrupt occurs when the counter matches the value in the PWMnLOAD (page 332) register value.									

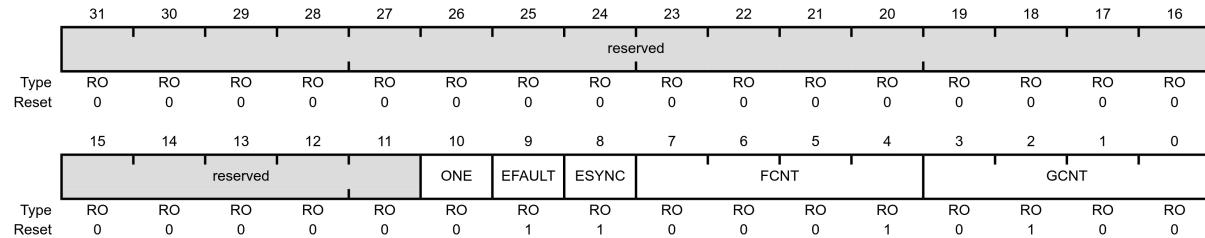
Bit/Field	Name	Type	Reset	Description						
0	INTCNTZERO	RW	0	<p>Interrupt for Counter=0</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No interrupt.</td> </tr> <tr> <td>1</td> <td>A raw interrupt occurs when the counter is zero.</td> </tr> </tbody> </table>	Value	Description	0	No interrupt.	1	A raw interrupt occurs when the counter is zero.
Value	Description									
0	No interrupt.									
1	A raw interrupt occurs when the counter is zero.									

23.4.26 PWM Peripheral Properties (PWMPP)

This register provides information regarding the properties of the PWM module.

Documentation:

- PWM0 base: 0x4002.8000
- PWM1 base: 0x4002.9000
- Offset 0xFC0
- Type RO, reset 0x0000.0314



Bit/Field	Name	Type	Reset	Description														
31:11	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.														
10	ONE	RO	0x0	<p>One-Shot Mode</p> <table> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>One-shot modes are not available.</td> </tr> <tr> <td>1</td> <td>One-shot modes are available.</td> </tr> </table>	Value	Description	0	One-shot modes are not available.	1	One-shot modes are available.								
Value	Description																	
0	One-shot modes are not available.																	
1	One-shot modes are available.																	
9	EFAULT	RO	0x1	<p>Extended Fault</p> <table> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>Extended fault capabilities are not available.</td> </tr> <tr> <td>1</td> <td>Extended fault capabilities are available.</td> </tr> </table>	Value	Description	0	Extended fault capabilities are not available.	1	Extended fault capabilities are available.								
Value	Description																	
0	Extended fault capabilities are not available.																	
1	Extended fault capabilities are available.																	
8	ESYNC	RO	0x1	<p>Extended Synchronization</p> <table> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>Extended synchronization is not available.</td> </tr> <tr> <td>1</td> <td>Extended synchronization is available.</td> </tr> </table>	Value	Description	0	Extended synchronization is not available.	1	Extended synchronization is available.								
Value	Description																	
0	Extended synchronization is not available.																	
1	Extended synchronization is available.																	
7:4	FCNT	RO	0x1	<p>Fault Inputs</p> <table> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0x0</td> <td>No fault inputs.</td> </tr> <tr> <td>0x1</td> <td>1 fault input.</td> </tr> <tr> <td>0x2</td> <td>2 fault inputs.</td> </tr> <tr> <td>0x3</td> <td>3 fault inputs.</td> </tr> <tr> <td>0x4</td> <td>4 fault inputs.</td> </tr> <tr> <td>0x5 - 0xF</td> <td>Reserved</td> </tr> </table>	Value	Description	0x0	No fault inputs.	0x1	1 fault input.	0x2	2 fault inputs.	0x3	3 fault inputs.	0x4	4 fault inputs.	0x5 - 0xF	Reserved
Value	Description																	
0x0	No fault inputs.																	
0x1	1 fault input.																	
0x2	2 fault inputs.																	
0x3	3 fault inputs.																	
0x4	4 fault inputs.																	
0x5 - 0xF	Reserved																	

Bit/Field	Name	Type	Reset	Description														
3:0	GCNT	RO	0x4	<p>Generators</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>No generators.</td> </tr> <tr> <td>0x1</td> <td>1 generator</td> </tr> <tr> <td>0x2</td> <td>2 generators</td> </tr> <tr> <td>0x3</td> <td>3 generators</td> </tr> <tr> <td>0x4</td> <td>4 generators</td> </tr> <tr> <td>0x5 - 0xF</td> <td>Reserved</td> </tr> </tbody> </table> <p>The number of PWM outputs is 2 times the number of PWM generators.</p>	Value	Description	0x0	No generators.	0x1	1 generator	0x2	2 generators	0x3	3 generators	0x4	4 generators	0x5 - 0xF	Reserved
Value	Description																	
0x0	No generators.																	
0x1	1 generator																	
0x2	2 generators																	
0x3	3 generators																	
0x4	4 generators																	
0x5 - 0xF	Reserved																	

23.4.27 PWMn Fault Status 0 (PWMnFLTSTAT0)

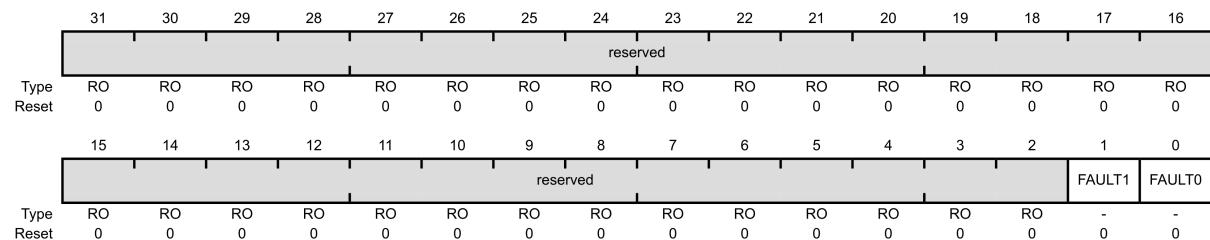
- PWM0 base: 0x4002.8000
- PWM1 base: 0x4002.9000
- Offset 0x804
- Type -, reset 0x0000.0000

Registers:

- Register 78: PWM0 Fault Status 0 (PWM0FLTSTAT0), offset 0x804
- Register 79: PWM1 Fault Status 0 (PWM1FLTSTAT0), offset 0x884
- Register 80: PWM2 Fault Status 0 (PWM2FLTSTAT0), offset 0x904
- Register 81: PWM3 Fault Status 0 (PWM3FLTSTAT0), offset 0x984

Along with the **PWMnFLTSTAT1** (page 373) register, this register provides status regarding the fault condition inputs.

If in unlatched mode, FAULT1 and FAULT0 indicate the status of fault conditions (read-only). If in latched mode, FAULT1 and FAULT0 indicate the status of fault conditions and can be cleared.



Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	FAULT1	-	0	<p>Fault Input 1</p> <p>If the PWMnCTL (page 322) register LATCH bit is clear, this bit is RO and represents the current state of the MnFAULT1 input signal after the logic sense adjustment.</p> <p>If the PWMnCTL (page 322) register LATCH bit is set, this bit is RW1C and represents a sticky version of the MnFAULT1 input signal after the logic sense adjustment.</p> <ul style="list-style-type: none"> • If FAULT1 is set, the input transitioned to the active state previously. • If FAULT1 is clear, the input has not transitioned to the active state since the last time it was cleared. • The FAULT1 bit is cleared by writing it with the value 1.

Bit/Field	Name	Type	Reset	Description
0	FAULT0	-	0	<p>Fault Input 0</p> <p>If the PWMnCTL (page 322) register LATCH bit is clear, this bit is RO and represents the current state of the input signal after the logic sense adjustment.</p> <p>If the PWMnCTL (page 322) register LATCH bit is set, this bit is RW1C and represents a sticky version of the input signal after the logic sense adjustment.</p> <ul style="list-style-type: none"> • If FAULT0 is set, the input transitioned to the active state previously. • If FAULT0 is clear, the input has not transitioned to the active state since the last time it was cleared. • The FAULT0 bit is cleared by writing it with the value 1.

23.4.28 PWMn Fault Status 1 (PWMnFLTSTAT1)

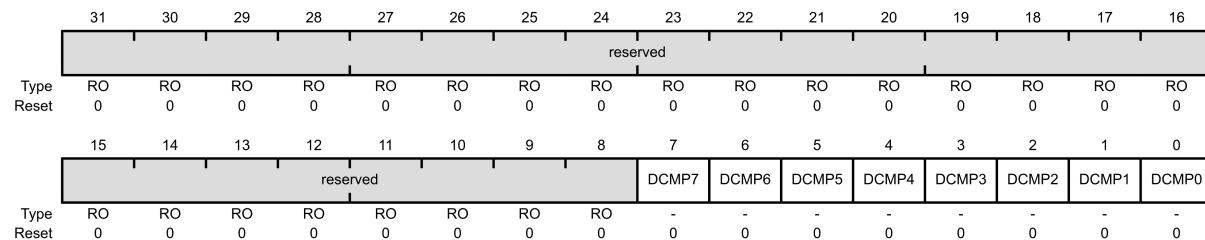
- PWM0 base: 0x4002.8000
- PWM1 base: 0x4002.9000
- Offset 0x808
- Type -, reset 0x0000.0000

Registers:

- Register 82: PWM0 Fault Status 1 (PWM0FLTSTAT1), offset 0x808
- Register 83: PWM1 Fault Status 1 (PWM1FLTSTAT1), offset 0x888
- Register 84: PWM2 Fault Status 1 (PWM2FLTSTAT1), offset 0x908
- Register 85: PWM3 Fault Status 1 (PWM3FLTSTAT1), offset 0x988

Along with the **PWMnFLTSTAT0** (page 371) register, this register provides status regarding the fault condition inputs.

If in unlatched mode, bits 0 to 7 indicate the status of fault conditions (read-only). If in latched mode, bits 0 to 7 indicate the status of fault conditions and can be cleared.



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	DCMP7	-	0	<p>Digital Comparator 7 Trigger</p> <p>If the PWMnCTL (page 322) register LATCH bit is clear, this bit represents the current state of the Digital Comparator 7 trigger input.</p> <p>If the PWMnCTL (page 322) register LATCH bit is set, this bit represents a sticky version of the trigger.</p> <ul style="list-style-type: none"> • If DCMP7 is set, the trigger transitioned to the active state previously. • If DCMP7 is clear, the trigger has not transitioned to the active state since the last time it was cleared. • The DCMP7 bit is cleared by writing it with the value 1.

Bit/Field	Name	Type	Reset	Description
6	DCMP6	-	0	<p>Digital Comparator 6 Trigger</p> <p>If the PWMnCTL (page 322) register LATCH bit is clear, this bit represents the current state of the Digital Comparator 6 trigger input.</p> <p>If the PWMnCTL (page 322) register LATCH bit is set, this bit represents a sticky version of the trigger.</p> <ul style="list-style-type: none"> • If DCMP6 is set, the trigger transitioned to the active state previously. • If DCMP6 is clear, the trigger has not transitioned to the active state since the last time it was cleared. • The DCMP6 bit is cleared by writing it with the value 1.
5	DCMP5	-	0	<p>Digital Comparator 5 Trigger</p> <p>If the PWMnCTL (page 322) register LATCH bit is clear, this bit represents the current state of the Digital Comparator 5 trigger input.</p> <p>If the PWMnCTL (page 322) register LATCH bit is set, this bit represents a sticky version of the trigger.</p> <ul style="list-style-type: none"> • If DCMP5 is set, the trigger transitioned to the active state previously. • If DCMP5 is clear, the trigger has not transitioned to the active state since the last time it was cleared. • The DCMP5 bit is cleared by writing it with the value 1.
4	DCMP4	-	0	<p>Digital Comparator 4 Trigger</p> <p>If the PWMnCTL (page 322) register LATCH bit is clear, this bit represents the current state of the Digital Comparator 4 trigger input.</p> <p>If the PWMnCTL (page 322) register LATCH bit is set, this bit represents a sticky version of the trigger.</p> <ul style="list-style-type: none"> • If DCMP4 is set, the trigger transitioned to the active state previously. • If DCMP4 is clear, the trigger has not transitioned to the active state since the last time it was cleared. • The DCMP4 bit is cleared by writing it with the value 1.

Bit/Field	Name	Type	Reset	Description
3	DCMP3	-	0	<p>Digital Comparator 3 Trigger</p> <p>If the PWMnCTL (page 322) register LATCH bit is clear, this bit represents the current state of the Digital Comparator 3 trigger input.</p> <p>If the PWMnCTL (page 322) register LATCH bit is set, this bit represents a sticky version of the trigger.</p> <ul style="list-style-type: none"> • If DCMP3 is set, the trigger transitioned to the active state previously. • If DCMP3 is clear, the trigger has not transitioned to the active state since the last time it was cleared. • The DCMP3 bit is cleared by writing it with the value 1.
2	DCMP2	-	0	<p>Digital Comparator 2 Trigger</p> <p>If the PWMnCTL (page 322) register LATCH bit is clear, this bit represents the current state of the Digital Comparator 2 trigger input.</p> <p>If the PWMnCTL (page 322) register LATCH bit is set, this bit represents a sticky version of the trigger.</p> <ul style="list-style-type: none"> • If DCMP2 is set, the trigger transitioned to the active state previously. • If DCMP2 is clear, the trigger has not transitioned to the active state since the last time it was cleared. • The DCMP2 bit is cleared by writing it with the value 1.
1	DCMP1	-	0	<p>Digital Comparator 1 Trigger</p> <p>If the PWMnCTL (page 322) register LATCH bit is clear, this bit represents the current state of the Digital Comparator 1 trigger input.</p> <p>If the PWMnCTL (page 322) register LATCH bit is set, this bit represents a sticky version of the trigger.</p> <ul style="list-style-type: none"> • If DCMP1 is set, the trigger transitioned to the active state previously. • If DCMP1 is clear, the trigger has not transitioned to the active state since the last time it was cleared. • The DCMP1 bit is cleared by writing it with the value 1.

Bit/Field	Name	Type	Reset	Description
0	DCMP0	-	0	<p>Digital Comparator 0 Trigger</p> <p>If the PWMnCTL (page 322) register LATCH bit is clear, this bit represents the current state of the Digital Comparator 0 trigger input.</p> <p>If the PWMnCTL (page 322) register LATCH bit is set, this bit represents a sticky version of the trigger.</p> <ul style="list-style-type: none"> • If DCMP0 is set, the trigger transitioned to the active state previously. • If DCMP0 is clear, the trigger has not transitioned to the active state since the last time it was cleared. • The DCMP0 bit is cleared by writing it with the value 1.

23.4.29 PWMn Counter (PWMnCOUNT)

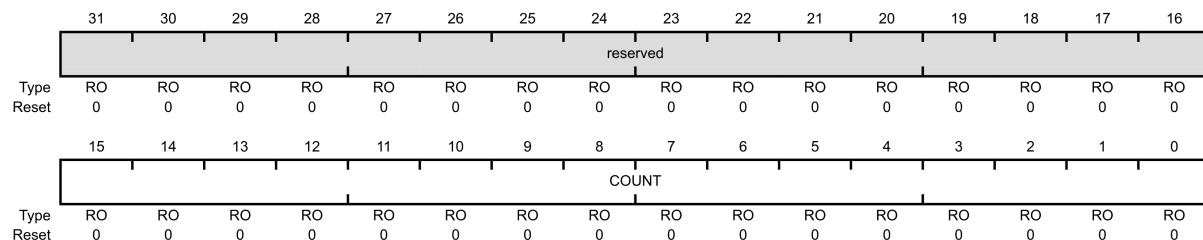
These registers contain the current value of the PWM counter (**PWM0COUNT** (page 376) is the value of the PWM generator 0 block, and so on).

Documentation:

- PWM0 base: 0x4002.8000
- PWM1 base: 0x4002.9000
- Offset 0x054
- Type RO, reset 0x0000.0000

Registers:

- Register 32: PWM0 Counter (PWM0COUNT), offset 0x054
- Register 33: PWM1 Counter (PWM1COUNT), offset 0x094
- Register 34: PWM2 Counter (PWM2COUNT), offset 0x0D4
- Register 35: PWM3 Counter (PWM3COUNT), offset 0x114



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	COUNT	RO	0x0000	<p>Counter Value</p> <p>The current value of the counter.</p>

23.4.30 PWM Raw Interrupt Status (PWMRIS)

This register provides the current set of interrupt sources that are asserted, regardless of whether they are enabled to cause an interrupt to be asserted to the interrupt controller.

Documentation:

- PWM0 base: 0x4002.8000
- PWM1 base: 0x4002.9000
- Offset 0x018
- Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved																
Type	RO	INTFAULT1	INTFAULT0													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
reserved																
Type	RO	INTPWM3	INTPWM2													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	INTPWM1
																INTPWM0

Bit/Field	Name	Type	Reset	Description						
31:18	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
17	INTFAULT1	RO	0	<p>Interrupt Fault PWM 1</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The fault condition for PWM generator 1 has not been asserted.</td> </tr> <tr> <td>1</td> <td>The fault condition for PWM generator 1 is asserted.</td> </tr> </tbody> </table> <p>Note: If the LATCH bit is set in the PWM1CTL (page 322) register, the INTFAULT1 bit in this register can be cleared by writing a 1 to the INTFAULT1 bit in the PWMISC (page 380) register. If the LATCH bit is 0 in the PWM1CTL (page 322) register, writing a 1 to the INTFAULT1 bit in the PWMISC (page 380) register has no effect.</p>	Value	Description	0	The fault condition for PWM generator 1 has not been asserted.	1	The fault condition for PWM generator 1 is asserted.
Value	Description									
0	The fault condition for PWM generator 1 has not been asserted.									
1	The fault condition for PWM generator 1 is asserted.									

Bit/Field	Name	Type	Reset	Description						
16	INTFAULT0	RO	0	<p>Interrupt Fault PWM 0</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The fault condition for PWM generator 0 has not been asserted.</td></tr> <tr> <td>1</td><td>The fault condition for PWM generator 0 is asserted.</td></tr> </tbody> </table> <p>Note: If the LATCH bit is set in the PWM0CTL (page 322) register, the INTFAULT0 bit in this register can be cleared by writing a 1 to the INTFAULT0 bit in the PWMISC (page 380) register. If the LATCH bit is 0 in the PWM0CTL (page 322) register, writing a 1 to the INTFAULT0 bit in the PWMISC (page 380) register has no effect.</p>	Value	Description	0	The fault condition for PWM generator 0 has not been asserted.	1	The fault condition for PWM generator 0 is asserted.
Value	Description									
0	The fault condition for PWM generator 0 has not been asserted.									
1	The fault condition for PWM generator 0 is asserted.									
15:4	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
3	INTPWM3	RO	0	<p>PWM3 Interrupt Asserted</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The PWM generator 3 block interrupt has not been asserted.</td></tr> <tr> <td>1</td><td>The PWM generator 3 block interrupt is asserted.</td></tr> </tbody> </table> <p>The PWM3RIS (page 383) register shows the source of this interrupt. This bit is cleared by writing a 1 to the corresponding bit in the PWM3ISC (page 385) register.</p>	Value	Description	0	The PWM generator 3 block interrupt has not been asserted.	1	The PWM generator 3 block interrupt is asserted.
Value	Description									
0	The PWM generator 3 block interrupt has not been asserted.									
1	The PWM generator 3 block interrupt is asserted.									
2	INTPWM2	RO	0	<p>PWM2 Interrupt Asserted</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The PWM generator 2 block interrupt has not been asserted.</td></tr> <tr> <td>1</td><td>The PWM generator 2 block interrupt is asserted.</td></tr> </tbody> </table> <p>The PWM2RIS (page 383) register shows the source of this interrupt. This bit is cleared by writing a 1 to the corresponding bit in the PWM2ISC (page 385) register.</p>	Value	Description	0	The PWM generator 2 block interrupt has not been asserted.	1	The PWM generator 2 block interrupt is asserted.
Value	Description									
0	The PWM generator 2 block interrupt has not been asserted.									
1	The PWM generator 2 block interrupt is asserted.									

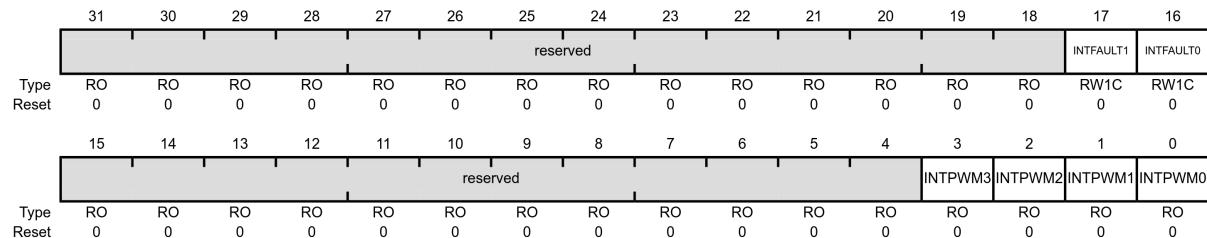
Bit/Field	Name	Type	Reset	Description						
1	INTPWM1	RO	0	<p>PWM1 Interrupt Asserted</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The PWM generator 1 block interrupt has not been asserted.</td> </tr> <tr> <td>1</td> <td>The PWM generator 1 block interrupt is asserted.</td> </tr> </tbody> </table> <p>The PWM1RIS (page 383) register shows the source of this interrupt. This bit is cleared by writing a 1 to the corresponding bit in the PWM1ISC (page 385) register.</p>	Value	Description	0	The PWM generator 1 block interrupt has not been asserted.	1	The PWM generator 1 block interrupt is asserted.
Value	Description									
0	The PWM generator 1 block interrupt has not been asserted.									
1	The PWM generator 1 block interrupt is asserted.									
0	INTPWM0	RO	0	<p>PWM0 Interrupt Asserted</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The PWM generator 0 block interrupt has not been asserted.</td> </tr> <tr> <td>1</td> <td>The PWM generator 0 block interrupt is asserted.</td> </tr> </tbody> </table> <p>The PWM0RIS (page 383) register shows the source of this interrupt. This bit is cleared by writing a 1 to the corresponding bit in the PWM0ISC (page 385) register.</p>	Value	Description	0	The PWM generator 0 block interrupt has not been asserted.	1	The PWM generator 0 block interrupt is asserted.
Value	Description									
0	The PWM generator 0 block interrupt has not been asserted.									
1	The PWM generator 0 block interrupt is asserted.									

23.4.31 PWM Interrupt Status and Clear (PWMISC)

This register provides a summary of the interrupt status of the individual PWM generator blocks. If a fault interrupt is set, the corresponding MnFAULTn input has caused an interrupt.

Documentation:

- PWM0 base: 0x4002.8000
- PWM1 base: 0x4002.9000
- Offset 0x01C
- Type RW1C, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description						
31:18	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
17	INTFAULT1	RW1C	0	<p>FAULT1 Interrupt Asserted</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The fault condition for PWM generator 1 has not been asserted or is not enabled.</td> </tr> <tr> <td>1</td> <td>An enabled interrupt for the fault condition for PWM generator 1 is asserted or is latched.</td> </tr> </tbody> </table> <p>Writing a 1 to this bit clears it and the INTFAULT1 bit in the PWMRIS (page 377) register.</p>	Value	Description	0	The fault condition for PWM generator 1 has not been asserted or is not enabled.	1	An enabled interrupt for the fault condition for PWM generator 1 is asserted or is latched.
Value	Description									
0	The fault condition for PWM generator 1 has not been asserted or is not enabled.									
1	An enabled interrupt for the fault condition for PWM generator 1 is asserted or is latched.									
16	INTFAULT0	RW1C	0	<p>FAULT0 Interrupt Asserted</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The fault condition for PWM generator 0 has not been asserted or is not enabled.</td> </tr> <tr> <td>1</td> <td>An enabled interrupt for the fault condition for PWM generator 0 is asserted or is latched.</td> </tr> </tbody> </table> <p>Writing a 1 to this bit clears it and the INTFAULT0 bit in the PWMRIS (page 377) register.</p>	Value	Description	0	The fault condition for PWM generator 0 has not been asserted or is not enabled.	1	An enabled interrupt for the fault condition for PWM generator 0 is asserted or is latched.
Value	Description									
0	The fault condition for PWM generator 0 has not been asserted or is not enabled.									
1	An enabled interrupt for the fault condition for PWM generator 0 is asserted or is latched.									
15:4	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						

Bit/Field	Name	Type	Reset	Description						
3	INTPWM3	RO	0	<p>PWM3 Interrupt Status</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The PWM generator 3 block interrupt is not asserted or is not enabled.</td> </tr> <tr> <td>1</td> <td>An enabled interrupt for the PWM generator 3 block is asserted.</td> </tr> </tbody> </table> <p>The PWM3RIS (page 383) register shows the source of this interrupt. This bit is cleared by writing a 1 to the corresponding bit in the PWM3ISC (page 385) register.</p>	Value	Description	0	The PWM generator 3 block interrupt is not asserted or is not enabled.	1	An enabled interrupt for the PWM generator 3 block is asserted.
Value	Description									
0	The PWM generator 3 block interrupt is not asserted or is not enabled.									
1	An enabled interrupt for the PWM generator 3 block is asserted.									
2	INTPWM2	RO	0	<p>PWM2 Interrupt Status</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The PWM generator 2 block interrupt is not asserted or is not enabled.</td> </tr> <tr> <td>1</td> <td>An enabled interrupt for the PWM generator 2 block is asserted.</td> </tr> </tbody> </table> <p>The PWM2RIS (page 383) register shows the source of this interrupt. This bit is cleared by writing a 1 to the corresponding bit in the PWM2ISC (page 385) register.</p>	Value	Description	0	The PWM generator 2 block interrupt is not asserted or is not enabled.	1	An enabled interrupt for the PWM generator 2 block is asserted.
Value	Description									
0	The PWM generator 2 block interrupt is not asserted or is not enabled.									
1	An enabled interrupt for the PWM generator 2 block is asserted.									
1	INTPWM1	RO	0	<p>PWM1 Interrupt Status</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The PWM generator 1 block interrupt is not asserted or is not enabled.</td> </tr> <tr> <td>1</td> <td>An enabled interrupt for the PWM generator 1 block is asserted.</td> </tr> </tbody> </table> <p>The PWM1RIS (page 383) register shows the source of this interrupt. This bit is cleared by writing a 1 to the corresponding bit in the PWM1ISC (page 385) register.</p>	Value	Description	0	The PWM generator 1 block interrupt is not asserted or is not enabled.	1	An enabled interrupt for the PWM generator 1 block is asserted.
Value	Description									
0	The PWM generator 1 block interrupt is not asserted or is not enabled.									
1	An enabled interrupt for the PWM generator 1 block is asserted.									

Bit/Field	Name	Type	Reset	Description						
0	INTPWM0	RO	0	<p>PWM0 Interrupt Status</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The PWM generator 0 block interrupt is not asserted or is not enabled.</td> </tr> <tr> <td>1</td> <td>An enabled interrupt for the PWM generator 0 block is asserted.</td> </tr> </tbody> </table> <p>The PWM0RIS (page 383) register shows the source of this interrupt. This bit is cleared by writing a 1 to the corresponding bit in the PWM0ISC (page 385) register.</p>	Value	Description	0	The PWM generator 0 block interrupt is not asserted or is not enabled.	1	An enabled interrupt for the PWM generator 0 block is asserted.
Value	Description									
0	The PWM generator 0 block interrupt is not asserted or is not enabled.									
1	An enabled interrupt for the PWM generator 0 block is asserted.									

23.4.32 PWMn Raw Interrupt Status (PWMnRIS)

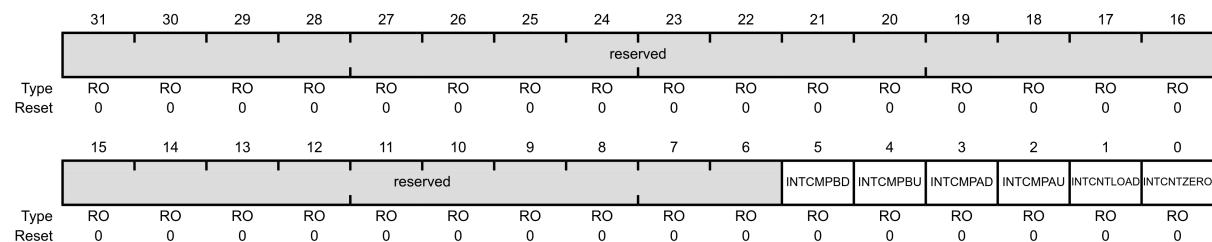
These registers provide the current set of interrupt sources that are asserted, regardless of whether they cause an interrupt to be asserted to the controller (**PWM0RIS** (page 383) controls the PWM generator 0 block, and so on).

Documentation:

- PWM0 base: 0x4002.8000
- PWM1 base: 0x4002.9000
- Offset 0x048
- Type RO, reset 0x0000.0000

Registers:

- Register 20: PWM0 Raw Interrupt Status (PWM0RIS), offset 0x048
- Register 21: PWM1 Raw Interrupt Status (PWM1RIS), offset 0x088
- Register 22: PWM2 Raw Interrupt Status (PWM2RIS), offset 0x0C8
- Register 23: PWM3 Raw Interrupt Status (PWM3RIS), offset 0x108



Bit/Field	Name	Type	Reset	Description						
31:6	reserved	RO	0	<p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p>						
5	INTCMPBD	RO	0	<p>Comparator B Down Interrupt Status</p> <table> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>An interrupt has not occurred.</td> </tr> <tr> <td>1</td> <td>The counter has matched the value in the PWMnCMPB (page 334) register while counting down.</td> </tr> </table> <p>This bit is cleared by writing a 1 to the INTCMPBD bit in the PWMnISC (page 385) register.</p>	Value	Description	0	An interrupt has not occurred.	1	The counter has matched the value in the PWMnCMPB (page 334) register while counting down.
Value	Description									
0	An interrupt has not occurred.									
1	The counter has matched the value in the PWMnCMPB (page 334) register while counting down.									
4	INTCMPBU	RO	0	<p>Comparator B Up Interrupt Status</p> <table> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>An interrupt has not occurred.</td> </tr> <tr> <td>1</td> <td>The counter has matched the value in the PWMnCMPB (page 334) register while counting up.</td> </tr> </table> <p>This bit is cleared by writing a 1 to the INTCMPBU bit in the PWMnISC (page 385) register.</p>	Value	Description	0	An interrupt has not occurred.	1	The counter has matched the value in the PWMnCMPB (page 334) register while counting up.
Value	Description									
0	An interrupt has not occurred.									
1	The counter has matched the value in the PWMnCMPB (page 334) register while counting up.									

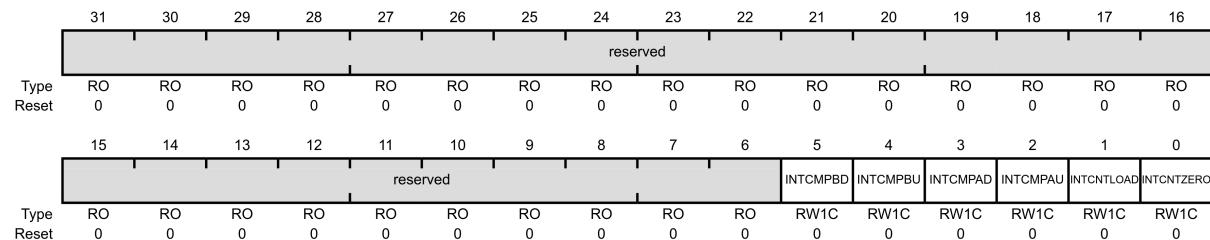
Bit/Field	Name	Type	Reset	Description						
3	INTCMPAD	RO	0	<p>Comparator A Down Interrupt Status</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>An interrupt has not occurred.</td></tr> <tr> <td>1</td><td>The counter has matched the value in the PWMnCMPA (page 333) register while counting down.</td></tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the INTCPAD bit in the PWMnISC (page 385) register.</p>	Value	Description	0	An interrupt has not occurred.	1	The counter has matched the value in the PWMnCMPA (page 333) register while counting down.
Value	Description									
0	An interrupt has not occurred.									
1	The counter has matched the value in the PWMnCMPA (page 333) register while counting down.									
2	INTCMPAU	RO	0	<p>Comparator A Up Interrupt Status</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>An interrupt has not occurred.</td></tr> <tr> <td>1</td><td>The counter has matched the value in the PWMnCMPA (page 333) register while counting up.</td></tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the INTCPAU bit in the PWMnISC (page 385) register.</p>	Value	Description	0	An interrupt has not occurred.	1	The counter has matched the value in the PWMnCMPA (page 333) register while counting up.
Value	Description									
0	An interrupt has not occurred.									
1	The counter has matched the value in the PWMnCMPA (page 333) register while counting up.									
1	INTCNTLOAD	RO	0	<p>Counter=Load Interrupt Status</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>An interrupt has not occurred.</td></tr> <tr> <td>1</td><td>The counter has matched the value in the PWMnLOAD (page 332) register.</td></tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the INTCNTLOAD bit in the PWMnISC (page 385) register.</p>	Value	Description	0	An interrupt has not occurred.	1	The counter has matched the value in the PWMnLOAD (page 332) register.
Value	Description									
0	An interrupt has not occurred.									
1	The counter has matched the value in the PWMnLOAD (page 332) register.									
0	INTCNTZERO	RO	0	<p>Counter=0 Interrupt Status</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>An interrupt has not occurred.</td></tr> <tr> <td>1</td><td>The counter has matched zero.</td></tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the INTCNTZERO bit in the PWMnISC (page 385) register.</p>	Value	Description	0	An interrupt has not occurred.	1	The counter has matched zero.
Value	Description									
0	An interrupt has not occurred.									
1	The counter has matched zero.									

23.4.33 PWMn Interrupt Status and Clear (PWMnISC)

These registers provide the current set of interrupt sources that are asserted to the interrupt controller (**PWM0ISC** (page 385) controls the PWM generator 0 block, and so on).

Documentation:

- PWM0 base: 0x4002.8000
- PWM1 base: 0x4002.9000
- Offset 0x04C
- Type RW1C, reset 0x0000.0000



Registers:

- Register 24: PWM0 Interrupt Status and Clear (PWM0ISC), offset 0x04C
- Register 25: PWM1 Interrupt Status and Clear (PWM1ISC), offset 0x08C
- Register 26: PWM2 Interrupt Status and Clear (PWM2ISC), offset 0x0CC
- Register 27: PWM3 Interrupt Status and Clear (PWM3ISC), offset 0x10C

Bit/Field	Name	Type	Reset	Description						
31:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
5	INTCMPBD	RW1C	0	<p>Comparator B Down Interrupt</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No interrupt has occurred or the interrupt is masked.</td> </tr> <tr> <td>1</td> <td>The INTCMPBD bits in the PWMnRIS (page 383) and PWMnINTEN (page 365) registers are set, providing an interrupt to the interrupt controller.</td> </tr> </tbody> </table> <p>This bit is cleared by writing a 1. Clearing this bit also clears the INTCMPBD bit in the PWMnRIS (page 383) register.</p>	Value	Description	0	No interrupt has occurred or the interrupt is masked.	1	The INTCMPBD bits in the PWMnRIS (page 383) and PWMnINTEN (page 365) registers are set, providing an interrupt to the interrupt controller.
Value	Description									
0	No interrupt has occurred or the interrupt is masked.									
1	The INTCMPBD bits in the PWMnRIS (page 383) and PWMnINTEN (page 365) registers are set, providing an interrupt to the interrupt controller.									

Bit/Field	Name	Type	Reset	Description						
4	INTCMPBU	RW1C	0	<p>Comparator B Up Interrupt</p> <table> <thead> <tr> <th data-bbox="759 309 822 345">Value</th> <th data-bbox="822 309 1013 345">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="759 345 822 428">0</td> <td data-bbox="822 345 1387 428">No interrupt has occurred or the interrupt is masked.</td> </tr> <tr> <td data-bbox="759 428 822 601">1</td> <td data-bbox="822 428 1387 601">The INTCMPBU bits in the PWMnRIS (page 383) and PWMnINTEN (page 365) registers are set, providing an interrupt to the interrupt controller.</td> </tr> </tbody> </table> <p>This bit is cleared by writing a 1. Clearing this bit also clears the INTCMPBU bit in the PWMnRIS (page 383) register.</p>	Value	Description	0	No interrupt has occurred or the interrupt is masked.	1	The INTCMPBU bits in the PWMnRIS (page 383) and PWMnINTEN (page 365) registers are set, providing an interrupt to the interrupt controller.
Value	Description									
0	No interrupt has occurred or the interrupt is masked.									
1	The INTCMPBU bits in the PWMnRIS (page 383) and PWMnINTEN (page 365) registers are set, providing an interrupt to the interrupt controller.									
3	INTCMPAD	RW1C	0	<p>Comparator A Down Interrupt</p> <table> <thead> <tr> <th data-bbox="759 819 822 855">Value</th> <th data-bbox="822 819 1013 855">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="759 855 822 938">0</td> <td data-bbox="822 855 1387 938">No interrupt has occurred or the interrupt is masked.</td> </tr> <tr> <td data-bbox="759 938 822 1111">1</td> <td data-bbox="822 938 1387 1111">The INTCMPAD bits in the PWMnRIS (page 383) and PWMnINTEN (page 365) registers are set, providing an interrupt to the interrupt controller.</td> </tr> </tbody> </table> <p>This bit is cleared by writing a 1. Clearing this bit also clears the INTCMPAD bit in the PWMnRIS (page 383) register.</p>	Value	Description	0	No interrupt has occurred or the interrupt is masked.	1	The INTCMPAD bits in the PWMnRIS (page 383) and PWMnINTEN (page 365) registers are set, providing an interrupt to the interrupt controller.
Value	Description									
0	No interrupt has occurred or the interrupt is masked.									
1	The INTCMPAD bits in the PWMnRIS (page 383) and PWMnINTEN (page 365) registers are set, providing an interrupt to the interrupt controller.									
2	INTCMPAU	RW1C	0	<p>Comparator A Up Interrupt</p> <table> <thead> <tr> <th data-bbox="759 1329 822 1365">Value</th> <th data-bbox="822 1329 1013 1365">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="759 1365 822 1448">0</td> <td data-bbox="822 1365 1387 1448">No interrupt has occurred or the interrupt is masked.</td> </tr> <tr> <td data-bbox="759 1448 822 1621">1</td> <td data-bbox="822 1448 1387 1621">The INTCMPAU bits in the PWMnRIS (page 383) and PWMnINTEN (page 365) registers are set, providing an interrupt to the interrupt controller.</td> </tr> </tbody> </table> <p>This bit is cleared by writing a 1. Clearing this bit also clears the INTCMPAU bit in the PWMnRIS (page 383) register.</p>	Value	Description	0	No interrupt has occurred or the interrupt is masked.	1	The INTCMPAU bits in the PWMnRIS (page 383) and PWMnINTEN (page 365) registers are set, providing an interrupt to the interrupt controller.
Value	Description									
0	No interrupt has occurred or the interrupt is masked.									
1	The INTCMPAU bits in the PWMnRIS (page 383) and PWMnINTEN (page 365) registers are set, providing an interrupt to the interrupt controller.									

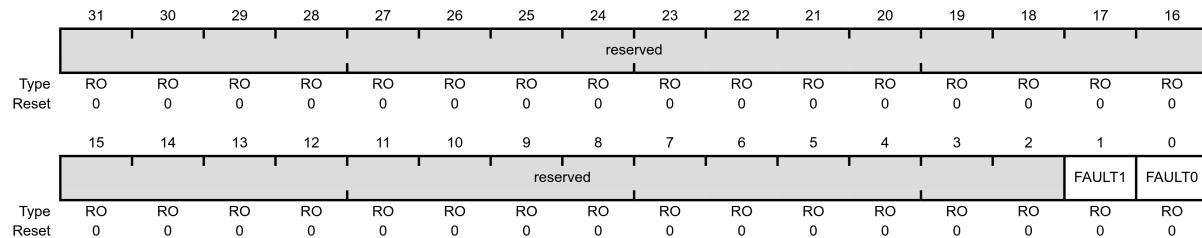
Bit/Field	Name	Type	Reset	Description						
1	INTCNTLOAD	RW1C	0	<p>Counter=Load Interrupt</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No interrupt has occurred or the interrupt is masked.</td> </tr> <tr> <td>1</td> <td>The INTCNTLOAD bits in the PWMnRIS (page 383) and PWMnINTEN (page 365) registers are set, providing an interrupt to the interrupt controller.</td> </tr> </tbody> </table> <p>This bit is cleared by writing a 1. Clearing this bit also clears the INTCNTLOAD bit in the PWMnRIS (page 383) register.</p>	Value	Description	0	No interrupt has occurred or the interrupt is masked.	1	The INTCNTLOAD bits in the PWMnRIS (page 383) and PWMnINTEN (page 365) registers are set, providing an interrupt to the interrupt controller.
Value	Description									
0	No interrupt has occurred or the interrupt is masked.									
1	The INTCNTLOAD bits in the PWMnRIS (page 383) and PWMnINTEN (page 365) registers are set, providing an interrupt to the interrupt controller.									
0	INTCNTZERO	RW1C	0	<p>Counter=0 Interrupt</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No interrupt has occurred or the interrupt is masked.</td> </tr> <tr> <td>1</td> <td>The INTCNTZERO bits in the PWMnRIS (page 383) and PWMnINTEN (page 365) registers are set, providing an interrupt to the interrupt controller.</td> </tr> </tbody> </table> <p>This bit is cleared by writing a 1. Clearing this bit also clears the INTCNTZERO bit in the PWMnRIS (page 383) register.</p>	Value	Description	0	No interrupt has occurred or the interrupt is masked.	1	The INTCNTZERO bits in the PWMnRIS (page 383) and PWMnINTEN (page 365) registers are set, providing an interrupt to the interrupt controller.
Value	Description									
0	No interrupt has occurred or the interrupt is masked.									
1	The INTCNTZERO bits in the PWMnRIS (page 383) and PWMnINTEN (page 365) registers are set, providing an interrupt to the interrupt controller.									

23.4.34 PWM Status (PWMSTATUS)

This register provides the unlatched status of the PWM generator fault condition.

Documentation:

- PWM0 base: 0x4002.8000
- PWM1 base: 0x4002.9000
- Offset 0x020
- Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description						
31:2	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
1	FAULT1	RO	0	<p>Generator 1 Fault Status</p> <table> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>The fault condition for PWM generator 1 is not asserted.</td> </tr> <tr> <td>1</td> <td>The fault condition for PWM generator 1 is asserted. If the FLTSRC bit in the PWM1CTL (page 322) register is clear, the input is the source of the fault condition, and is therefore asserted.</td> </tr> </table>	Value	Description	0	The fault condition for PWM generator 1 is not asserted.	1	The fault condition for PWM generator 1 is asserted. If the FLTSRC bit in the PWM1CTL (page 322) register is clear, the input is the source of the fault condition, and is therefore asserted.
Value	Description									
0	The fault condition for PWM generator 1 is not asserted.									
1	The fault condition for PWM generator 1 is asserted. If the FLTSRC bit in the PWM1CTL (page 322) register is clear, the input is the source of the fault condition, and is therefore asserted.									
0	FAULT0	RO	0	<p>Generator 0 Fault Status</p> <table> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>The fault condition for PWM generator 0 is not asserted.</td> </tr> <tr> <td>1</td> <td>The fault condition for PWM generator 0 is asserted. If the FLTSRC bit in the PWM0CTL (page 322) register is clear, the input is the source of the fault condition, and is therefore asserted.</td> </tr> </table>	Value	Description	0	The fault condition for PWM generator 0 is not asserted.	1	The fault condition for PWM generator 0 is asserted. If the FLTSRC bit in the PWM0CTL (page 322) register is clear, the input is the source of the fault condition, and is therefore asserted.
Value	Description									
0	The fault condition for PWM generator 0 is not asserted.									
1	The fault condition for PWM generator 0 is asserted. If the FLTSRC bit in the PWM0CTL (page 322) register is clear, the input is the source of the fault condition, and is therefore asserted.									

23.5 PWM module clock source

Each of the Cortex M4's two PWM module has two clock sources:

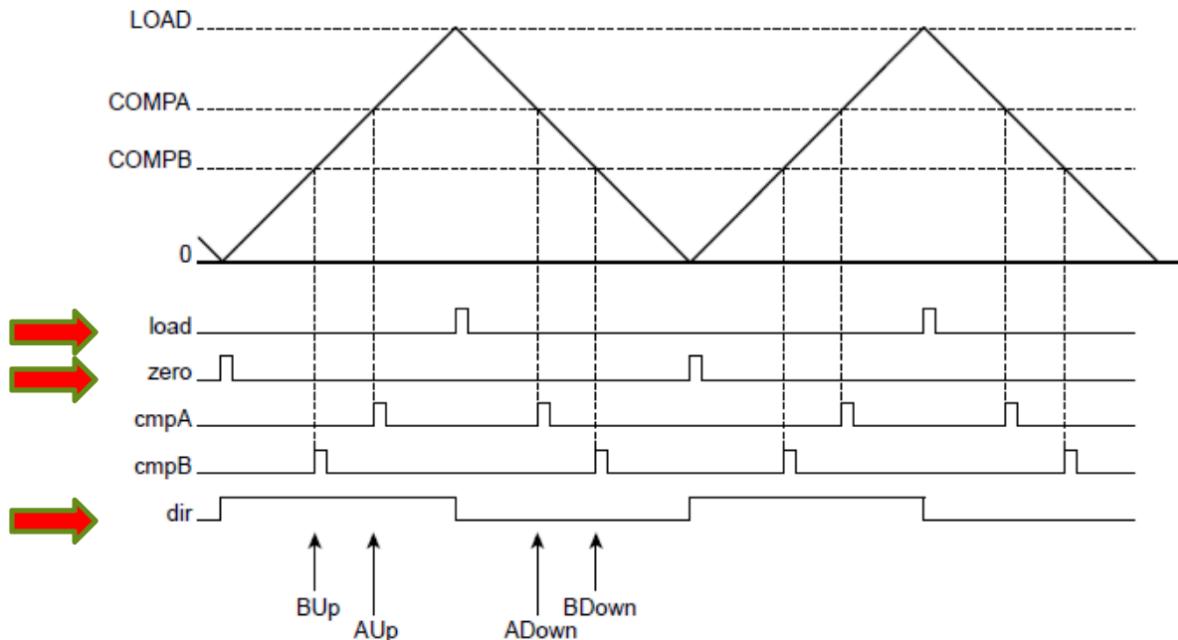
- System clock (TM4C123GH6PM's system clock is 80 MHz)
- PWM clock, which is a pre-divided system clock ($\frac{80 \text{ MHz}}{64}$ (default))

The clock source is selected by programming the USEPWMDIV bit, bit 20, in the **Run-Mode Clock Configuration (RCC)** (page 310) register at system control offset 0x060.

The PWMDIV bit field specifies the divisor of the system clock that is used to create the PWM clock.

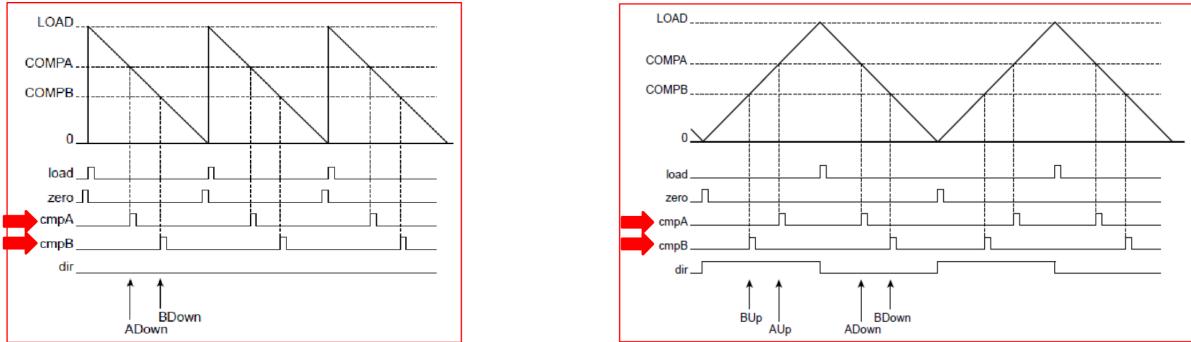
23.6 PWM generator timer

- Count-down mode: The timer starts from a digital value (called the load value) and decrements continuously to zero. Then, it is reset to the load value and is decremented continuously to zero again.
- Count-up/down mode: The timer starts from zero and increments continuously to a digital value (called the load value), and then decrements continuously to zero, and the process repeats.
- There are 3 different types of pulses, zero pulses, direction pulses and load pulses.



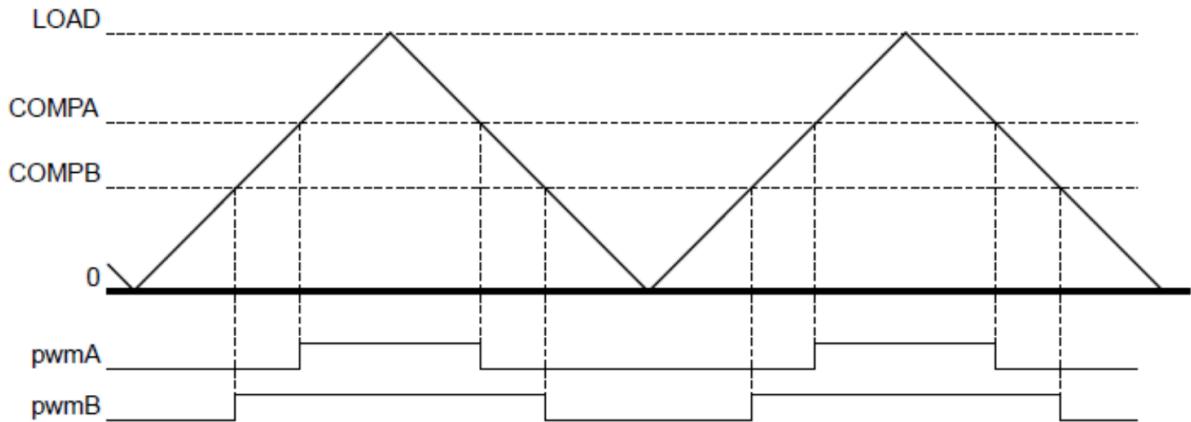
23.7 PWM generator comparators

- Each PWM generator has two digital comparators.
- One digital comparator holds the match value COMPA.
- Another digital comparator holds the match value COMPB.
- When the timer's count reaches one of these two match values, a high pulse is produced.



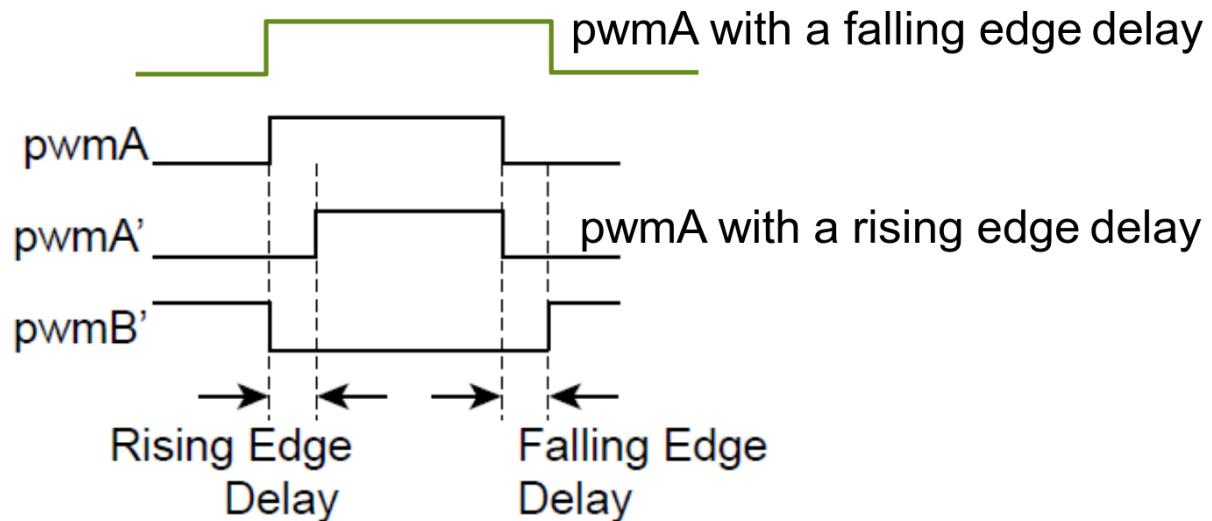
23.8 PWM signal generator

- The PWM generator has 5 internal pulses: load pulses, zero pulses, direction pulses, COMPA pulses, and COMPB pulses.
- The signal generator takes these pulses and produces two PWM waves: pwmA and pwmB.



23.9 PWM dead-band generator

- The dead-band generator gets inputs pwmA and pwmB from the signal generator and produces outputs pwmA' and pwmB'.
- pwmA' is the version of pwmA with a rising edge delay.
- pwmB' is the inverted version of pwmA with a falling edge delay.

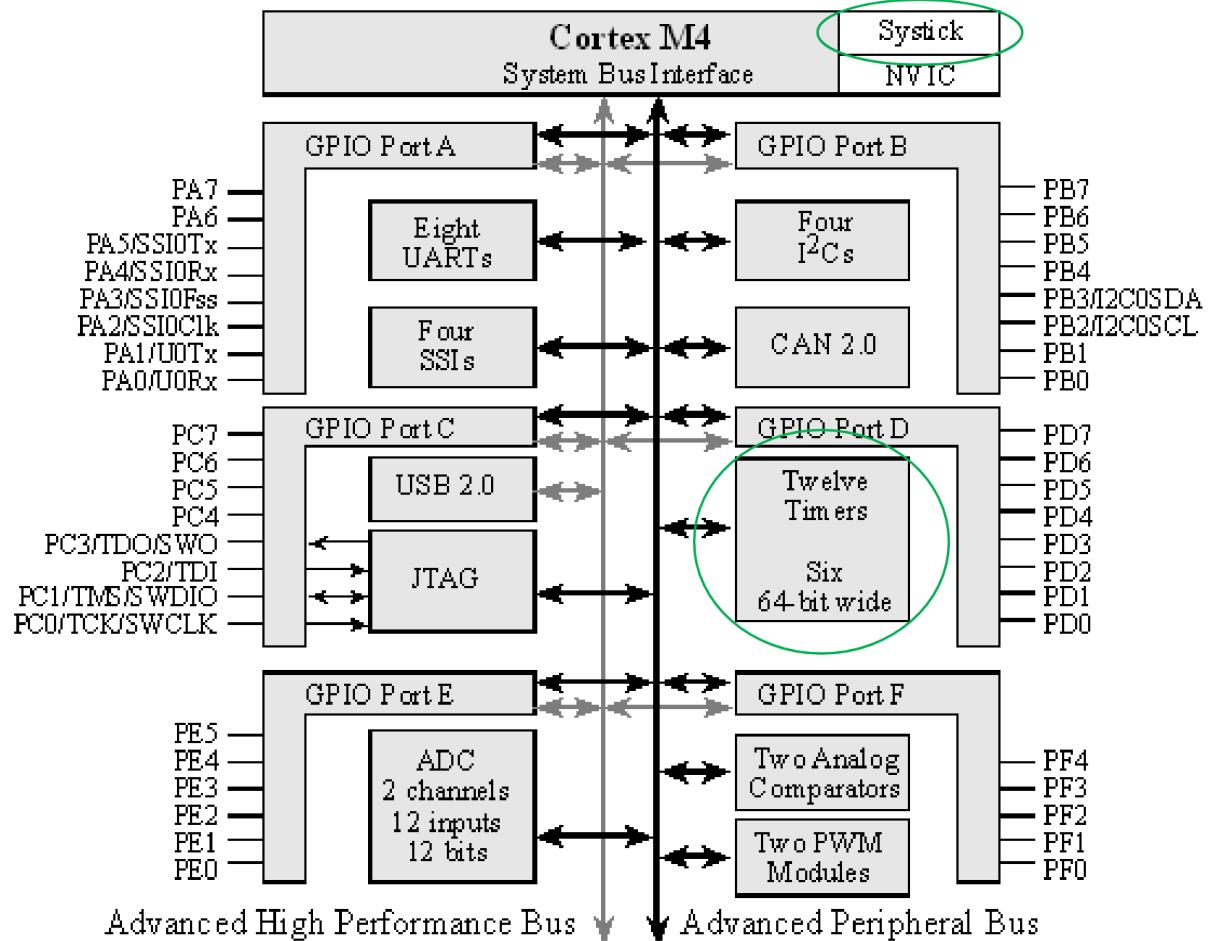


23.10 Initialisation and configuration

Initialise PWM generator 0 with a 25 kHz frequency, a 25% duty cycle on the MnPWM0 pin, and a 75% duty cycle on the MnPWM1 pin. The system clock is 20 MHz.

1. Enable the PWM clock by writing a value of 0x0010.0000 to the **RCGC0** ([page 317](#)) register in the system control module.
2. Enable the clock to the appropriate module via the **RCGC2** ([page 320](#)) register in the system control module.
3. In the GPIO module, enable the appropriate pins for their alternate function using the **GPIOAFSEL** ([page 148](#)) register. To determine which GPIOs to configure, see [page 305](#).
4. Configure the PMCn fields in the **GPIOPCTL** ([page 158](#)) register to assign the PWM signals to the appropriate pins. See [page 130](#) and [page 158](#).
5. Configure the **Run-Mode Clock Configuration (RCC)** ([page 310](#)) register in the system control module to use the PWM divide (USEPWMDIV) and set the divider (PWMDIV) to divide by 2 (0x00).
6. Configure the PWM generator for count-down mode with immediate updates to the parameters.
 - Write 0x0000.0000 (reset) to the **PWM0CTL** ([page 322](#)) register.
 - Write 0x0000.008C to the **PWM0GENA** ([page 328](#)) register.
 - Write 0x0000.080C to the **PWM0GENB** ([page 330](#)) register.
7. Set the period.
 - For a 25 kHz frequency, each period is $\frac{1}{25,000}$ or 40 μ s.
 - The PWM clock source is 10 MHz, which is the system clock, 20 MHz, divided by 2. 10 MHz is 10 clock ticks, or 1 μ s. Hence, 40 μ s is 400 clock ticks.
 - Use this value to set the **PWM0LOAD** ([page 332](#)) register. In count-down mode, set the LOAD field in the **PWM0LOAD** ([page 332](#)) register to the requested period minus 1.
 - Hence, write 0x0000.018F (399) to the **PWM0LOAD** ([page 332](#)) register.
8. Set the pulse width of the MnPWM0 pin for a 25% duty cycle.
 - Write 0x0000.012B (299) to the **PWM0CMPA** ([page 333](#)) register.
9. Set the pulse width of the MnPWM1 pin for a 75% duty cycle.
 - Write 0x0000.0063 (99) to the **PWM0CMPB** ([page 334](#)) register.
10. Start the timers in PWM generator 0.
 - Write 0x0000.0001 (enable bit 0) to the **PWM0CTL** ([page 322](#)) register.
11. Enable PWM outputs.
 - Write 0x0000.0003 to the **PWMENABLE** ([page 335](#)) register.

24 TM4C123GH6PM timers and counters



24.1 System timer (SysTick)

The Cortex-m4 has an integrated 24-bit system timer (SysTick) running on the bus clock frequency.

SysTick features:

- 24-bit clear-on write
- Decrementing
- Wrap-on-zero counter with flexible control mechanism

The SysTick can be used as a **timer** in several different ways:

- A **real time operating system (RTOS) tick timer** that fires at a programmable rate, like at 100 Hz for example, and invokes a SysTick routine.
- A **high-speed alarm timer** using the system clock.
- A **variable rate alarm or signal timer**. The duration is range-dependent on the reference clock used and the dynamic range of the counter.

The SysTick can be used as a **counter** in the following ways:

- A **simple counter** used to measure time to completion and time taken.
- An **internal clock source** control based on missing or meeting durations.
- The COUNT bit in the **STCTRL** (page 395) control and status register can be used to determine if an action completed with a set duration.

24.1.1 Register map

Base: 0xE000.E000

Offset	Name	Type	Reset	Description
System Timer (SysTick) Registers				
0x010	STCTRL	RW	0x0000.0004	SysTick Control and Status Register
0x014	STRELOAD	RW	-	SysTick Reload Value Register
0x018	STCURRENT	RWC	-	SysTick Current Value Register

24.1.2 Registers

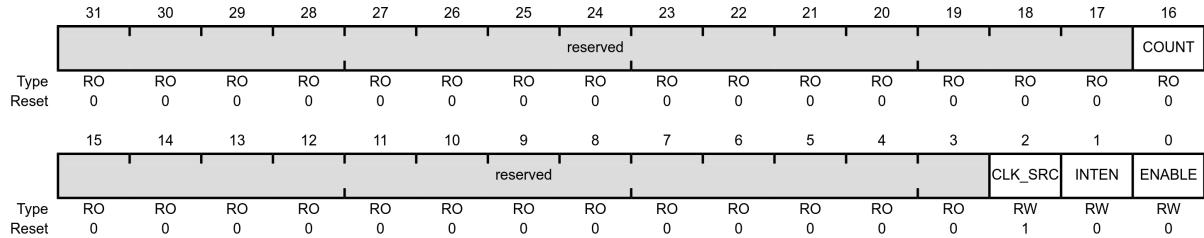
The timer consists of 3 registers:

1. SysTick Control and Status (STCTRL) ([page 395](#))
 - Control the status counter to configure its clock.
 - Enable the counter.
 - Enable the SysTick interrupt.
 - Determine the counter status.
2. SysTick Reload Value (STRELOAD) ([page 397](#))
 - Reload the value for the counter.
 - Provide the counter's wrap value.
3. SysTick Current Value (STCURRENT) ([page 398](#))
 - The current value of the counter.

24.1.2.1 SysTick Control and Status Register (STCTRL)

Note: This register can only be accessed from privileged mode.

- Base 0xE000.E000
- Offset 0x010
- Type RW, reset 0x0000.0004



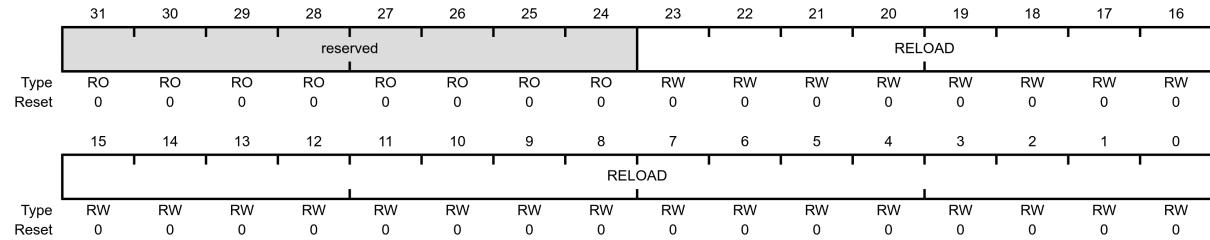
Bit/Field	Name	Type	Reset	Description
31:17	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	COUNT	RO	0	<p>Count Flag</p> <p>Value Description</p> <p>0 The SysTick timer has not counted to 0 since the last time this bit was read.</p> <p>1 The SysTick timer has counted to 0 since the last time this bit was read.</p> <p>This bit is cleared by a read of the register or if the STCURRENT (page 398) register is written with any value.</p> <p>If read by the debugger using the DAP, this bit is cleared only if the MasterType bit in the AHB-AP Control Register is clear. Otherwise, the COUNT bit is not changed by the debugger read. See the <i>ARM® Debug Interface V5 Architecture Specification</i> for more information on MasterType.</p>
15:3	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	CLK_SRC	RW	1	<p>Clock Source</p> <p>Value Description</p> <p>0 Precision internal oscillator (PIOSC) divided by 4</p> <p>1 System clock</p>

Bit/Field	Name	Type	Reset	Description						
1	INTEN	RW	0	<p>Interrupt Enable</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Interrupt generation is disabled. Software can use the COUNT bit to determine if the counter has ever reached 0.</td> </tr> <tr> <td>1</td> <td>An interrupt is generated to the NVIC when SysTick counts to 0.</td> </tr> </tbody> </table>	Value	Description	0	Interrupt generation is disabled. Software can use the COUNT bit to determine if the counter has ever reached 0.	1	An interrupt is generated to the NVIC when SysTick counts to 0.
Value	Description									
0	Interrupt generation is disabled. Software can use the COUNT bit to determine if the counter has ever reached 0.									
1	An interrupt is generated to the NVIC when SysTick counts to 0.									
0	ENABLE	RW	0	<p>Enable</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The counter is disabled.</td> </tr> <tr> <td>1</td> <td>Enables SysTick to operate in a multi-shot way. That is, the counter loads the RELOAD value and begins counting down. On reaching 0, the COUNT bit is set and an interrupt is generated if enabled by INTEN. The counter then loads the RELOAD value again and begins counting.</td> </tr> </tbody> </table>	Value	Description	0	The counter is disabled.	1	Enables SysTick to operate in a multi-shot way. That is, the counter loads the RELOAD value and begins counting down. On reaching 0, the COUNT bit is set and an interrupt is generated if enabled by INTEN. The counter then loads the RELOAD value again and begins counting.
Value	Description									
0	The counter is disabled.									
1	Enables SysTick to operate in a multi-shot way. That is, the counter loads the RELOAD value and begins counting down. On reaching 0, the COUNT bit is set and an interrupt is generated if enabled by INTEN. The counter then loads the RELOAD value again and begins counting.									

24.1.2.2 SysTick Reload Value Register (STRELOAD)

Note: This register can only be accessed from privileged mode.

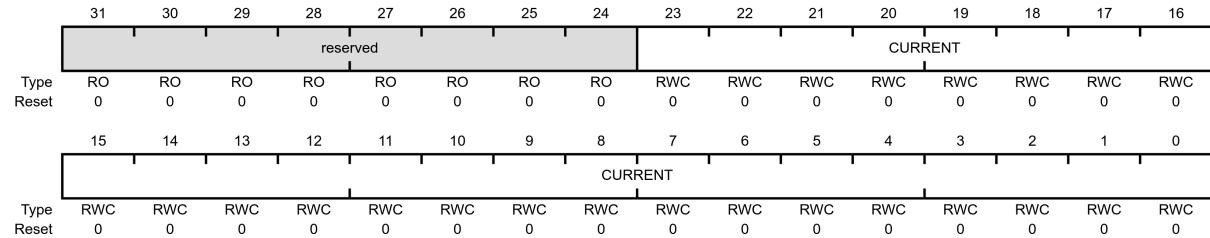
- Base 0xE000.E000
- Offset 0x014
- Type RW, reset -



Bit/Field	Name	Type	Reset	Description
31:24	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23:0	RELOAD	RW	0x00.0000	Reload Value Value to load into the SysTick Current Value (STCURRENT) (page 398) register when the counter reaches 0.

24.1.2.3 SysTick Current Value Register (STCURRENT)

- Base 0xE000.E000
- Offset 0x018
- Type RWC, reset -



Bit/Field	Name	Type	Reset	Description
31:24	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23:0	CURRENT	RWC	0x00.0000	<p>Current Value</p> <p>This field contains the current value at the time the register is accessed. No read-modify-write protection is provided, so change with care.</p> <p>This register is write-clear. Writing to it with any value clears the register. Clearing this register also clears the COUNT bit of the STCTRL (page 395) register.</p>

24.1.3 Initialisation sequence

1. Clear the **ENABLE** bit, which is bit 0 of the **STCTRL** (page 395) register, to turn off the SysTick during initialisation.
2. Program the value in the **STRELOAD** (page 397) register to a value that is **1 bit less** than the actual value required.

For example, for a 80 MHz clock, the value of 800,000 is 10 ms.

3. Clear the **STCURRENT** (page 398) register by writing to it with any value.
4. Configure the **STCTRL** (page 395) register for the required operation.
 - Set **CLK_SRC** (bit 2) to 1 to use the system clock.
 - Clear **INTEN** (bit 1) if interrupt is not needed.
 - Set **ENABLE** (bit 0) to 1 to turn on the SysTick timer.

Note: When the processor is halted for debugging, the counter does not decrement.

- When SysTick timer is enabled, the timer counts down on each clock from the **STRELOAD** (page 397) value to zero.
- Reloads (wraps) to the value in the **STRELOAD** (page 397) register on the next clock edge.
- Clearing the **STRELOAD** (page 397) register disables the counter on the next wrap.
- When the timer reaches zero, the **COUNT** status (bit 16) in the **STCTRL** (page 395) register is set. The **COUNT** bit clears on reads.
- On the next clock cycle, the **STCURRENT** (page 398) register is loaded with the **STRELOAD** (page 397) value again.
- Writing to the **STCURRENT** (page 398) register clears the register and the **COUNT** status bit.
- On a read, the current value is the value of the register at the time the register is accessed.

24.1.3.1 Example code

```
STCTRL      EQU 0xE000E010      ; STCTRL register address
STRELOAD    EQU 0xE000E014      ; STRELOAD register address
STCURRENT   EQU 0xE000E018      ; STCURRENT register address

AREA |.text|, CODE, READONLY, ALIGN=2
THUMB

LDR R1, =STCTRL
LDR R0, [R1]
LDR R0, =0          ; Disable SysTick during setup
STR R0, [R1]

LDR R1, =STRELOAD
LDR R0, =0x00FFFFFF      ; Set the maximum reload value
STR R0, [R1]

LDR R1, =STCURRENT
LDR R0, =0          ; Write any number to clear STCURRENT register
STR R0, [R1]          ; Clear counter

LDR R1, =STCTRL
LDR R0, =0x05        ; Set enable bit and set CLK_SRC bit
STR R0, [R1]
BX LR
END                 ; End of file
```

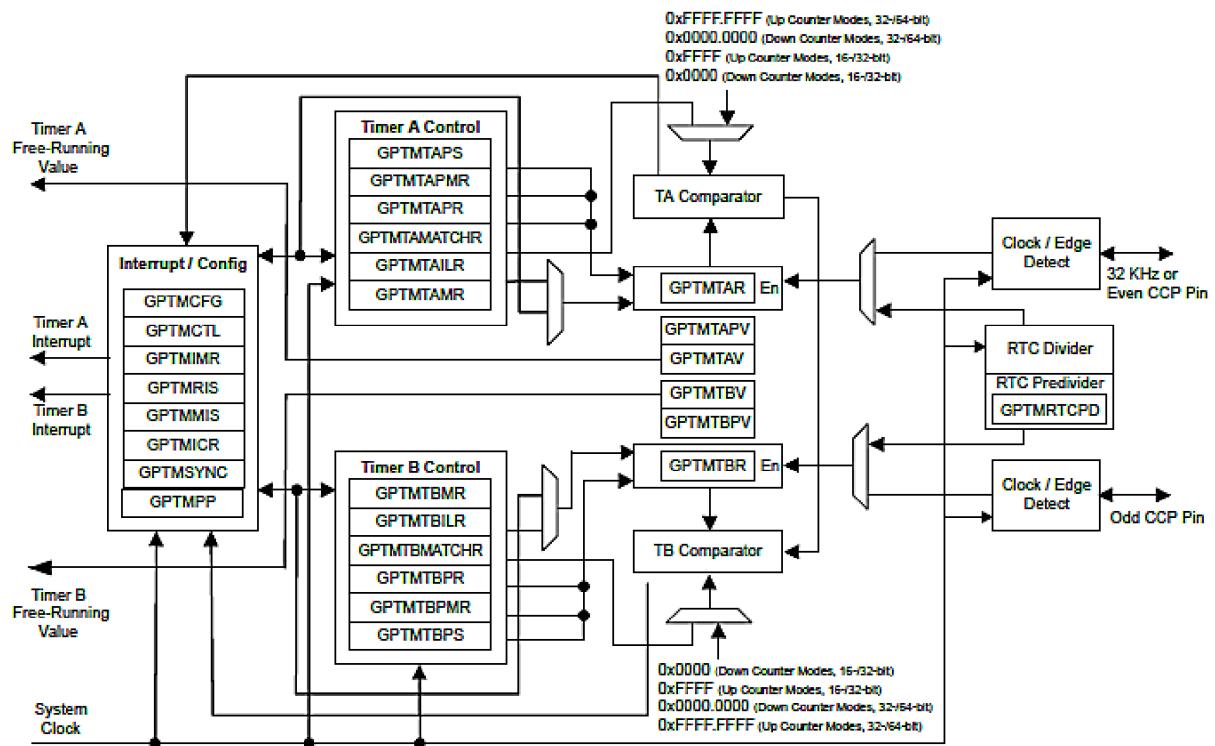
24.2 General purpose timers (GPTM)

- These are different timers to SysTick.
- Input capture mode to make time measurements of input signals.
- Input capture to measure period or pulse width of digital signals.
- Use to trigger interrupts on rising or falling edges of external signals.
- Programmable timers can be used to count or time external events that drive the Timer input pins.
- Timers can be used to trigger analogue-to-digital conversions (ADC) when a time-out occurs in periodic and one-shot mode.
- The general-purpose timer module (GPTM) contains six 16/32-bit GPTM blocks and six 32/64-bit wide GPTM blocks.
- Count up or down.
- Twelve 16/32-bit Capture Compare PWM pins (CCP).
- Twelve 32/64-bit Capture Compare PWM pins (CCP).
- Daisy chaining of timer modules to allow a single timer to initiate multiple timing events.
- Timer synchronisation allows selected timers to start counting on the same clock cycle.
- ADC event trigger.
- User-enabled stalling when the microcontroller asserts CPU Halt flag during debug (excluding RTC mode).
- Ability to determine the elapsed time between the assertion of the timer interrupt and entry into the interrupt service routine.
- Efficient transfers using Micro Direct Memory Access Controller (μ DMA)
 - Dedicated channel for each timer.
 - Burst request generated on timer interrupt.

Each block has the following functional options:

- 16/32-bit operating modes:
 - 16 or 32-bit programmable one-shot timers.
 - 16 or 32-bit programmable periodic timer.
 - 16-bit general-purpose timer with an 8-bit prescaler.
 - 32-bit Real-Time Clock (RTC) when using an external 32.768 kHz clock as the input.
 - 16-bit input-edge count or time-capture modes with an 8-bit prescaler.
 - 16-bit PWM mode with an 8-bit prescaler and software-programmable output inversion of the PWM signal.
- 32/64-bit operating modes:
 - 32 or 64-bit programmable one-shot timer.
 - 32 or 64-bit programmable periodic timer.
 - 32-bit general-purpose timer with an 16-bit prescaler.
 - 64-bit Real-Time Clock (RTC) when using an external 32.768 kHz clock as the input.
 - 32-bit input-edge count or time-capture modes with a 16-bit prescaler.
 - 32-bit PWM mode with a 16-bit prescaler and software-programmable output inversion of the PWM signal.

24.2.1 Block diagram



24.2.2 Input capture pins

Input capture pins are T_nCCP0 or T_nCCP1.

Timer	Up/Down Counter	Even CCP Pin	Odd CCP Pin
16/32-Bit Timer 0	Timer A	T0CCP0	-
	Timer B	-	T0CCP1
16/32-Bit Timer 1	Timer A	T1CCP0	-
	Timer B	-	T1CCP1
16/32-Bit Timer 2	Timer A	T2CCP0	-
	Timer B	-	T2CCP1
16/32-Bit Timer 3	Timer A	T3CCP0	-
	Timer B	-	T3CCP1
16/32-Bit Timer 4	Timer A	T4CCP0	-
	Timer B	-	T4CCP1
16/32-Bit Timer 5	Timer A	T5CCP0	-
	Timer B	-	T5CCP1
32/64-Bit Wide Timer 0	Timer A	WT0CCP0	-
	Timer B	-	WT0CCP1
32/64-Bit Wide Timer 1	Timer A	WT1CCP0	-
	Timer B	-	WT1CCP1
32/64-Bit Wide Timer 2	Timer A	WT2CCP0	-
	Timer B	-	WT2CCP1
32/64-Bit Wide Timer 3	Timer A	WT3CCP0	-
	Timer B	-	WT3CCP1
32/64-Bit Wide Timer 4	Timer A	WT4CCP0	-
	Timer B	-	WT4CCP1
32/64-Bit Wide Timer 5	Timer A	WT5CCP0	-
	Timer B	-	WT5CCP1

For input capture, an external signal is connected to an input, either T_nCCP0 or T_nCCP1. Three different input capture applications are:

- Interrupt service routine (ISR) is executed on the active edge of an external signal.
- Perform two rising edge input captures and subtract the two to get the **period**.
- Perform a rising edge and then a falling edge capture and subtract the two measurements to get the **pulse width**.

24.2.3 Functional description

- Each GPTM block are two free-running up/down counters, referred to as Timer A and Timer B, controlled by the **GPTMTAV** (page 419).
- Two prescaler registers **GPTMTAPR** (page 420).
- Two match registers **GPTMTAMATCHR** (page 421).
- Two prescaler match registers **GPTMTAPMR** (page 422).
- 16-bit counters have a 16-bit counting range and can be joined together to provide a 32-bit counting range.
- 32-bit counters have a 32-bit counting range and can be joined together to provide a 64-bit counting range.

24.2.4 Timer modes

Mode	Timer Use	Count Direction	Counter Size		Prescaler Size ^a		Prescaler Behavior (Count Direction)
			16/32-bit GPTM	32/64-bit Wide GPTM	16/32-bit GPTM	32/64-bit Wide GPTM	
One-shot	Individual	Up or Down	16-bit	32-bit	8-bit	16-bit	Timer Extension (Up), Prescaler (Down)
	Concatenated	Up or Down	32-bit	64-bit	-	-	N/A
Periodic	Individual	Up or Down	16-bit	32-bit	8-bit	16-bit	Timer Extension (Up), Prescaler (Down)
	Concatenated	Up or Down	32-bit	64-bit	-	-	N/A
RTC	Concatenated	Up	32-bit	64-bit	-	-	N/A
Edge Count	Individual	Up or Down	16-bit	32-bit	8-bit	16-bit	Timer Extension (Both)
Edge Time	Individual	Up or Down	16-bit	32-bit	8-bit	16-bit	Timer Extension (Both)
PWM	Individual	Down	16-bit	32-bit	8-bit	16-bit	Timer Extension

a. The prescaler is only available when the timers are used individually.

24.2.5 Register map

Base addresses

- 16/32-bit Timer 0: 0x4003.0000
- 16/32-bit Timer 1: 0x4003.1000
- 16/32-bit Timer 2: 0x4003.2000
- 16/32-bit Timer 3: 0x4003.3000
- 16/32-bit Timer 4: 0x4003.4000
- 16/32-bit Timer 5: 0x4003.5000
- 32/64-bit Wide Timer 0: 0x4003.6000
- 32/64-bit Wide Timer 1: 0x4003.7000
- 32/64-bit Wide Timer 2: 0x4004.C000
- 32/64-bit Wide Timer 3: 0x4004.D000
- 32/64-bit Wide Timer 4: 0x4004.E000
- 32/64-bit Wide Timer 5: 0x4004.F000

Offset	Name	Type	Reset	Description
0x000	GPTMCFG	RW	0x0000.0000	GPTM Configuration
0x004	GPTMTAMR	RW	0x0000.0000	GPTM Timer A Mode
0x008	GPTMTBMR	RW	0x0000.0000	GPTM Timer B Mode
0x00C	GPTMCTL	RW	0x0000.0000	GPTM Control
0x010	GPTMSYNC	RW	0x0000.0000	GPTM Synchronize
0x018	GPTMIMR	RW	0x0000.0000	GPTM Interrupt Mask
0x01C	GPTMRIS	RO	0x0000.0000	GPTM Raw Interrupt Status
0x020	GPTMMIS	RO	0x0000.0000	GPTM Masked Interrupt Status
0x024	GPTMICR	W1C	0x0000.0000	GPTM Interrupt Clear
0x028	GPTMTAILR	RW	0xFFFF.FFFF	GPTM Timer A Interval Load
0x02C	GPTMTBILR	RW	-	GPTM Timer B Interval Load
0x030	GPTMTAMATCHR	RW	0xFFFF.FFFF	GPTM Timer A Match
0x034	GPTMTBMATCHR	RW	0x0000.0000	GPTM Timer B Match
0x038	GPTMTAPR	RW	0x0000.0000	GPTM Timer A Prescale
0x03C	GPTMTBPR	RW	0x0000.0000	GPTM Timer B Prescale
0x040	GPTMTAPMR	RW	0x0000.0000	GPTM Timer A Prescale Match
0x044	GPTMTBPMR	RW	0x0000.0000	GPTM Timer B Prescale Match
0x048	GPTMTAR	RO	0xFFFF.FFFF	GPTM Timer A
0x04C	GPTMTBR	RO	-	GPTM Timer B
0x050	GPTMTAV	RW	0xFFFF.FFFF	GPTM Timer A Value
0x054	GPTMTBV	RW	-	GPTM Timer B Value
0x058	GPTMRTCPD	RO	0x0000.7FFF	GPTM RTC Predivide
0x05C	GPTMTAPS	RO	0x0000.0000	GPTM Timer A Prescale Snapshot
0x060	GPTMTBPS	RO	0x0000.0000	GPTM Timer B Prescale Snapshot
0x064	GPTMTAPV	RO	0x0000.0000	GPTM Timer A Prescale Value
0x068	GPTMTBPV	RO	0x0000.0000	GPTM Timer B Prescale Value
0xFC0	GPTMPP	RO	0x0000.0000	GPTM Peripheral Properties

24.2.6 Timer 0 registers

\$4003.0000	31-3	2-0	GPTMCFG	Name													
\$4003.0004	31-4	3	TAAMS	TACMR	TAMR	TIMER0_TAMR_R											
\$4003.0008	31-4	3	TBAMS	TBCMR	TBMR	TIMER0_TBMR_R											
\$4003.000C	14	13	11-10	8	6	5	3-2	0	TBPWML	TBOTE	TBEVENT	TBEN	TAPWML	TAOTE	TAEVENT	TAEN	TIMER0_CTL_R
\$4003.0018	31-11	10	9	8	7-4	2	1	0	CBEIM	CBMIM	TBTOIM	CAEIM	CAMIM	TATOIM	TIMER0_IMR_R		
\$4003.001C	31-11	10	9	8	7-4	2	1	0	CBERIS	CBMRIS	TBTORIS	CAERIS	CAMRIS	TATORIS	TIMER0_RIS_R		
\$4003.0020	31-11	10	9	8	7-4	2	1	0	CBEMIS	CBMMIS	TBTOMIS	CAEMIS	CAMMIS	TATOMIS	TIMER0_MIS_R		
\$4003.0024	31-11	10	9	8	7-4	2	1	0	CBECINT	CBMCINT	TBTOCINT	CAECINT	CAMCINT	TATOCINT	TIMER0_ICR_R		
\$4003.0028	31-16	TAIRH	TAILRL	15-0	TIMER0_TAILR_R												
\$4003.002C	31-16	TBILRL	15-0	TIMER0_TBILR_R													
\$4003.0030	31-16	TAMRH	TAMRL	15-0	TAMATCHR_R												
\$4003.0034	31-16	TBMRL	15-0	TBMATCHR_R													
\$4003.0038	31-8	TAPSR	7-0	TIMER0_TAPR_R													
\$4003.003C	31-8	TBPSR	7-0	TIMER0_TBPR_R													
\$4003.0040	31-8	TAPSMR	7-0	TIMER0_TAPMR_R													
\$4003.0044	31-8	TBPSMR	7-0	TIMER0_TBPMR_R													
\$4003.0048	31-16	TARH	TARL	15-0	TIMER0_TAR_R												
\$4003.004C	31-16	TBRL	15-0	TIMER0_TBR_R													

24.2.7 Essential function in registers

- External input pins (CCP0, CCP1)
- Trigger flag bit [Raw Interrupt Status] (CAERIS)
- Two edge control bits (positive, negative, both edges) or event mode (TAEVENT)
- Interrupt Mask (CAEIM)
- 16/32-bit capture register (TAR, TBR)

24.2.8 Overview of GPTM use

- An external signal is connected to input capture pin.
- Specify whether input capture event will trigger on rising or falling edge during initialisation.
- The timer counts down from the 16-bit integer limit. When it hits zero, it rolls over back to 0xFFFF and continues counting down.
- When an input capture event happens, the current timer value is copied into the input capture register (TAR, TBR).
- The input capture flag is set (CAERIS).
- An interrupt is requested if armed (CAEIM).

24.2.9 Prescaler configurations

Prescale (8-bit value)	Number of Timer Clocks (Tc ^a)	Max Time	Units
00000000	1	0.8192	ms
00000001	2	1.6384	ms
00000010	3	2.4576	ms
-----	-	-	-
11111101	254	208.0768	ms
11111110	255	208.896	ms
11111111	256	209.7152	ms

a. Tc is the clock period.

Table 1: 16-bit Timer with Prescaler Configurations

Prescale (16-bit value)	Number of Timer Clocks (Tc ^a)	Max Time	Units
0x0000	1	53.687	s
0x0001	2	107.374	s
0x0002	3	214.748	s
-----	-	-	-
0xFFFFD	65534	0.879	10 ⁶ s
0xFFFFE	65535	1.759	10 ⁶ s
0xFFFF	65536	3.518	10 ⁶ s

a. Tc is the clock period.

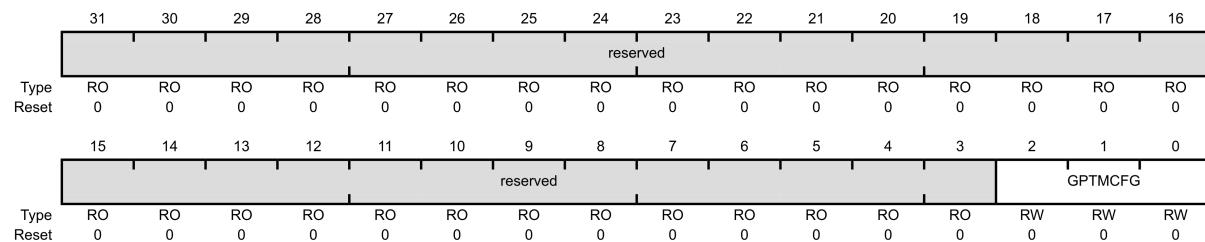
Table 2: 32-bit Timer (configured in 32/64-bit mode) with Prescaler Configurations

24.2.10 Registers

- GPTM configuration ([GPTMCFG \(page 407\)](#)) register.
- GPTM Timer A mode ([GPTMTAMR \(page 414\)](#)) register.
- GPTM Timer B mode ([GPTMTBMR \(page 427\)](#)) register.
 - When in one of the concatenated modes, Timer A and Timer B can only operate in one mode.
 - When configured in individual mode, Timer A and Timer B can be independently configured in any combination of the individual modes.

24.2.10.1 GPTM Configuration (GPTMCFG)

- 16/32-bit Timer 0 base: 0x4003.0000
- 16/32-bit Timer 1 base: 0x4003.1000
- 16/32-bit Timer 2 base: 0x4003.2000
- 16/32-bit Timer 3 base: 0x4003.3000
- 16/32-bit Timer 4 base: 0x4003.4000
- 16/32-bit Timer 5 base: 0x4003.5000
- 32/64-bit Wide Timer 0 base: 0x4003.6000
- 32/64-bit Wide Timer 1 base: 0x4003.7000
- 32/64-bit Wide Timer 2 base: 0x4004.C000
- 32/64-bit Wide Timer 3 base: 0x4004.D000
- 32/64-bit Wide Timer 4 base: 0x4004.E000
- 32/64-bit Wide Timer 5 base: 0x4004.F000
- Offset 0x000
- Type RW, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description												
31:3	Reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.												
2:0	GPTMCFG	RW	0x0	<p>GPTM Configuration</p> <p>The GPTMCFG values are defined as follows:</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>For a 16/32-bit timer, this value selects the 32-bit timer configuration. For a 32/64-bit wide timer, this value selects the 64-bit configuration.</td> </tr> <tr> <td>0x1</td> <td>For a 16/32-bit timer, this value selects the 32-bit real-time clock (RTC) counter configuration. For a 32/64-bit wide timer, this value selects the 64-bit real-time clock (RTC) counter configuration.</td> </tr> <tr> <td>0x2 - 0x3</td> <td>Reserved</td> </tr> <tr> <td>0x4</td> <td>For a 16/32-bit timer, this value selects the 16-bit timer configuration. For a 32/64-bit wide timer, this value selects the 32-bit timer configuration. The function is controlled by bits 1:0 of GPTMTAMR (page 414) and GPTMTBMR (page 427).</td> </tr> <tr> <td>0x5 - 0x7</td> <td>Reserved</td> </tr> </tbody> </table>	Value	Description	0x0	For a 16/32-bit timer, this value selects the 32-bit timer configuration. For a 32/64-bit wide timer, this value selects the 64-bit configuration.	0x1	For a 16/32-bit timer, this value selects the 32-bit real-time clock (RTC) counter configuration. For a 32/64-bit wide timer, this value selects the 64-bit real-time clock (RTC) counter configuration.	0x2 - 0x3	Reserved	0x4	For a 16/32-bit timer, this value selects the 16-bit timer configuration. For a 32/64-bit wide timer, this value selects the 32-bit timer configuration. The function is controlled by bits 1:0 of GPTMTAMR (page 414) and GPTMTBMR (page 427).	0x5 - 0x7	Reserved
Value	Description															
0x0	For a 16/32-bit timer, this value selects the 32-bit timer configuration. For a 32/64-bit wide timer, this value selects the 64-bit configuration.															
0x1	For a 16/32-bit timer, this value selects the 32-bit real-time clock (RTC) counter configuration. For a 32/64-bit wide timer, this value selects the 64-bit real-time clock (RTC) counter configuration.															
0x2 - 0x3	Reserved															
0x4	For a 16/32-bit timer, this value selects the 16-bit timer configuration. For a 32/64-bit wide timer, this value selects the 32-bit timer configuration. The function is controlled by bits 1:0 of GPTMTAMR (page 414) and GPTMTBMR (page 427).															
0x5 - 0x7	Reserved															

24.2.10.2 GPTM Control (GPTMCTL)

- 16/32-bit Timer 0 base: 0x4003.0000
- 16/32-bit Timer 1 base: 0x4003.1000
- 16/32-bit Timer 2 base: 0x4003.2000
- 16/32-bit Timer 3 base: 0x4003.3000
- 16/32-bit Timer 4 base: 0x4003.4000
- 16/32-bit Timer 5 base: 0x4003.5000
- 32/64-bit Wide Timer 0 base: 0x4003.6000
- 32/64-bit Wide Timer 1 base: 0x4003.7000
- 32/64-bit Wide Timer 2 base: 0x4004.C000
- 32/64-bit Wide Timer 3 base: 0x4004.D000
- 32/64-bit Wide Timer 4 base: 0x4004.E000
- 32/64-bit Wide Timer 5 base: 0x4004.F000
- Offset 0x00C
- Type RW, reset 0x0000.0000

Note: Bits in this register should only be changed when the TnEN bit for the respective timer is cleared.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved																
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	TBPWML	TBOTE	reserved	TBEVENT	TBSTALL	TBEN	reserved	TAPWML	TAOTE	RTCEN	TAEVENT	TASTALL	TAEN		
Type	RO	RW	RW	RO	RW	RW	RW	RW	RO	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description						
31:15	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
14	TBPWML	RW	0	<p>GPTM Timer B PWM Output Level</p> <p>The TBPWML values are defined as follows:</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Output is unaffected.</td> </tr> <tr> <td>1</td> <td>Output is inverted.</td> </tr> </tbody> </table>	Value	Description	0	Output is unaffected.	1	Output is inverted.
Value	Description									
0	Output is unaffected.									
1	Output is inverted.									

Bit/Field	Name	Type	Reset	Description										
13	TBOTE	RW	0	<p>GPTM Timer B Output Trigger Enable</p> <p>The TBOTE values are defined as follows:</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The output Timer B ADC trigger is disabled.</td> </tr> <tr> <td>1</td> <td>The output Timer B ADC trigger is enabled.</td> </tr> </tbody> </table> <p>Note: The timer must be configured for one-shot or periodic time-out mode to produce an ADC trigger assertion. The GPTM does not generate triggers for match, compare events or compare match events.</p> <p>In addition, the ADC must be enabled and the timer selected as a trigger source with the EXn bit in the ADCCEMUX (page 178) register.</p>	Value	Description	0	The output Timer B ADC trigger is disabled.	1	The output Timer B ADC trigger is enabled.				
Value	Description													
0	The output Timer B ADC trigger is disabled.													
1	The output Timer B ADC trigger is enabled.													
12	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.										
11:10	TBEVENT	RW	0x0	<p>GPTM Timer B Event Mode</p> <p>The TBEVENT values are defined as follows:</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Positive edge</td> </tr> <tr> <td>0x1</td> <td>Negative edge</td> </tr> <tr> <td>0x2</td> <td>Reserved</td> </tr> <tr> <td>0x3</td> <td>Both edges</td> </tr> </tbody> </table> <p>Note: If PWM output inversion is enabled, edge detection interrupt behavior is reversed. Thus, if a positive-edge interrupt trigger has been set and the PWM inversion generates a positive edge, no event-trigger interrupt asserts. Instead, the interrupt is generated on the negative edge of the PWM signal.</p>	Value	Description	0x0	Positive edge	0x1	Negative edge	0x2	Reserved	0x3	Both edges
Value	Description													
0x0	Positive edge													
0x1	Negative edge													
0x2	Reserved													
0x3	Both edges													

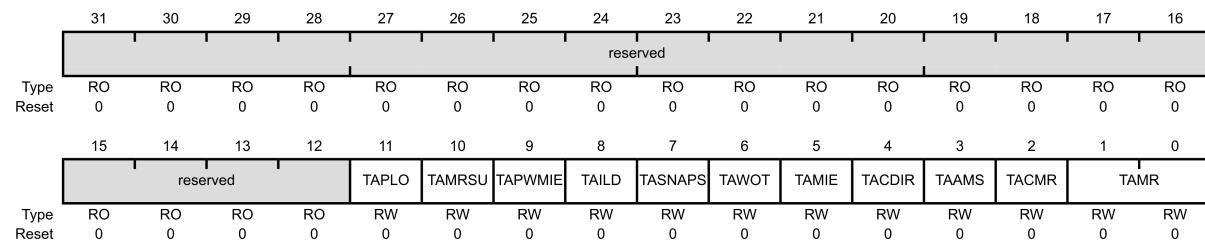
Bit/Field	Name	Type	Reset	Description						
9	TBSTALL	RW	0	<p>GPTM Timer B Stall Enable</p> <p>The TBSTALL values are defined as follows:</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Timer B continues counting while the processor is halted by the debugger.</td> </tr> <tr> <td>1</td> <td>Timer B freezes counting while the processor is halted by the debugger.</td> </tr> </tbody> </table> <p>If the processor is executing normally, the TBSTALL bit is ignored.</p>	Value	Description	0	Timer B continues counting while the processor is halted by the debugger.	1	Timer B freezes counting while the processor is halted by the debugger.
Value	Description									
0	Timer B continues counting while the processor is halted by the debugger.									
1	Timer B freezes counting while the processor is halted by the debugger.									
8	TBEN	RW	0	<p>GPTM Timer B Enable</p> <p>The TBEN values are defined as follows:</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Timer B is disabled.</td> </tr> <tr> <td>1</td> <td>Timer B is enabled and begins counting or the capture logic is enabled based on the GPTMCFG (page 407) register.</td> </tr> </tbody> </table>	Value	Description	0	Timer B is disabled.	1	Timer B is enabled and begins counting or the capture logic is enabled based on the GPTMCFG (page 407) register.
Value	Description									
0	Timer B is disabled.									
1	Timer B is enabled and begins counting or the capture logic is enabled based on the GPTMCFG (page 407) register.									
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
6	TAPWML	RW	0	<p>GPTM Timer A PWM Output Level</p> <p>The TAPWML values are defined as follows:</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Output is unaffected.</td> </tr> <tr> <td>1</td> <td>Output is inverted.</td> </tr> </tbody> </table>	Value	Description	0	Output is unaffected.	1	Output is inverted.
Value	Description									
0	Output is unaffected.									
1	Output is inverted.									

Bit/Field	Name	Type	Reset	Description										
5	TAOTE	RW	0	<p>GPTM Timer A Output Trigger Enable</p> <p>The TAOTE values are defined as follows:</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The output Timer A ADC trigger is disabled.</td> </tr> <tr> <td>1</td> <td>The output Timer A ADC trigger is enabled.</td> </tr> </tbody> </table> <p>Note: The timer must be configured for one-shot or periodic time-out mode to produce an ADC trigger assertion. The GPTM does not generate triggers for match, compare events or compare match events.</p> <p>In addition, the ADC must be enabled and the timer selected as a trigger source with the EXn bit in the ADCEMUX (page 178) register.</p>	Value	Description	0	The output Timer A ADC trigger is disabled.	1	The output Timer A ADC trigger is enabled.				
Value	Description													
0	The output Timer A ADC trigger is disabled.													
1	The output Timer A ADC trigger is enabled.													
4	RTCEN	RW	0	<p>GPTM RTC Stall Enable</p> <p>The RTCEN values are defined as follows:</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>RTC counting continues while the processor is halted by the debugger.</td> </tr> <tr> <td>1</td> <td>RTC counting continues while the processor is halted by the debugger.</td> </tr> </tbody> </table> <p>If the RTCEN bit is set, it prevents the timer from stalling in all operating modes, even if TnSTALL is set.</p>	Value	Description	0	RTC counting continues while the processor is halted by the debugger.	1	RTC counting continues while the processor is halted by the debugger.				
Value	Description													
0	RTC counting continues while the processor is halted by the debugger.													
1	RTC counting continues while the processor is halted by the debugger.													
3:2	TAEVENT	RW	0x0	<p>GPTM Timer A Event Mode</p> <p>The TAEVENT values are defined as follows:</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Positive edge</td> </tr> <tr> <td>0x1</td> <td>Negative edge</td> </tr> <tr> <td>0x2</td> <td>Reserved</td> </tr> <tr> <td>0x3</td> <td>Both edges</td> </tr> </tbody> </table> <p>Note: If PWM output inversion is enabled, edge detection interrupt behavior is reversed. Thus, if a positive-edge interrupt trigger has been set and the PWM inversion generates a positive edge, no event-trigger interrupt asserts. Instead, the interrupt is generated on the negative edge of the PWM signal.</p>	Value	Description	0x0	Positive edge	0x1	Negative edge	0x2	Reserved	0x3	Both edges
Value	Description													
0x0	Positive edge													
0x1	Negative edge													
0x2	Reserved													
0x3	Both edges													

Bit/Field	Name	Type	Reset	Description						
1	TASTALL	RW	0	<p>GPTM Timer A Stall Enable</p> <p>The TASTALL values are defined as follows:</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Timer A continues counting while the processor is halted by the debugger.</td> </tr> <tr> <td>1</td> <td>Timer A freezes counting while the processor is halted by the debugger.</td> </tr> </tbody> </table> <p>If the processor is executing normally, the TASTALL bit is ignored.</p>	Value	Description	0	Timer A continues counting while the processor is halted by the debugger.	1	Timer A freezes counting while the processor is halted by the debugger.
Value	Description									
0	Timer A continues counting while the processor is halted by the debugger.									
1	Timer A freezes counting while the processor is halted by the debugger.									
0	TAEN	RW	0	<p>GPTM Timer A Enable</p> <p>The TAEN values are defined as follows:</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Timer A is disabled.</td> </tr> <tr> <td>1</td> <td>Timer A is enabled and begins counting or the capture logic is enabled based on the GPTMCFG (page 407) register.</td> </tr> </tbody> </table>	Value	Description	0	Timer A is disabled.	1	Timer A is enabled and begins counting or the capture logic is enabled based on the GPTMCFG (page 407) register.
Value	Description									
0	Timer A is disabled.									
1	Timer A is enabled and begins counting or the capture logic is enabled based on the GPTMCFG (page 407) register.									

24.2.10.3 GPTM Timer A Mode (GPTMTAMR)

- 16/32-bit Timer 0 base: 0x4003.0000
- 16/32-bit Timer 1 base: 0x4003.1000
- 16/32-bit Timer 2 base: 0x4003.2000
- 16/32-bit Timer 3 base: 0x4003.3000
- 16/32-bit Timer 4 base: 0x4003.4000
- 16/32-bit Timer 5 base: 0x4003.5000
- 32/64-bit Wide Timer 0 base: 0x4003.6000
- 32/64-bit Wide Timer 1 base: 0x4003.7000
- 32/64-bit Wide Timer 2 base: 0x4004.C000
- 32/64-bit Wide Timer 3 base: 0x4004.D000
- 32/64-bit Wide Timer 4 base: 0x4004.E000
- 32/64-bit Wide Timer 5 base: 0x4004.F000
- Offset 0x004
- Type RW, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description	
31:12	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.	
11	TAPLO	RW	0	GPTM Timer A PWM Legacy Operation Value Description 0 Legacy operation with CCP pin driven Low when the GPTMTAILR (page 423) is reloaded after the timer reaches 0. 1 CCP is driven High when the GPTMTAILR (page 423) is reloaded after the timer reaches 0. This bit is only valid in PWM mode.	

Bit/Field	Name	Type	Reset	Description						
10	TAMRSU	RW	0	<p>GPTM Timer A Match Register Update</p> <table> <thead> <tr> <th data-bbox="747 316 827 350">Value</th> <th data-bbox="827 316 986 350">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="747 361 827 451">0</td> <td data-bbox="827 361 1335 451">Update the GPTMTAMATCHR (page 421) register and the GPTMTAPR (page 420) register, if used, on the next cycle.</td> </tr> <tr> <td data-bbox="747 462 827 597">1</td> <td data-bbox="827 462 1335 597">Update the GPTMTAMATCHR (page 421) register and the GPTMTAPR (page 420) register, if used, on the next timeout.</td> </tr> </tbody> </table> <p>If the timer is disabled (TAEN is clear) when this bit is set, GPTMTAMATCHR (page 421) and GPTMTAPR (page 420) are updated when the timer is enabled. If the timer is stalled (TASTALL is set), GPTMTAMATCHR (page 421) and GPTMTAPR (page 420) are updated according to the configuration of this bit.</p>	Value	Description	0	Update the GPTMTAMATCHR (page 421) register and the GPTMTAPR (page 420) register, if used, on the next cycle.	1	Update the GPTMTAMATCHR (page 421) register and the GPTMTAPR (page 420) register, if used, on the next timeout.
Value	Description									
0	Update the GPTMTAMATCHR (page 421) register and the GPTMTAPR (page 420) register, if used, on the next cycle.									
1	Update the GPTMTAMATCHR (page 421) register and the GPTMTAPR (page 420) register, if used, on the next timeout.									
9	TAPWMIE	RW	0	<p>GPTM A PWM Interrupt Enable</p> <p>This bit enables interrupts in PWM mode on rising, falling, or both edges of the CCP output, as defined by the TAEVENT field in the GPTMCTL (page 409) register.</p> <table> <thead> <tr> <th data-bbox="747 1136 827 1170">Value</th> <th data-bbox="827 1136 986 1170">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="747 1181 827 1215">0</td> <td data-bbox="827 1181 1271 1215">Capture event interrupt is disabled.</td> </tr> <tr> <td data-bbox="747 1226 827 1260">1</td> <td data-bbox="827 1226 1271 1260">Capture event interrupt is enabled.</td> </tr> </tbody> </table> <p>This bit is only valid in PWM mode.</p>	Value	Description	0	Capture event interrupt is disabled.	1	Capture event interrupt is enabled.
Value	Description									
0	Capture event interrupt is disabled.									
1	Capture event interrupt is enabled.									

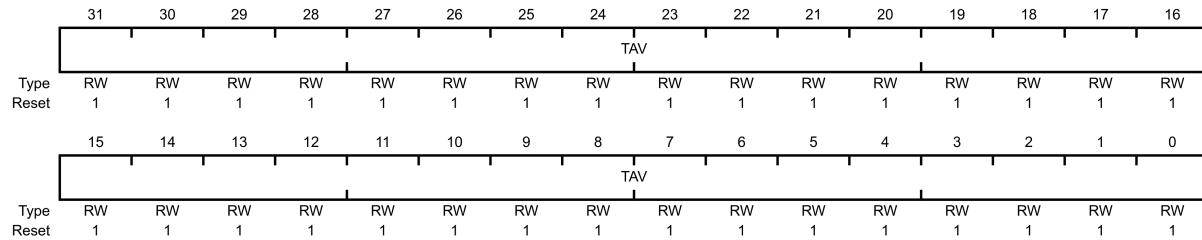
Bit/Field	Name	Type	Reset	Description						
8	TAILD	RW	0	<p>GPTM Timer A Interval Load Write</p> <table> <thead> <tr> <th data-bbox="747 316 811 350">Value</th> <th data-bbox="811 316 1002 350">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="747 361 779 395">0</td> <td data-bbox="811 361 1367 653">Update the GPTMTAR (page 424) and GPTMTAV (page 419) registers with the value in the GPTMTAILR (page 423) register on the next cycle. Also update the GPTMTAPS (page 425) and GPTMTAPV (page 426) registers with the value in the GPTMTAPR (page 420) register on the next cycle.</td> </tr> <tr> <td data-bbox="747 676 779 709">1</td> <td data-bbox="811 676 1367 968">Update the GPTMTAR (page 424) and GPTMTAV (page 419) registers with the value in the GPTMTAILR (page 423) register on the next timeout. Also update the GPTMTAPS (page 425) and GPTMTAPV (page 426) registers with the value in the GPTMTAPR (page 420) register on the next timeout.</td> </tr> </tbody> </table> <p>Note the state of this bit has no effect when counting up.</p> <p>The bit descriptions above apply if the timer is enabled and running. If the timer is disabled (TAEN is clear) when this bit is set, GPTMTAR (page 424), GPTMTAV (page 419), GPTMTAPS (page 425), and GPTMTAPV (page 426) are updated when the timer is enabled. If the timer is stalled (TASTALL is set), GPTMTAR (page 424) and GPTMTAPS (page 425) are updated according to the configuration of this bit.</p>	Value	Description	0	Update the GPTMTAR (page 424) and GPTMTAV (page 419) registers with the value in the GPTMTAILR (page 423) register on the next cycle. Also update the GPTMTAPS (page 425) and GPTMTAPV (page 426) registers with the value in the GPTMTAPR (page 420) register on the next cycle.	1	Update the GPTMTAR (page 424) and GPTMTAV (page 419) registers with the value in the GPTMTAILR (page 423) register on the next timeout. Also update the GPTMTAPS (page 425) and GPTMTAPV (page 426) registers with the value in the GPTMTAPR (page 420) register on the next timeout.
Value	Description									
0	Update the GPTMTAR (page 424) and GPTMTAV (page 419) registers with the value in the GPTMTAILR (page 423) register on the next cycle. Also update the GPTMTAPS (page 425) and GPTMTAPV (page 426) registers with the value in the GPTMTAPR (page 420) register on the next cycle.									
1	Update the GPTMTAR (page 424) and GPTMTAV (page 419) registers with the value in the GPTMTAILR (page 423) register on the next timeout. Also update the GPTMTAPS (page 425) and GPTMTAPV (page 426) registers with the value in the GPTMTAPR (page 420) register on the next timeout.									
7	TASNAPS	RW	0	<p>GPTM Timer A Snap-Shot Mode</p> <table> <thead> <tr> <th data-bbox="747 1520 811 1554">Value</th> <th data-bbox="811 1520 1002 1554">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="747 1565 779 1599">0</td> <td data-bbox="811 1565 1192 1599">Snap-shot mode is disabled.</td> </tr> <tr> <td data-bbox="747 1619 779 1653">1</td> <td data-bbox="811 1619 1367 1956">If Timer A is configured in the periodic mode, the actual free-running capture or snapshot value of Timer A is loaded at the time-out event/capture or snapshot event into the GPTM Timer A (GPTMTAR (page 424)) register. If the timer prescaler is used, the prescaler snapshot is loaded into the GPTM Timer A (GPTMTAPR (page 420)) register.</td> </tr> </tbody> </table>	Value	Description	0	Snap-shot mode is disabled.	1	If Timer A is configured in the periodic mode, the actual free-running capture or snapshot value of Timer A is loaded at the time-out event/capture or snapshot event into the GPTM Timer A (GPTMTAR (page 424)) register. If the timer prescaler is used, the prescaler snapshot is loaded into the GPTM Timer A (GPTMTAPR (page 420)) register.
Value	Description									
0	Snap-shot mode is disabled.									
1	If Timer A is configured in the periodic mode, the actual free-running capture or snapshot value of Timer A is loaded at the time-out event/capture or snapshot event into the GPTM Timer A (GPTMTAR (page 424)) register. If the timer prescaler is used, the prescaler snapshot is loaded into the GPTM Timer A (GPTMTAPR (page 420)) register.									

Bit/Field	Name	Type	Reset	Description								
6	TAWOT	RW	0	<p>GPTM Timer A Wait-on-Trigger</p> <table> <thead> <tr> <th data-bbox="747 316 811 350">Value</th> <th data-bbox="811 316 986 350">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="747 361 811 428">0</td> <td data-bbox="811 361 1335 428">Timer A begins counting as soon as it is enabled.</td> </tr> <tr> <td data-bbox="747 440 811 698">1</td> <td data-bbox="811 440 1351 698">If Timer A is enabled (TAEN is set in the GPTMCTL (page 409) register), Timer A does not begin counting until it receives a trigger from the timer in the previous position in the daisy chain. This function is valid for one-shot, periodic, and PWM modes.</td> </tr> </tbody> </table> <p>This bit must be clear for GP Timer Module 0, Timer A.</p>	Value	Description	0	Timer A begins counting as soon as it is enabled.	1	If Timer A is enabled (TAEN is set in the GPTMCTL (page 409) register), Timer A does not begin counting until it receives a trigger from the timer in the previous position in the daisy chain. This function is valid for one-shot, periodic, and PWM modes.		
Value	Description											
0	Timer A begins counting as soon as it is enabled.											
1	If Timer A is enabled (TAEN is set in the GPTMCTL (page 409) register), Timer A does not begin counting until it receives a trigger from the timer in the previous position in the daisy chain. This function is valid for one-shot, periodic, and PWM modes.											
5	TAMIE	RW	0	<p>GPTM Timer A Match Interrupt Enable</p> <table> <thead> <tr> <th data-bbox="747 891 811 925">Value</th> <th data-bbox="811 891 986 925">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="747 936 811 1003">0</td> <td data-bbox="811 936 1335 1003">The match interrupt is disabled for match events.</td> </tr> <tr> <td data-bbox="747 1015 811 1394">1</td> <td data-bbox="811 1015 1367 1394"> <p>Note: Clearing the TAMIE bit in the GPTMTAMR (page 414) register prevents assertion of μDMA or ADC requests generated on a match event. Even if the TATODMAEN bit is set in the GPTMDMAEV register or the TATOADCEN bit is set in the GPTMADCEV register, a μDMA or ADC match trigger is not sent to the μDMA or ADC, respectively, when the TAMIE bit is clear.</p> </td> </tr> <tr> <td data-bbox="747 1405 811 1572">1</td> <td data-bbox="811 1405 1367 1572">An interrupt is generated when the match value in the GPTMTAMATCHR (page 421) register is reached in the one-shot and periodic modes.</td> </tr> </tbody> </table>	Value	Description	0	The match interrupt is disabled for match events.	1	<p>Note: Clearing the TAMIE bit in the GPTMTAMR (page 414) register prevents assertion of μDMA or ADC requests generated on a match event. Even if the TATODMAEN bit is set in the GPTMDMAEV register or the TATOADCEN bit is set in the GPTMADCEV register, a μDMA or ADC match trigger is not sent to the μDMA or ADC, respectively, when the TAMIE bit is clear.</p>	1	An interrupt is generated when the match value in the GPTMTAMATCHR (page 421) register is reached in the one-shot and periodic modes.
Value	Description											
0	The match interrupt is disabled for match events.											
1	<p>Note: Clearing the TAMIE bit in the GPTMTAMR (page 414) register prevents assertion of μDMA or ADC requests generated on a match event. Even if the TATODMAEN bit is set in the GPTMDMAEV register or the TATOADCEN bit is set in the GPTMADCEV register, a μDMA or ADC match trigger is not sent to the μDMA or ADC, respectively, when the TAMIE bit is clear.</p>											
1	An interrupt is generated when the match value in the GPTMTAMATCHR (page 421) register is reached in the one-shot and periodic modes.											
4	TACDIR	RW	0	<p>GPTM Timer A Count Direction</p> <table> <thead> <tr> <th data-bbox="747 1664 811 1697">Value</th> <th data-bbox="811 1664 986 1697">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="747 1709 811 1742">0</td> <td data-bbox="811 1709 1129 1742">The timer counts down.</td> </tr> <tr> <td data-bbox="747 1754 811 1821">1</td> <td data-bbox="811 1754 1319 1821">The timer counts up. When counting up, the timer starts from a value of 0x0.</td> </tr> </tbody> </table> <p>When in PWM or RTC mode, the status of this bit is ignored. PWM mode always counts down and RTC mode always counts up.</p>	Value	Description	0	The timer counts down.	1	The timer counts up. When counting up, the timer starts from a value of 0x0.		
Value	Description											
0	The timer counts down.											
1	The timer counts up. When counting up, the timer starts from a value of 0x0.											

Bit/Field	Name	Type	Reset	Description										
3	TAAMS	RW	0	<p>GPTM Timer A Alternate Mode Select</p> <p>The TAAMS values are defined as follows:</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Capture or compare mode is enabled.</td> </tr> <tr> <td>1</td> <td>PWM mode is enabled.</td> </tr> </tbody> </table> <p>Note: To enable PWM mode, you must also clear the TACMR bit and configure the TAMR field to 0x1 or 0x2.</p>	Value	Description	0	Capture or compare mode is enabled.	1	PWM mode is enabled.				
Value	Description													
0	Capture or compare mode is enabled.													
1	PWM mode is enabled.													
2	TACMR	RW	0	<p>GPTM Timer A Capture Mode</p> <p>The TACMR values are defined as follows:</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Edge-Count mode</td> </tr> <tr> <td>1</td> <td>Edge-Time mode</td> </tr> </tbody> </table>	Value	Description	0	Edge-Count mode	1	Edge-Time mode				
Value	Description													
0	Edge-Count mode													
1	Edge-Time mode													
1:0	TAMR	RW	0x0	<p>GPTM Timer A Mode</p> <p>The TAMR values are defined as follows:</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Reserved</td> </tr> <tr> <td>0x1</td> <td>One-Shot Timer mode</td> </tr> <tr> <td>0x2</td> <td>Periodic Timer mode</td> </tr> <tr> <td>0x3</td> <td>Capture mode</td> </tr> </tbody> </table> <p>The Timer mode is based on the timer configuration defined by bits 2:0 in the GPTMCFG (page 407) register.</p>	Value	Description	0x0	Reserved	0x1	One-Shot Timer mode	0x2	Periodic Timer mode	0x3	Capture mode
Value	Description													
0x0	Reserved													
0x1	One-Shot Timer mode													
0x2	Periodic Timer mode													
0x3	Capture mode													

24.2.10.4 GPTM Timer A Value (GPTMTAV)

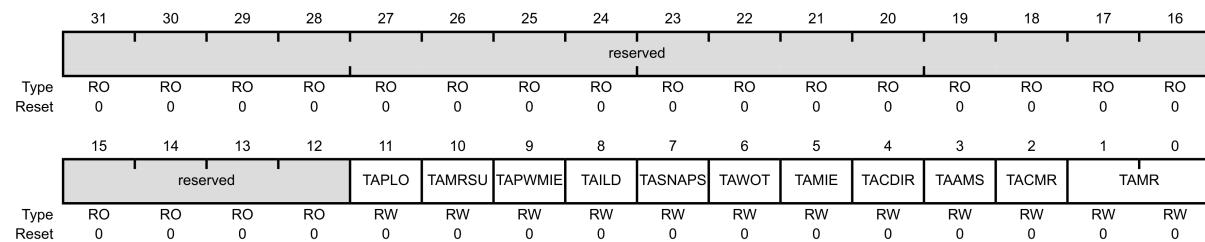
- 16/32-bit Timer 0 base: 0x4003.0000
- 16/32-bit Timer 1 base: 0x4003.1000
- 16/32-bit Timer 2 base: 0x4003.2000
- 16/32-bit Timer 3 base: 0x4003.3000
- 16/32-bit Timer 4 base: 0x4003.4000
- 16/32-bit Timer 5 base: 0x4003.5000
- 32/64-bit Wide Timer 0 base: 0x4003.6000
- 32/64-bit Wide Timer 1 base: 0x4003.7000
- 32/64-bit Wide Timer 2 base: 0x4004.C000
- 32/64-bit Wide Timer 3 base: 0x4004.D000
- 32/64-bit Wide Timer 4 base: 0x4004.E000
- 32/64-bit Wide Timer 5 base: 0x4004.F000
- Offset 0x050
- Type RW, reset 0xFFFF.FFFF



Bit/Field	Name	Type	Reset	Description
31:0	TAV	RW	0xFFFF.FFFF	<p>GPTM Timer A Value</p> <p>A read returns the current, free-running value of Timer A in all modes. When written, the value written into this register is loaded into the GPTMTAR (page 424) register on the next clock cycle.</p> <p>Note: In 16-bit mode, only the lower 16-bits of the GPTMTAV (page 419) register can be written with a new value. Writes to the prescaler bits have no effect.</p>

24.2.10.5 GPTM Timer A Prescale (GPTMTAPR)

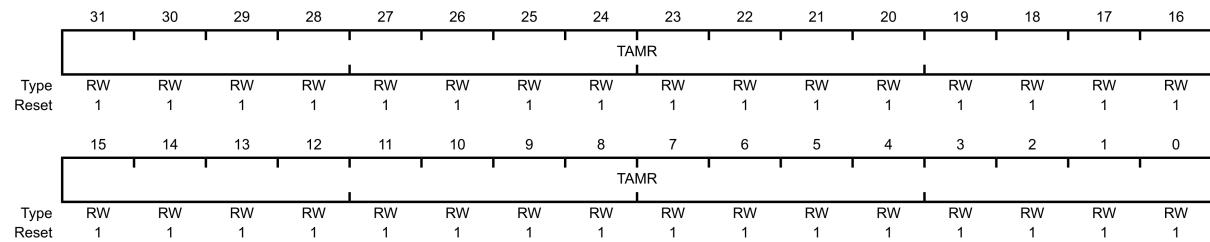
- 16/32-bit Timer 0 base: 0x4003.0000
- 16/32-bit Timer 1 base: 0x4003.1000
- 16/32-bit Timer 2 base: 0x4003.2000
- 16/32-bit Timer 3 base: 0x4003.3000
- 16/32-bit Timer 4 base: 0x4003.4000
- 16/32-bit Timer 5 base: 0x4003.5000
- 32/64-bit Wide Timer 0 base: 0x4003.6000
- 32/64-bit Wide Timer 1 base: 0x4003.7000
- 32/64-bit Wide Timer 2 base: 0x4004.C000
- 32/64-bit Wide Timer 3 base: 0x4004.D000
- 32/64-bit Wide Timer 4 base: 0x4004.E000
- 32/64-bit Wide Timer 5 base: 0x4004.F000
- Offset 0x038
- Type RW, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:8	TAPSRH	RW	0x00	<p>GPTM Timer A Prescale High Byte</p> <p>The register loads this value on a write. A read returns the current value of the register.</p> <p>For the 16/32-bit GPTM, this field is reserved. For the 32/64-bit Wide GPTM, this field contains the upper 8-bits of the 16-bit prescaler.</p> <p>Refer to page 406 for more details.</p>
7:0	TAPSR	RW	0x00	<p>GPTM Timer A Prescale</p> <p>The register loads this value on a write. A read returns the current value of the register.</p> <p>For the 16/32-bit GPTM, this field contains the entire 8-bit prescaler. For the 32/64-bit Wide GPTM, this field contains the lower 8-bits of the 16-bit prescaler.</p> <p>Refer to page 406 for more details.</p>

24.2.10.6 GPTM Timer A Match (GPTMTAMATCHR)

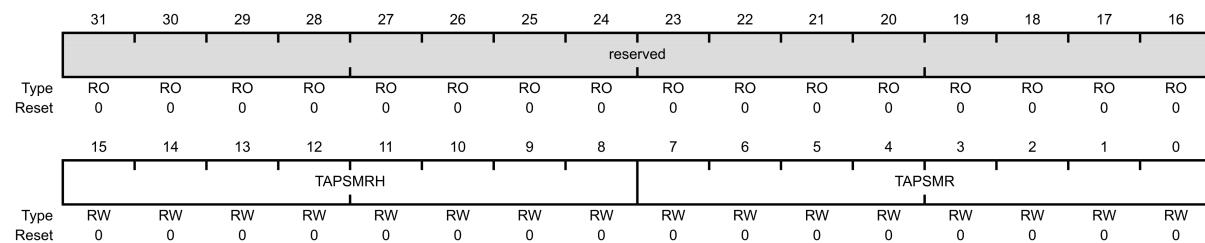
- 16/32-bit Timer 0 base: 0x4003.0000
- 16/32-bit Timer 1 base: 0x4003.1000
- 16/32-bit Timer 2 base: 0x4003.2000
- 16/32-bit Timer 3 base: 0x4003.3000
- 16/32-bit Timer 4 base: 0x4003.4000
- 16/32-bit Timer 5 base: 0x4003.5000
- 32/64-bit Wide Timer 0 base: 0x4003.6000
- 32/64-bit Wide Timer 1 base: 0x4003.7000
- 32/64-bit Wide Timer 2 base: 0x4004.C000
- 32/64-bit Wide Timer 3 base: 0x4004.D000
- 32/64-bit Wide Timer 4 base: 0x4004.E000
- 32/64-bit Wide Timer 5 base: 0x4004.F000
- Offset 0x030
- Type RW, reset 0xFFFF.FFFF



Bit/Field	Name	Type	Reset	Description
31:0	TAMR	RW	0xFFFF.FFFF	GPTM Timer A Match Register This value is compared to the GPTMTAR (page 424) register to determine match events.

24.2.10.7 GPTM Timer A Prescale Match (GPTMTAPMR)

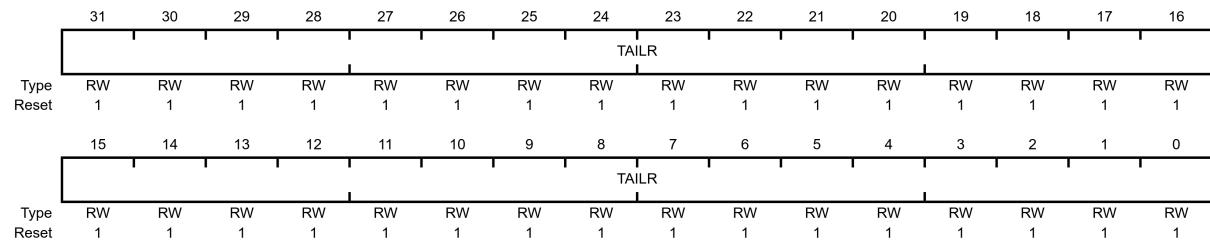
- 16/32-bit Timer 0 base: 0x4003.0000
- 16/32-bit Timer 1 base: 0x4003.1000
- 16/32-bit Timer 2 base: 0x4003.2000
- 16/32-bit Timer 3 base: 0x4003.3000
- 16/32-bit Timer 4 base: 0x4003.4000
- 16/32-bit Timer 5 base: 0x4003.5000
- 32/64-bit Wide Timer 0 base: 0x4003.6000
- 32/64-bit Wide Timer 1 base: 0x4003.7000
- 32/64-bit Wide Timer 2 base: 0x4004.C000
- 32/64-bit Wide Timer 3 base: 0x4004.D000
- 32/64-bit Wide Timer 4 base: 0x4004.E000
- 32/64-bit Wide Timer 5 base: 0x4004.F000
- Offset 0x040
- Type RW, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:8	TAPSRMH	RW	0x00	GPTM Timer A Prescale Match High Byte This value is used alongside GPTMTAMATCHR (page 421) to detect timer match events while using a prescaler. For the 16/32-bit GPTM, this field is reserved. For the 32/64-bit Wide GPTM, this field contains the upper 8-bits of the 16-bit prescale match value.
7:0	TAPSMR	RW	0x00	GPTM Timer A Prescale Match This value is used alongside GPTMTAMATCHR (page 421) to detect timer match events while using a prescaler. For the 16/32-bit GPTM, this field contains the entire 8-bit prescaler match value. For the 32/64-bit Wide GPTM, this field contains the lower 8-bits of the 16-bit prescaler match value.

24.2.10.8 GPTM Timer A Interval Load (GPTMTAILR)

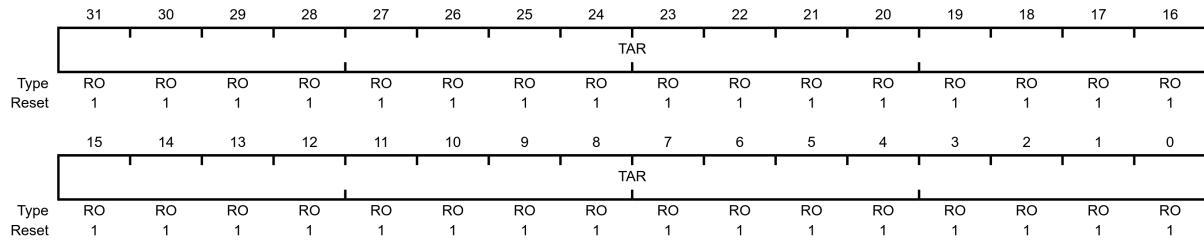
- 16/32-bit Timer 0 base: 0x4003.0000
- 16/32-bit Timer 1 base: 0x4003.1000
- 16/32-bit Timer 2 base: 0x4003.2000
- 16/32-bit Timer 3 base: 0x4003.3000
- 16/32-bit Timer 4 base: 0x4003.4000
- 16/32-bit Timer 5 base: 0x4003.5000
- 32/64-bit Wide Timer 0 base: 0x4003.6000
- 32/64-bit Wide Timer 1 base: 0x4003.7000
- 32/64-bit Wide Timer 2 base: 0x4004.C000
- 32/64-bit Wide Timer 3 base: 0x4004.D000
- 32/64-bit Wide Timer 4 base: 0x4004.E000
- 32/64-bit Wide Timer 5 base: 0x4004.F000
- Offset 0x028
- Type RW, reset 0xFFFF.FFFF



Bit/Field	Name	Type	Reset	Description
31:0	TAILR	RW	0xFFFF.FFFF	GPTM Timer A Interval Load Register Writing this field loads the counter for Timer A. A read returns the current value of GPTMTAILR (page 423).

24.2.10.9 GPTM Timer A (GPTMTAR)

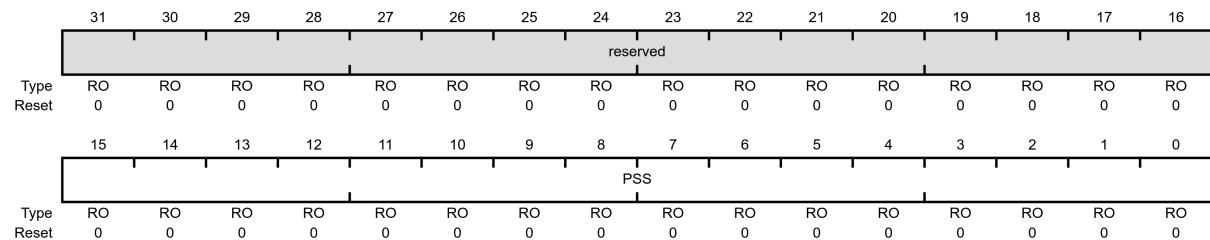
- 16/32-bit Timer 0 base: 0x4003.0000
- 16/32-bit Timer 1 base: 0x4003.1000
- 16/32-bit Timer 2 base: 0x4003.2000
- 16/32-bit Timer 3 base: 0x4003.3000
- 16/32-bit Timer 4 base: 0x4003.4000
- 16/32-bit Timer 5 base: 0x4003.5000
- 32/64-bit Wide Timer 0 base: 0x4003.6000
- 32/64-bit Wide Timer 1 base: 0x4003.7000
- 32/64-bit Wide Timer 2 base: 0x4004.C000
- 32/64-bit Wide Timer 3 base: 0x4004.D000
- 32/64-bit Wide Timer 4 base: 0x4004.E000
- 32/64-bit Wide Timer 5 base: 0x4004.F000
- Offset 0x048
- Type RO, reset 0xFFFF.FFFF



Bit/Field	Name	Type	Reset	Description
31:0	TAR	RO	0xFFFF.FFFF	<p>GPTM Timer A Register</p> <p>A read returns the current value of the GPTM Timer A Count Register (page 424), in all cases except for Input Edge Count and Time modes. In the Input Edge Count mode, this register contains the number of edges that have occurred. In the Input Edge Time mode, this register contains the time at which the last edge event took place.</p>

24.2.10.10 GPTM Timer A Prescale Snapshot (GPTMTAPS)

- 16/32-bit Timer 0 base: 0x4003.0000
- 16/32-bit Timer 1 base: 0x4003.1000
- 16/32-bit Timer 2 base: 0x4003.2000
- 16/32-bit Timer 3 base: 0x4003.3000
- 16/32-bit Timer 4 base: 0x4003.4000
- 16/32-bit Timer 5 base: 0x4003.5000
- 32/64-bit Wide Timer 0 base: 0x4003.6000
- 32/64-bit Wide Timer 1 base: 0x4003.7000
- 32/64-bit Wide Timer 2 base: 0x4004.C000
- 32/64-bit Wide Timer 3 base: 0x4004.D000
- 32/64-bit Wide Timer 4 base: 0x4004.E000
- 32/64-bit Wide Timer 5 base: 0x4004.F000
- Offset 0x05C
- Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	PSS	RO	0x0000	GPTM Timer A Prescaler Snapshot A read returns the current value of the GPTM Timer A Prescaler (page 425).

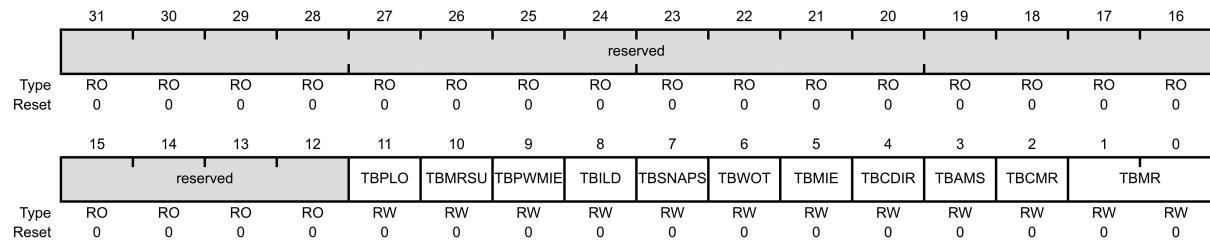
24.2.10.11 GPTM Timer A Prescale Value (GPTMTAPV)

- 16/32-bit Timer 0 base: 0x4003.0000
- 16/32-bit Timer 1 base: 0x4003.1000
- 16/32-bit Timer 2 base: 0x4003.2000
- 16/32-bit Timer 3 base: 0x4003.3000
- 16/32-bit Timer 4 base: 0x4003.4000
- 16/32-bit Timer 5 base: 0x4003.5000
- 32/64-bit Wide Timer 0 base: 0x4003.6000
- 32/64-bit Wide Timer 1 base: 0x4003.7000
- 32/64-bit Wide Timer 2 base: 0x4004.C000
- 32/64-bit Wide Timer 3 base: 0x4004.D000
- 32/64-bit Wide Timer 4 base: 0x4004.E000
- 32/64-bit Wide Timer 5 base: 0x4004.F000
- Offset 0x064
- Type RO, reset 0x0000.0000

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	PSV	RO	0x0000	GPTM Timer A Prescaler Value A read returns the current, free-running value of the Timer A prescaler.

24.2.10.12 GPTM Timer B Mode (GPTMTBMR)

- 16/32-bit Timer 0 base: 0x4003.0000
- 16/32-bit Timer 1 base: 0x4003.1000
- 16/32-bit Timer 2 base: 0x4003.2000
- 16/32-bit Timer 3 base: 0x4003.3000
- 16/32-bit Timer 4 base: 0x4003.4000
- 16/32-bit Timer 5 base: 0x4003.5000
- 32/64-bit Wide Timer 0 base: 0x4003.6000
- 32/64-bit Wide Timer 1 base: 0x4003.7000
- 32/64-bit Wide Timer 2 base: 0x4004.C000
- 32/64-bit Wide Timer 3 base: 0x4004.D000
- 32/64-bit Wide Timer 4 base: 0x4004.E000
- 32/64-bit Wide Timer 5 base: 0x4004.F000
- Offset 0x008
- Type RW, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description						
31:12	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
11	TBPL0	RW	0	<p>GPTM Timer B PWM Legacy Operation</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Legacy operation with CCP pin driven Low when the GPTMTAILR (page 423) is reloaded after the timer reaches 0.</td> </tr> <tr> <td>1</td> <td>CCP is driven High when the GPTMTAILR (page 423) is reloaded after the timer reaches 0.</td> </tr> </tbody> </table> <p>This bit is only valid in PWM mode.</p>	Value	Description	0	Legacy operation with CCP pin driven Low when the GPTMTAILR (page 423) is reloaded after the timer reaches 0.	1	CCP is driven High when the GPTMTAILR (page 423) is reloaded after the timer reaches 0.
Value	Description									
0	Legacy operation with CCP pin driven Low when the GPTMTAILR (page 423) is reloaded after the timer reaches 0.									
1	CCP is driven High when the GPTMTAILR (page 423) is reloaded after the timer reaches 0.									

Bit/Field	Name	Type	Reset	Description						
10	TBMRSU	RW	0	<p>GPTM Timer B Match Register Update</p> <table> <thead> <tr> <th data-bbox="774 316 854 350">Value</th> <th data-bbox="854 316 1029 350">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="774 361 798 395">0</td> <td data-bbox="878 361 1370 467">Update the GPTMTBMATCHR (page 434) register and the GPTMTBPR (page 433) register, if used, on the next cycle.</td> </tr> <tr> <td data-bbox="774 478 798 512">1</td> <td data-bbox="878 478 1370 624">Update the GPTMTBMATCHR (page 434) register and the GPTMTBPR (page 433) register, if used, on the next timeout.</td> </tr> </tbody> </table> <p>If the timer is disabled (TBEN is clear) when this bit is set, GPTMTBMATCHR (page 434) and GPTMTBPR (page 433) are updated when the timer is enabled. If the timer is stalled (TBSTALL is set), GPTMTBMATCHR (page 434) and GPTMTBPR (page 433) are updated according to the configuration of this bit.</p>	Value	Description	0	Update the GPTMTBMATCHR (page 434) register and the GPTMTBPR (page 433) register, if used, on the next cycle.	1	Update the GPTMTBMATCHR (page 434) register and the GPTMTBPR (page 433) register, if used, on the next timeout.
Value	Description									
0	Update the GPTMTBMATCHR (page 434) register and the GPTMTBPR (page 433) register, if used, on the next cycle.									
1	Update the GPTMTBMATCHR (page 434) register and the GPTMTBPR (page 433) register, if used, on the next timeout.									
9	TBPWMIE	RW	0	<p>GPTM Timer B PWM Interrupt Enable</p> <p>This bit enables interrupts in PWM mode on rising, falling, or both edges of the CCP output as defined by the TBEVENT field in the GPTMCTL (page 409) register.</p> <table> <thead> <tr> <th data-bbox="774 1170 854 1203">Value</th> <th data-bbox="854 1170 1029 1203">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="774 1215 798 1248">0</td> <td data-bbox="878 1215 1306 1248">Capture event interrupt is disabled.</td> </tr> <tr> <td data-bbox="774 1260 798 1293">1</td> <td data-bbox="878 1260 1306 1293">Capture event interrupt is enabled.</td> </tr> </tbody> </table> <p>This bit is only valid in PWM mode.</p>	Value	Description	0	Capture event interrupt is disabled.	1	Capture event interrupt is enabled.
Value	Description									
0	Capture event interrupt is disabled.									
1	Capture event interrupt is enabled.									

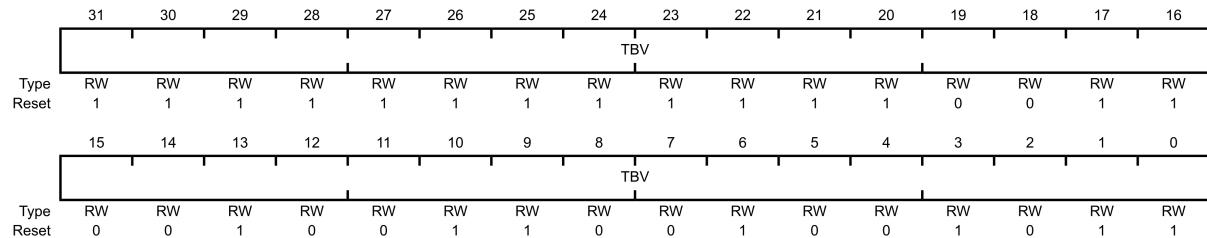
Bit/Field	Name	Type	Reset	Description						
8	TBILD	RW	0	<p>GPTM Timer B Interval Load Write</p> <table> <thead> <tr> <th data-bbox="774 316 854 350">Value</th> <th data-bbox="854 316 1029 350">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="774 361 798 395">0</td> <td data-bbox="878 361 1370 658">Update the GPTMTBR (page 437) and GPTMTBV (page 432) registers with the value in the GPTMTBILR (page 436) register on the next cycle. Also update the GPTMTBPS (page 438) and GPTMTBPV (page 439) registers with the value in the GPTMTBPR (page 433) register on the next cycle.</td> </tr> <tr> <td data-bbox="774 676 798 709">1</td> <td data-bbox="878 676 1370 972">Update the GPTMTBR (page 437) and GPTMTBV (page 432) registers with the value in the GPTMTBILR (page 436) register on the next timeout. Also update the GPTMTBPS (page 438) and GPTMTBPV (page 439) registers with the value in the GPTMTBPR (page 433) register on the next timeout.</td> </tr> </tbody> </table> <p>Note the state of this bit has no effect when counting up.</p> <p>The bit descriptions above apply if the timer is enabled and running. If the timer is disabled (TBEN is clear) when this bit is set, GPTMTBR (page 437), GPTMTBV (page 432), GPTMTBPS (page 438), and GPTMTBPV (page 439) are updated when the timer is enabled. If the timer is stalled (TBSTALL is set), GPTMTBR (page 437) and GPTMTBPS (page 438) are updated according to the configuration of this bit.</p>	Value	Description	0	Update the GPTMTBR (page 437) and GPTMTBV (page 432) registers with the value in the GPTMTBILR (page 436) register on the next cycle. Also update the GPTMTBPS (page 438) and GPTMTBPV (page 439) registers with the value in the GPTMTBPR (page 433) register on the next cycle.	1	Update the GPTMTBR (page 437) and GPTMTBV (page 432) registers with the value in the GPTMTBILR (page 436) register on the next timeout. Also update the GPTMTBPS (page 438) and GPTMTBPV (page 439) registers with the value in the GPTMTBPR (page 433) register on the next timeout.
Value	Description									
0	Update the GPTMTBR (page 437) and GPTMTBV (page 432) registers with the value in the GPTMTBILR (page 436) register on the next cycle. Also update the GPTMTBPS (page 438) and GPTMTBPV (page 439) registers with the value in the GPTMTBPR (page 433) register on the next cycle.									
1	Update the GPTMTBR (page 437) and GPTMTBV (page 432) registers with the value in the GPTMTBILR (page 436) register on the next timeout. Also update the GPTMTBPS (page 438) and GPTMTBPV (page 439) registers with the value in the GPTMTBPR (page 433) register on the next timeout.									
7	TBSNAPS	RW	0	<p>GPTM Timer B Snap-Shot Mode</p> <table> <thead> <tr> <th data-bbox="774 1529 854 1563">Value</th> <th data-bbox="854 1529 1029 1563">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="774 1574 798 1608">0</td> <td data-bbox="878 1574 1211 1608">Snap-shot mode is disabled.</td> </tr> <tr> <td data-bbox="774 1626 798 1659">1</td> <td data-bbox="878 1626 1370 1922">If Timer B is configured in the periodic mode, the actual free-running value of Timer B is loaded at the time-out event into the GPTM Timer B (GPTMTBR) (page 437) register. If the timer prescaler is used, the prescaler snapshot is loaded into the GPTM Timer B (GPTMTBPR) (page 433).</td> </tr> </tbody> </table>	Value	Description	0	Snap-shot mode is disabled.	1	If Timer B is configured in the periodic mode, the actual free-running value of Timer B is loaded at the time-out event into the GPTM Timer B (GPTMTBR) (page 437) register. If the timer prescaler is used, the prescaler snapshot is loaded into the GPTM Timer B (GPTMTBPR) (page 433).
Value	Description									
0	Snap-shot mode is disabled.									
1	If Timer B is configured in the periodic mode, the actual free-running value of Timer B is loaded at the time-out event into the GPTM Timer B (GPTMTBR) (page 437) register. If the timer prescaler is used, the prescaler snapshot is loaded into the GPTM Timer B (GPTMTBPR) (page 433).									

Bit/Field	Name	Type	Reset	Description						
6	TBWOT	RW	0	<p>GPTM Timer B Wait-on-Trigger</p> <table> <thead> <tr> <th data-bbox="774 316 838 350">Value</th> <th data-bbox="838 316 1029 350">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="774 361 798 395">0</td> <td data-bbox="838 361 1362 428">Timer B begins counting as soon as it is enabled.</td> </tr> <tr> <td data-bbox="774 451 798 485">1</td> <td data-bbox="838 451 1370 702">If Timer B is enabled (TBEN is set in the GPTMCTL (page 409) register), Timer B does not begin counting until it receives a trigger from the timer in the previous position in the daisy chain. This function is valid for one-shot, periodic, and PWM modes.</td> </tr> </tbody> </table>	Value	Description	0	Timer B begins counting as soon as it is enabled.	1	If Timer B is enabled (TBEN is set in the GPTMCTL (page 409) register), Timer B does not begin counting until it receives a trigger from the timer in the previous position in the daisy chain. This function is valid for one-shot, periodic, and PWM modes.
Value	Description									
0	Timer B begins counting as soon as it is enabled.									
1	If Timer B is enabled (TBEN is set in the GPTMCTL (page 409) register), Timer B does not begin counting until it receives a trigger from the timer in the previous position in the daisy chain. This function is valid for one-shot, periodic, and PWM modes.									
5	TBMIE	RW	0	<p>GPTM Timer B Match Interrupt Enable</p> <table> <thead> <tr> <th data-bbox="774 795 838 828">Value</th> <th data-bbox="838 795 1029 828">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="774 840 798 873">0</td> <td data-bbox="838 840 1314 907">The match interrupt is disabled for match events.</td> </tr> <tr> <td data-bbox="774 929 798 963">1</td> <td data-bbox="838 929 1370 1080">An interrupt is generated when the match value in the GPTMTBMR (page 427) register is reached in the one-shot and periodic modes.</td> </tr> </tbody> </table> <p>Note: Clearing the TBMIE bit in the GPTMTBMR (page 427) register prevents assertion of μDMA or ADC requests generated on a match event. Even if the TBTODMAEN bit is set in the GPTMDMAEV register or the TBTOADCEN bit is set in the GPTMADCEV register, a μDMA or ADC match trigger is not sent to the μDMA or ADC, respectively, when the TBMIE bit is clear.</p>	Value	Description	0	The match interrupt is disabled for match events.	1	An interrupt is generated when the match value in the GPTMTBMR (page 427) register is reached in the one-shot and periodic modes.
Value	Description									
0	The match interrupt is disabled for match events.									
1	An interrupt is generated when the match value in the GPTMTBMR (page 427) register is reached in the one-shot and periodic modes.									
4	TBCDIR	RW	0	<p>GPTM Timer B Count Direction</p> <table> <thead> <tr> <th data-bbox="774 1610 838 1644">Value</th> <th data-bbox="838 1610 1029 1644">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="774 1655 798 1688">0</td> <td data-bbox="838 1655 1171 1688">The timer counts down.</td> </tr> <tr> <td data-bbox="774 1711 798 1745">1</td> <td data-bbox="838 1711 1370 1778">The timer counts up. When counting up, the timer starts from a value of 0x0.</td> </tr> </tbody> </table> <p>When in PWM or RTC mode, the status of this bit is ignored. PWM mode always counts down and RTC mode always counts up.</p>	Value	Description	0	The timer counts down.	1	The timer counts up. When counting up, the timer starts from a value of 0x0.
Value	Description									
0	The timer counts down.									
1	The timer counts up. When counting up, the timer starts from a value of 0x0.									

Bit/Field	Name	Type	Reset	Description										
3	TBAMS	RW	0	<p>GPTM Timer B Alternate Mode Select</p> <p>The TBAMS values are defined as follows:</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Capture or compare mode is enabled.</td> </tr> <tr> <td>1</td> <td>PWM mode is enabled.</td> </tr> </tbody> </table> <p>Note: To enable PWM mode, you must also clear the TBCMR bit and configure the TBMR field to 0x1 or 0x2.</p>	Value	Description	0	Capture or compare mode is enabled.	1	PWM mode is enabled.				
Value	Description													
0	Capture or compare mode is enabled.													
1	PWM mode is enabled.													
2	TBCMR	RW	0	<p>GPTM Timer B Capture Mode</p> <p>The TBCMR values are defined as follows:</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Edge-Count mode</td> </tr> <tr> <td>1</td> <td>Edge-Time mode</td> </tr> </tbody> </table>	Value	Description	0	Edge-Count mode	1	Edge-Time mode				
Value	Description													
0	Edge-Count mode													
1	Edge-Time mode													
1:0	TBMR	RW	0x0	<p>GPTM Timer B Mode</p> <p>The TBMR values are defined as follows:</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Reserved</td> </tr> <tr> <td>0x1</td> <td>One-Shot Timer mode</td> </tr> <tr> <td>0x2</td> <td>Periodic Timer mode</td> </tr> <tr> <td>0x3</td> <td>Capture mode</td> </tr> </tbody> </table> <p>The timer mode is based on the timer configuration defined by bits 2:0 in the GPTMCFG (page 407) register.</p>	Value	Description	0x0	Reserved	0x1	One-Shot Timer mode	0x2	Periodic Timer mode	0x3	Capture mode
Value	Description													
0x0	Reserved													
0x1	One-Shot Timer mode													
0x2	Periodic Timer mode													
0x3	Capture mode													

24.2.10.13 GPTM Timer B Value (GPTMTBV)

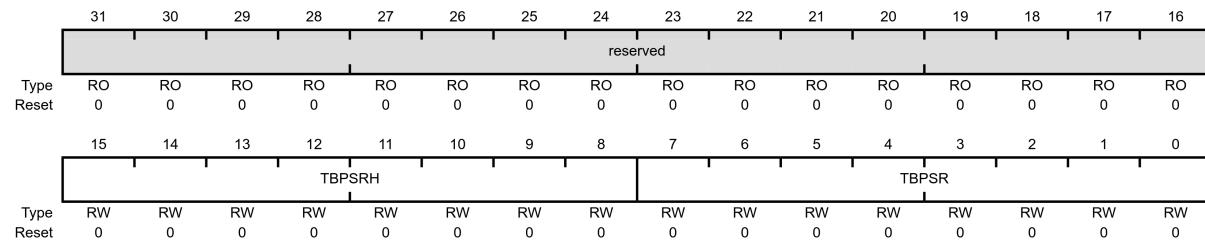
- 16/32-bit Timer 0 base: 0x4003.0000
- 16/32-bit Timer 1 base: 0x4003.1000
- 16/32-bit Timer 2 base: 0x4003.2000
- 16/32-bit Timer 3 base: 0x4003.3000
- 16/32-bit Timer 4 base: 0x4003.4000
- 16/32-bit Timer 5 base: 0x4003.5000
- 32/64-bit Wide Timer 0 base: 0x4003.6000
- 32/64-bit Wide Timer 1 base: 0x4003.7000
- 32/64-bit Wide Timer 2 base: 0x4004.C000
- 32/64-bit Wide Timer 3 base: 0x4004.D000
- 32/64-bit Wide Timer 4 base: 0x4004.E000
- 32/64-bit Wide Timer 5 base: 0x4004.F000
- Offset 0x054
- Type RW, reset -



Bit/Field	Name	Type	Reset	Description
31:0	TBV	RW	0x0000.FFFF (for 16/32-bit) 0xFFFF.FFFF (for 32/64-bit)	<p>GPTM Timer B Value</p> <p>A read returns the current, free-running value of Timer A in all modes. When written, the value written into this register is loaded into the GPTMTAR (page 424) register on the next clock cycle.</p> <p>Note: In 16-bit mode, only the lower 16-bits of the GPTMTBV (page 432) register can be written with a new value. Writes to the prescaler bits have no effect.</p>

24.2.10.14 GPTM Timer B Prescale (GPTMTBPR)

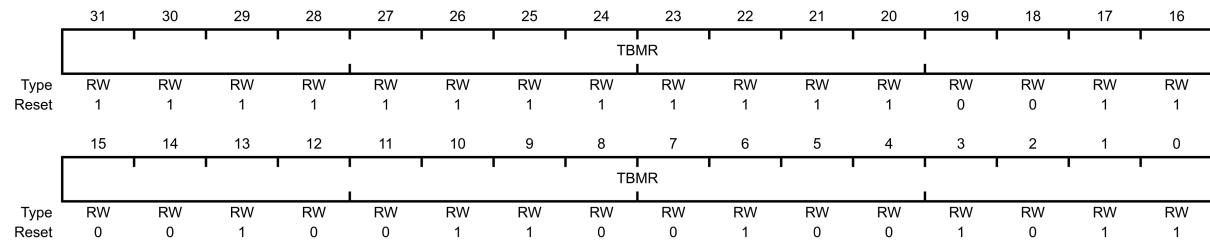
- 16/32-bit Timer 0 base: 0x4003.0000
- 16/32-bit Timer 1 base: 0x4003.1000
- 16/32-bit Timer 2 base: 0x4003.2000
- 16/32-bit Timer 3 base: 0x4003.3000
- 16/32-bit Timer 4 base: 0x4003.4000
- 16/32-bit Timer 5 base: 0x4003.5000
- 32/64-bit Wide Timer 0 base: 0x4003.6000
- 32/64-bit Wide Timer 1 base: 0x4003.7000
- 32/64-bit Wide Timer 2 base: 0x4004.C000
- 32/64-bit Wide Timer 3 base: 0x4004.D000
- 32/64-bit Wide Timer 4 base: 0x4004.E000
- 32/64-bit Wide Timer 5 base: 0x4004.F000
- Offset 0x03C
- Type RW, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:8	TBPSRH	RW	0x00	<p>GPTM Timer B Prescale High Byte</p> <p>The register loads this value on a write. A read returns the current value of the register.</p> <p>For the 16/32-bit GPTM, this field is reserved. For the 32/64-bit Wide GPTM, this field contains the upper 8-bits of the 16-bit prescaler.</p> <p>Refer to page 406 for more details.</p>
7:0	TBPSR	RW	0x00	<p>GPTM Timer B Prescale</p> <p>The register loads this value on a write. A read returns the current value of this register.</p> <p>For the 16/32-bit GPTM, this field contains the entire 8-bit prescaler. For the 32/64-bit Wide GPTM, this field contains the lower 8-bits of the 16-bit prescaler.</p> <p>Refer to page 406 for more details.</p>

24.2.10.15 GPTM Timer B Match (GPTMTBMATCHR)

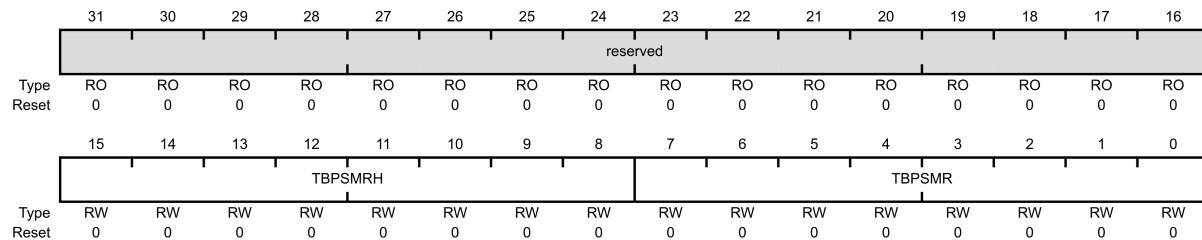
- 16/32-bit Timer 0 base: 0x4003.0000
- 16/32-bit Timer 1 base: 0x4003.1000
- 16/32-bit Timer 2 base: 0x4003.2000
- 16/32-bit Timer 3 base: 0x4003.3000
- 16/32-bit Timer 4 base: 0x4003.4000
- 16/32-bit Timer 5 base: 0x4003.5000
- 32/64-bit Wide Timer 0 base: 0x4003.6000
- 32/64-bit Wide Timer 1 base: 0x4003.7000
- 32/64-bit Wide Timer 2 base: 0x4004.C000
- 32/64-bit Wide Timer 3 base: 0x4004.D000
- 32/64-bit Wide Timer 4 base: 0x4004.E000
- 32/64-bit Wide Timer 5 base: 0x4004.F000
- Offset 0x034
- Type RW, reset -



Bit/Field	Name	Type	Reset	Description
31:0	TBMR	RW	0x0000.FFFF (for 16/32-bit) 0xFFFF.FFFF (for 32/64-bit)	GPTM Timer B Match Register This value is compared to the GPTMTBR (page 437) register to determine match events.

24.2.10.16 GPTM Timer B Prescale Match (GPTMTBPMR)

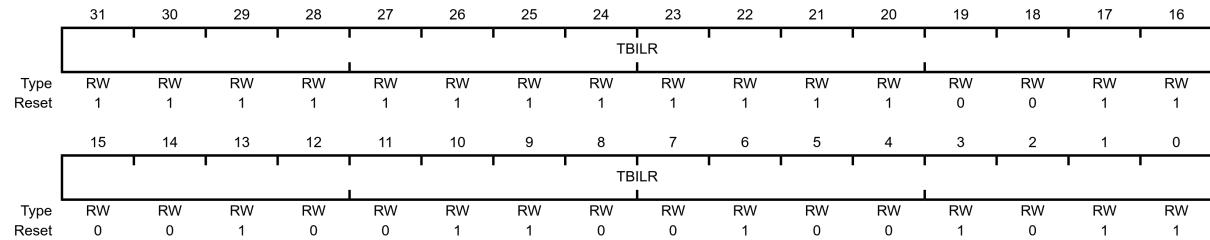
- 16/32-bit Timer 0 base: 0x4003.0000
- 16/32-bit Timer 1 base: 0x4003.1000
- 16/32-bit Timer 2 base: 0x4003.2000
- 16/32-bit Timer 3 base: 0x4003.3000
- 16/32-bit Timer 4 base: 0x4003.4000
- 16/32-bit Timer 5 base: 0x4003.5000
- 32/64-bit Wide Timer 0 base: 0x4003.6000
- 32/64-bit Wide Timer 1 base: 0x4003.7000
- 32/64-bit Wide Timer 2 base: 0x4004.C000
- 32/64-bit Wide Timer 3 base: 0x4004.D000
- 32/64-bit Wide Timer 4 base: 0x4004.E000
- 32/64-bit Wide Timer 5 base: 0x4004.F000
- Offset 0x044
- Type RW, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:8	TBPSRMH	RW	0x00	GPTM Timer B Prescale Match High Byte This value is used alongside GPTMTBPMR (page 435) to detect timer match events while using a prescaler. For the 16/32-bit GPTM, this field is reserved. For the 32/64-bit Wide GPTM, this field contains the upper 8-bits of the 16-bit prescale match value.
7:0	TBPSMR	RW	0x00	GPTM Timer B Prescale Match This value is used alongside GPTMTBPMR (page 435) to detect timer match events while using a prescaler. For the 16/32-bit GPTM, this field contains the entire 8-bit prescaler match value. For the 32/64-bit Wide GPTM, this field contains the lower 8-bits of the 16-bit prescaler match value.

24.2.10.17 GPTM Timer B Interval Load (GPTMTBILR)

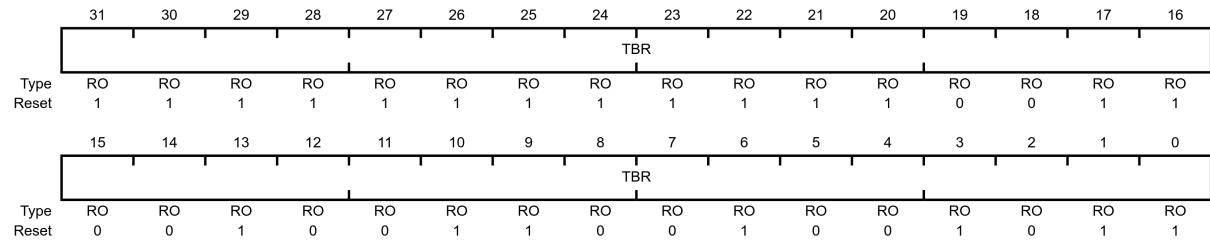
- 16/32-bit Timer 0 base: 0x4003.0000
- 16/32-bit Timer 1 base: 0x4003.1000
- 16/32-bit Timer 2 base: 0x4003.2000
- 16/32-bit Timer 3 base: 0x4003.3000
- 16/32-bit Timer 4 base: 0x4003.4000
- 16/32-bit Timer 5 base: 0x4003.5000
- 32/64-bit Wide Timer 0 base: 0x4003.6000
- 32/64-bit Wide Timer 1 base: 0x4003.7000
- 32/64-bit Wide Timer 2 base: 0x4004.C000
- 32/64-bit Wide Timer 3 base: 0x4004.D000
- 32/64-bit Wide Timer 4 base: 0x4004.E000
- 32/64-bit Wide Timer 5 base: 0x4004.F000
- Offset 0x02C
- Type RW, reset -



Bit/Field	Name	Type	Reset	Description
31:0	TBILR	RW	0x0000.FFFF (for 16/32-bit) 0xFFFF.FFFF (for 32/64-bit)	GPTM Timer B Interval Load Register Writing this field loads the counter for Timer B. A read returns the current value of GPTMTBILR (page 436) . When a 16/32-bit GPTM is in 32-bit mode, writes are ignored, and reads return the current value of GPTMTBILR (page 436) .

24.2.10.18 GPTM Timer B (GPTMTBR)

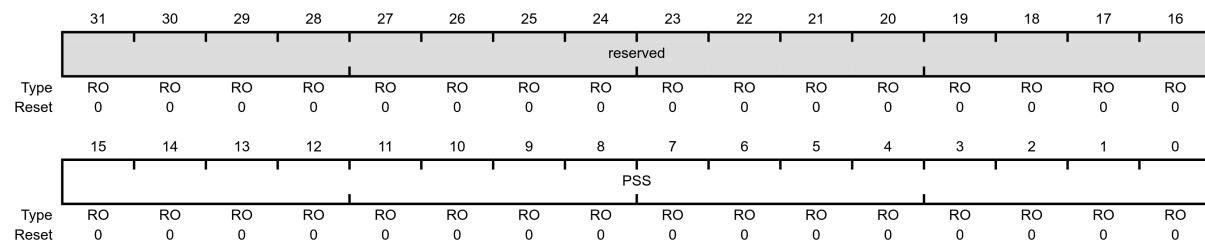
- 16/32-bit Timer 0 base: 0x4003.0000
- 16/32-bit Timer 1 base: 0x4003.1000
- 16/32-bit Timer 2 base: 0x4003.2000
- 16/32-bit Timer 3 base: 0x4003.3000
- 16/32-bit Timer 4 base: 0x4003.4000
- 16/32-bit Timer 5 base: 0x4003.5000
- 32/64-bit Wide Timer 0 base: 0x4003.6000
- 32/64-bit Wide Timer 1 base: 0x4003.7000
- 32/64-bit Wide Timer 2 base: 0x4004.C000
- 32/64-bit Wide Timer 3 base: 0x4004.D000
- 32/64-bit Wide Timer 4 base: 0x4004.E000
- 32/64-bit Wide Timer 5 base: 0x4004.F000
- Offset 0x04C
- Type RO, reset -



Bit/Field	Name	Type	Reset	Description
31:0	TBR	RO	0x0000.FFFF (for 16/32-bit) 0xFFFF.FFFF (for 32/64-bit)	GPTM Timer B Register A read returns the current value of the GPTM Timer B Count Register (page 437), in all cases except for Input Edge Count and Time modes. In the Input Edge Count mode, this register contains the number of edges that have occurred. In the Input Edge Time mode, this register contains the time at which the last edge event took place.

24.2.10.19 GPTM Timer B Prescale Snapshot (GPTMTBPS)

- 16/32-bit Timer 0 base: 0x4003.0000
- 16/32-bit Timer 1 base: 0x4003.1000
- 16/32-bit Timer 2 base: 0x4003.2000
- 16/32-bit Timer 3 base: 0x4003.3000
- 16/32-bit Timer 4 base: 0x4003.4000
- 16/32-bit Timer 5 base: 0x4003.5000
- 32/64-bit Wide Timer 0 base: 0x4003.6000
- 32/64-bit Wide Timer 1 base: 0x4003.7000
- 32/64-bit Wide Timer 2 base: 0x4004.C000
- 32/64-bit Wide Timer 3 base: 0x4004.D000
- 32/64-bit Wide Timer 4 base: 0x4004.E000
- 32/64-bit Wide Timer 5 base: 0x4004.F000
- Offset 0x060
- Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	PSS	RO	0x0000	GPTM Timer B Prescaler Snapshot A read returns the current value of the GPTM Timer B Prescaler (page 438).

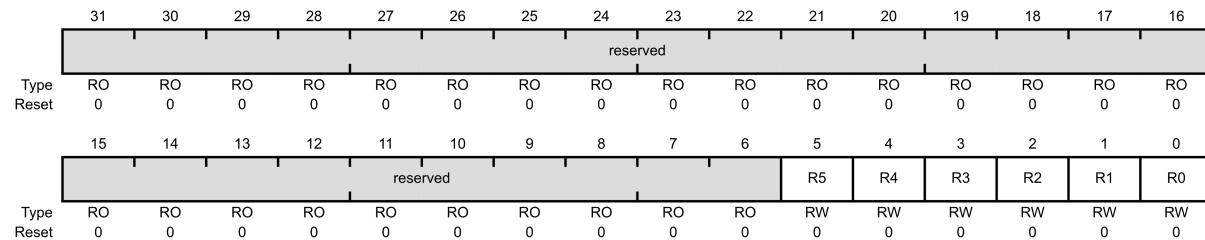
24.2.10.20 GPTM Timer B Prescale Value (GPTMTBPV)

- 16/32-bit Timer 0 base: 0x4003.0000
- 16/32-bit Timer 1 base: 0x4003.1000
- 16/32-bit Timer 2 base: 0x4003.2000
- 16/32-bit Timer 3 base: 0x4003.3000
- 16/32-bit Timer 4 base: 0x4003.4000
- 16/32-bit Timer 5 base: 0x4003.5000
- 32/64-bit Wide Timer 0 base: 0x4003.6000
- 32/64-bit Wide Timer 1 base: 0x4003.7000
- 32/64-bit Wide Timer 2 base: 0x4004.C000
- 32/64-bit Wide Timer 3 base: 0x4004.D000
- 32/64-bit Wide Timer 4 base: 0x4004.E000
- 32/64-bit Wide Timer 5 base: 0x4004.F000
- Offset 0x068
- Type RO, reset 0x0000.0000

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	PSV	RO	0x0000	GPTM Timer B Prescaler Value A read returns the current, free-running value of the Timer B prescaler.

24.2.10.21 16/32-Bit General-Purpose Timer Run Mode Clock Gating Control (RCGCTIMER)

- Base 0x400F.E000
- Offset 0x604
- Type RW, reset 0x0000.0000

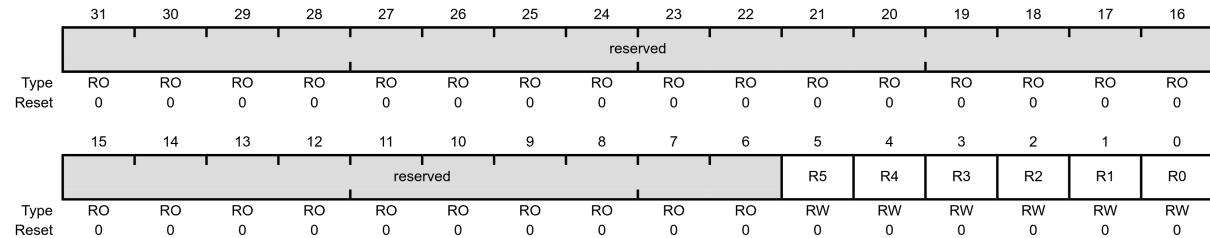


Bit/Field	Name	Type	Reset	Description						
31:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
5	R5	RW	0	<p>16/32-Bit General-Purpose Timer 5 Run Mode Clock Gating Control</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>16/32-bit general-purpose timer module 5 is disabled.</td> </tr> <tr> <td>1</td> <td>Enable and provide a clock to 16/32-bit general-purpose timer module 5 in Run mode.</td> </tr> </tbody> </table>	Value	Description	0	16/32-bit general-purpose timer module 5 is disabled.	1	Enable and provide a clock to 16/32-bit general-purpose timer module 5 in Run mode.
Value	Description									
0	16/32-bit general-purpose timer module 5 is disabled.									
1	Enable and provide a clock to 16/32-bit general-purpose timer module 5 in Run mode.									
4	R4	RW	0	<p>16/32-Bit General-Purpose Timer 4 Run Mode Clock Gating Control</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>16/32-bit general-purpose timer module 4 is disabled.</td> </tr> <tr> <td>1</td> <td>Enable and provide a clock to 16/32-bit general-purpose timer module 4 in Run mode.</td> </tr> </tbody> </table>	Value	Description	0	16/32-bit general-purpose timer module 4 is disabled.	1	Enable and provide a clock to 16/32-bit general-purpose timer module 4 in Run mode.
Value	Description									
0	16/32-bit general-purpose timer module 4 is disabled.									
1	Enable and provide a clock to 16/32-bit general-purpose timer module 4 in Run mode.									
3	R3	RW	0	<p>16/32-Bit General-Purpose Timer 3 Run Mode Clock Gating Control</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>16/32-bit general-purpose timer module 3 is disabled.</td> </tr> <tr> <td>1</td> <td>Enable and provide a clock to 16/32-bit general-purpose timer module 3 in Run mode.</td> </tr> </tbody> </table>	Value	Description	0	16/32-bit general-purpose timer module 3 is disabled.	1	Enable and provide a clock to 16/32-bit general-purpose timer module 3 in Run mode.
Value	Description									
0	16/32-bit general-purpose timer module 3 is disabled.									
1	Enable and provide a clock to 16/32-bit general-purpose timer module 3 in Run mode.									

Bit/Field	Name	Type	Reset	Description						
2	R2	RW	0	<p>16/32-Bit General-Purpose Timer 2 Run Mode Clock Gating Control</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>16/32-bit general-purpose timer module 2 is disabled.</td> </tr> <tr> <td>1</td> <td>Enable and provide a clock to 16/32-bit general-purpose timer module 2 in Run mode.</td> </tr> </tbody> </table>	Value	Description	0	16/32-bit general-purpose timer module 2 is disabled.	1	Enable and provide a clock to 16/32-bit general-purpose timer module 2 in Run mode.
Value	Description									
0	16/32-bit general-purpose timer module 2 is disabled.									
1	Enable and provide a clock to 16/32-bit general-purpose timer module 2 in Run mode.									
1	R1	RW	0	<p>16/32-Bit General-Purpose Timer 1 Run Mode Clock Gating Control</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>16/32-bit general-purpose timer module 1 is disabled.</td> </tr> <tr> <td>1</td> <td>Enable and provide a clock to 16/32-bit general-purpose timer module 1 in Run mode.</td> </tr> </tbody> </table>	Value	Description	0	16/32-bit general-purpose timer module 1 is disabled.	1	Enable and provide a clock to 16/32-bit general-purpose timer module 1 in Run mode.
Value	Description									
0	16/32-bit general-purpose timer module 1 is disabled.									
1	Enable and provide a clock to 16/32-bit general-purpose timer module 1 in Run mode.									
0	R0	RW	0	<p>16/32-Bit General-Purpose Timer 0 Run Mode Clock Gating Control</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>16/32-bit general-purpose timer module 0 is disabled.</td> </tr> <tr> <td>1</td> <td>Enable and provide a clock to 16/32-bit general-purpose timer module 0 in Run mode.</td> </tr> </tbody> </table>	Value	Description	0	16/32-bit general-purpose timer module 0 is disabled.	1	Enable and provide a clock to 16/32-bit general-purpose timer module 0 in Run mode.
Value	Description									
0	16/32-bit general-purpose timer module 0 is disabled.									
1	Enable and provide a clock to 16/32-bit general-purpose timer module 0 in Run mode.									

24.2.10.22 32/64-Bit Wide General-Purpose Timer Run Mode Clock Gating Control (RCGCWTIMER)

- Base 0x400F.E000
- Offset 0x65C
- Type RW, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description						
31:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
5	R5	RW	0	<p>32/64-Bit Wide General-Purpose Timer 5 Run Mode Clock Gating Control</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>32/64-bit wide general-purpose timer module 5 is disabled.</td></tr> <tr> <td>1</td><td>Enable and provide a clock to 32/64-bit wide general-purpose timer module 5 in Run mode.</td></tr> </tbody> </table>	Value	Description	0	32/64-bit wide general-purpose timer module 5 is disabled.	1	Enable and provide a clock to 32/64-bit wide general-purpose timer module 5 in Run mode.
Value	Description									
0	32/64-bit wide general-purpose timer module 5 is disabled.									
1	Enable and provide a clock to 32/64-bit wide general-purpose timer module 5 in Run mode.									
4	R4	RW	0	<p>32/64-Bit Wide General-Purpose Timer 4 Run Mode Clock Gating Control</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>32/64-bit wide general-purpose timer module 4 is disabled.</td></tr> <tr> <td>1</td><td>Enable and provide a clock to 32/64-bit wide general-purpose timer module 4 in Run mode.</td></tr> </tbody> </table>	Value	Description	0	32/64-bit wide general-purpose timer module 4 is disabled.	1	Enable and provide a clock to 32/64-bit wide general-purpose timer module 4 in Run mode.
Value	Description									
0	32/64-bit wide general-purpose timer module 4 is disabled.									
1	Enable and provide a clock to 32/64-bit wide general-purpose timer module 4 in Run mode.									
3	R3	RW	0	<p>32/64-Bit Wide General-Purpose Timer 3 Run Mode Clock Gating Control</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>32/64-bit wide general-purpose timer module 3 is disabled.</td></tr> <tr> <td>1</td><td>Enable and provide a clock to 32/64-bit wide general-purpose timer module 3 in Run mode.</td></tr> </tbody> </table>	Value	Description	0	32/64-bit wide general-purpose timer module 3 is disabled.	1	Enable and provide a clock to 32/64-bit wide general-purpose timer module 3 in Run mode.
Value	Description									
0	32/64-bit wide general-purpose timer module 3 is disabled.									
1	Enable and provide a clock to 32/64-bit wide general-purpose timer module 3 in Run mode.									

Bit/Field	Name	Type	Reset	Description						
2	R2	RW	0	<p>32/64-Bit Wide General-Purpose Timer 2 Run Mode Clock Gating Control</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>32/64-bit wide general-purpose timer module 2 is disabled.</td> </tr> <tr> <td>1</td> <td>Enable and provide a clock to 32/64-bit wide general-purpose timer module 2 in Run mode.</td> </tr> </tbody> </table>	Value	Description	0	32/64-bit wide general-purpose timer module 2 is disabled.	1	Enable and provide a clock to 32/64-bit wide general-purpose timer module 2 in Run mode.
Value	Description									
0	32/64-bit wide general-purpose timer module 2 is disabled.									
1	Enable and provide a clock to 32/64-bit wide general-purpose timer module 2 in Run mode.									
1	R1	RW	0	<p>32/64-Bit Wide General-Purpose Timer 1 Run Mode Clock Gating Control</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>32/64-bit wide general-purpose timer module 1 is disabled.</td> </tr> <tr> <td>1</td> <td>Enable and provide a clock to 32/64-bit wide general-purpose timer module 1 in Run mode.</td> </tr> </tbody> </table>	Value	Description	0	32/64-bit wide general-purpose timer module 1 is disabled.	1	Enable and provide a clock to 32/64-bit wide general-purpose timer module 1 in Run mode.
Value	Description									
0	32/64-bit wide general-purpose timer module 1 is disabled.									
1	Enable and provide a clock to 32/64-bit wide general-purpose timer module 1 in Run mode.									
0	R0	RW	0	<p>32/64-Bit Wide General-Purpose Timer 0 Run Mode Clock Gating Control</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>32/64-bit wide general-purpose timer module 0 is disabled.</td> </tr> <tr> <td>1</td> <td>Enable and provide a clock to 32/64-bit wide general-purpose timer module 0 in Run mode.</td> </tr> </tbody> </table>	Value	Description	0	32/64-bit wide general-purpose timer module 0 is disabled.	1	Enable and provide a clock to 32/64-bit wide general-purpose timer module 0 in Run mode.
Value	Description									
0	32/64-bit wide general-purpose timer module 0 is disabled.									
1	Enable and provide a clock to 32/64-bit wide general-purpose timer module 0 in Run mode.									

24.2.10.23 GPTM Interrupt Clear (GPTMICR)

- 16/32-bit Timer 0 base: 0x4003.0000
- 16/32-bit Timer 1 base: 0x4003.1000
- 16/32-bit Timer 2 base: 0x4003.2000
- 16/32-bit Timer 3 base: 0x4003.3000
- 16/32-bit Timer 4 base: 0x4003.4000
- 16/32-bit Timer 5 base: 0x4003.5000
- 32/64-bit Wide Timer 0 base: 0x4003.6000
- 32/64-bit Wide Timer 1 base: 0x4003.7000
- 32/64-bit Wide Timer 2 base: 0x4004.C000
- 32/64-bit Wide Timer 3 base: 0x4004.D000
- 32/64-bit Wide Timer 4 base: 0x4004.E000
- 32/64-bit Wide Timer 5 base: 0x4004.F000
- Offset 0x024
- Type W1C, reset 0x0000.0000

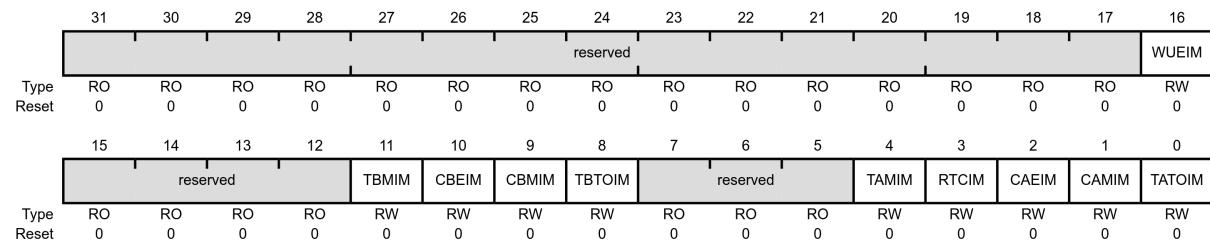
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
	reserved																
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	reserved				TBMCINT	CBECINT	CBMCINT	TBTCCINT	reserved				TAMCINT	RTCCINT	CAECINT	CAMCINT	TATOCINT
Type	RO	RO	RO	RO	W1C	W1C	W1C	W1C	RO	RO	RO	RO	W1C	W1C	W1C	W1C	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Bit/Field	Name	Type	Reset	Description
31:17	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	WUECINT	RW	0	32/64-Bit Wide GPTM Write Update Error Interrupt Clear Writing a 1 to this bit clears the WUERIS bit in the GPTMRIS (page 449) register and the WUEMIS bit in the GPTMMIS (page 453) register.
15:12	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	TBMCINT	W1C	0	GPTM Timer B Match Interrupt Clear Writing a 1 to this bit clears the TBMRIS bit in the GPTMRIS (page 449) register and the TBMMIS bit in the GPTMMIS (page 453) register.
10	CBECINT	W1C	0	GPTM Timer B Capture Mode Event Interrupt Clear Writing a 1 to this bit clears the CBERIS bit in the GPTMRIS (page 449) register and the CBEMIS bit in the GPTMMIS (page 453) register.

Bit/Field	Name	Type	Reset	Description
9	CBMCINT	W1C	0	GPTM Timer B Capture Mode Match Interrupt Clear Writing a 1 to this bit clears the CBMRIS bit in the GPTMRIS (page 449) register and the CBMMIS bit in the GPTMMIS (page 453) register.
8	TBTOCINT	W1C	0	GPTM Timer B Time-Out Interrupt Clear Writing a 1 to this bit clears the TBTORIS bit in the GPTMRIS (page 449) register and the TBTOMIS bit in the GPTMMIS (page 453) register.
7:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	TAMCINT	W1C	0	GPTM Timer A Match Interrupt Clear Writing a 1 to this bit clears the TAMRIS bit in the GPTMRIS (page 449) register and the TAMMIS bit in the GPTMMIS (page 453) register.
3	RTCCINT	W1C	0	GPTM RTC Interrupt Clear Writing a 1 to this bit clears the RTCRIS bit in the GPTMRIS (page 449) register and the RTCMIS bit in the GPTMMIS (page 453) register.
2	CAECINT	W1C	0	GPTM Timer A Capture Mode Event Interrupt Clear Writing a 1 to this bit clears the CAERIS bit in the GPTMRIS (page 449) register and the CAEMIS bit in the GPTMMIS (page 453) register.
1	CAMCINT	W1C	0	GPTM Timer A Capture Mode Match Interrupt Clear Writing a 1 to this bit clears the CAMRIS bit in the GPTMRIS (page 449) register and the CAMMIS bit in the GPTMMIS (page 453) register.
0	TATOCINT	W1C	0	GPTM Timer A Time-Out Raw Interrupt Writing a 1 to this bit clears the TATORIS bit in the GPTMRIS (page 449) register and the TATOMIS bit in the GPTMMIS (page 453) register.

24.2.10.24 GPTM Interrupt Mask (GPTMIMR)

- 16/32-bit Timer 0 base: 0x4003.0000
- 16/32-bit Timer 1 base: 0x4003.1000
- 16/32-bit Timer 2 base: 0x4003.2000
- 16/32-bit Timer 3 base: 0x4003.3000
- 16/32-bit Timer 4 base: 0x4003.4000
- 16/32-bit Timer 5 base: 0x4003.5000
- 32/64-bit Wide Timer 0 base: 0x4003.6000
- 32/64-bit Wide Timer 1 base: 0x4003.7000
- 32/64-bit Wide Timer 2 base: 0x4004.C000
- 32/64-bit Wide Timer 3 base: 0x4004.D000
- 32/64-bit Wide Timer 4 base: 0x4004.E000
- 32/64-bit Wide Timer 5 base: 0x4004.F000
- Offset 0x018
- Type RW, reset 0x0000.0000



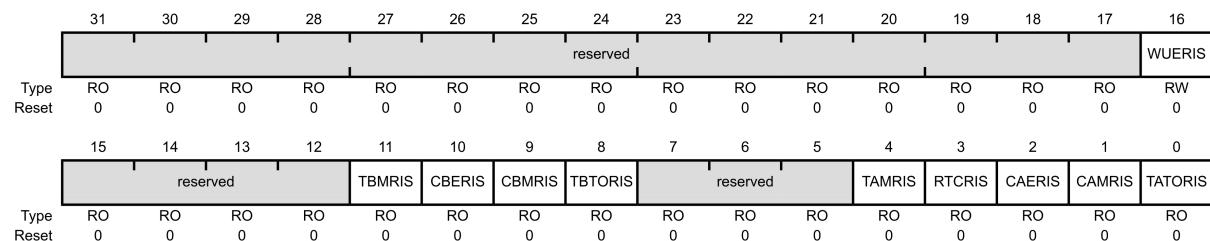
Bit/Field	Name	Type	Reset	Description
31:17	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	WUEIM	RW	0	32/64-Bit Wide GPTM Write Update Error Interrupt Mask The WUEIM values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
15:12	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	TBMIM	RW	0	GPTM Timer B Match Interrupt Mask The TBMIM values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.

Bit/Field	Name	Type	Reset	Description						
10	CBEIM	RW	0	<p>GPTM Timer B Capture Mode Event Interrupt Mask</p> <p>The CBEIM values are defined as follows:</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Interrupt is disabled.</td> </tr> <tr> <td>1</td> <td>Interrupt is enabled.</td> </tr> </tbody> </table>	Value	Description	0	Interrupt is disabled.	1	Interrupt is enabled.
Value	Description									
0	Interrupt is disabled.									
1	Interrupt is enabled.									
9	CBMIM	RW	0	<p>GPTM Timer B Capture Mode Match Interrupt Mask</p> <p>The CBMIM values are defined as follows:</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Interrupt is disabled.</td> </tr> <tr> <td>1</td> <td>Interrupt is enabled.</td> </tr> </tbody> </table>	Value	Description	0	Interrupt is disabled.	1	Interrupt is enabled.
Value	Description									
0	Interrupt is disabled.									
1	Interrupt is enabled.									
8	TBTOIM	RW	0	<p>GPTM Timer B Time-Out Interrupt Mask</p> <p>The TBTOIM values are defined as follows:</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Interrupt is disabled.</td> </tr> <tr> <td>1</td> <td>Interrupt is enabled.</td> </tr> </tbody> </table>	Value	Description	0	Interrupt is disabled.	1	Interrupt is enabled.
Value	Description									
0	Interrupt is disabled.									
1	Interrupt is enabled.									
7:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
4	TAMIM	RW	0	<p>GPTM Timer A Match Interrupt Mask</p> <p>The TAMIM values are defined as follows:</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Interrupt is disabled.</td> </tr> <tr> <td>1</td> <td>Interrupt is enabled.</td> </tr> </tbody> </table>	Value	Description	0	Interrupt is disabled.	1	Interrupt is enabled.
Value	Description									
0	Interrupt is disabled.									
1	Interrupt is enabled.									
3	RTCIM	RW	0	<p>GPTM RTC Interrupt Mask</p> <p>The RTCIM values are defined as follows:</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Interrupt is disabled.</td> </tr> <tr> <td>1</td> <td>Interrupt is enabled.</td> </tr> </tbody> </table>	Value	Description	0	Interrupt is disabled.	1	Interrupt is enabled.
Value	Description									
0	Interrupt is disabled.									
1	Interrupt is enabled.									
2	CAEIM	RW	0	<p>GPTM Timer A Capture Mode Event Interrupt Mask</p> <p>The CAEIM values are defined as follows:</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Interrupt is disabled.</td> </tr> <tr> <td>1</td> <td>Interrupt is enabled.</td> </tr> </tbody> </table>	Value	Description	0	Interrupt is disabled.	1	Interrupt is enabled.
Value	Description									
0	Interrupt is disabled.									
1	Interrupt is enabled.									

Bit/Field	Name	Type	Reset	Description						
1	CAMIM	RW	0	<p>GPTM Timer A Capture Mode Match Interrupt Mask</p> <p>The CAMIM values are defined as follows:</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Interrupt is disabled.</td> </tr> <tr> <td>1</td> <td>Interrupt is enabled.</td> </tr> </tbody> </table>	Value	Description	0	Interrupt is disabled.	1	Interrupt is enabled.
Value	Description									
0	Interrupt is disabled.									
1	Interrupt is enabled.									
0	TATOIM	RW	0	<p>GPTM Timer A Time-Out Interrupt Mask</p> <p>The TATOIM values are defined as follows:</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Interrupt is disabled.</td> </tr> <tr> <td>1</td> <td>Interrupt is enabled.</td> </tr> </tbody> </table>	Value	Description	0	Interrupt is disabled.	1	Interrupt is enabled.
Value	Description									
0	Interrupt is disabled.									
1	Interrupt is enabled.									

24.2.10.25 GPTM Raw Interrupt Status (GPTMRIS)

- 16/32-bit Timer 0 base: 0x4003.0000
- 16/32-bit Timer 1 base: 0x4003.1000
- 16/32-bit Timer 2 base: 0x4003.2000
- 16/32-bit Timer 3 base: 0x4003.3000
- 16/32-bit Timer 4 base: 0x4003.4000
- 16/32-bit Timer 5 base: 0x4003.5000
- 32/64-bit Wide Timer 0 base: 0x4003.6000
- 32/64-bit Wide Timer 1 base: 0x4003.7000
- 32/64-bit Wide Timer 2 base: 0x4004.C000
- 32/64-bit Wide Timer 3 base: 0x4004.D000
- 32/64-bit Wide Timer 4 base: 0x4004.E000
- 32/64-bit Wide Timer 5 base: 0x4004.F000
- Offset 0x01C
- Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description						
31:17	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
16	WUERIS	RW	0	<p>32/64-Bit Wide GPTM Write Update Error Raw Interrupt Status</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No error.</td> </tr> <tr> <td>1</td> <td>Either a Timer A register or a Timer B register was written twice in a row or a Timer A register was written before the corresponding Timer B register was written.</td> </tr> </tbody> </table>	Value	Description	0	No error.	1	Either a Timer A register or a Timer B register was written twice in a row or a Timer A register was written before the corresponding Timer B register was written.
Value	Description									
0	No error.									
1	Either a Timer A register or a Timer B register was written twice in a row or a Timer A register was written before the corresponding Timer B register was written.									
15:12	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						

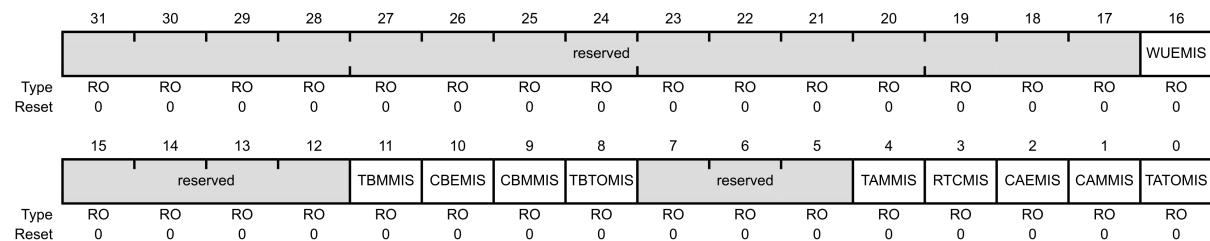
Bit/Field	Name	Type	Reset	Description						
11	TBMRIS	RO	0	<p>GPTM Timer B Match Raw Interrupt</p> <table> <thead> <tr> <th data-bbox="684 316 747 350">Value</th> <th data-bbox="747 316 938 350">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="684 361 716 395">0</td> <td data-bbox="747 361 1256 395">The match value has not been reached.</td> </tr> <tr> <td data-bbox="684 406 716 440">1</td> <td data-bbox="747 406 1367 624">The TBMRIS bit is set in the GPTMTBMR (page 427) register, and the match values in the GPTMTBMATCHR (page 434) and (optionally) GPTMTBPMR (page 435) registers have been reached when configured in one-shot or periodic mode.</td> </tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the TBMCINT bit in the GPTMICR (page 444) register.</p>	Value	Description	0	The match value has not been reached.	1	The TBMRIS bit is set in the GPTMTBMR (page 427) register, and the match values in the GPTMTBMATCHR (page 434) and (optionally) GPTMTBPMR (page 435) registers have been reached when configured in one-shot or periodic mode.
Value	Description									
0	The match value has not been reached.									
1	The TBMRIS bit is set in the GPTMTBMR (page 427) register, and the match values in the GPTMTBMATCHR (page 434) and (optionally) GPTMTBPMR (page 435) registers have been reached when configured in one-shot or periodic mode.									
10	CBERIS	RO	0	<p>GPTM Timer B Capture Mode Event Raw Interrupt</p> <table> <thead> <tr> <th data-bbox="684 810 747 844">Value</th> <th data-bbox="747 810 938 844">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="684 855 716 923">0</td> <td data-bbox="747 855 1319 923">The capture mode event for Timer B has not occurred.</td> </tr> <tr> <td data-bbox="684 945 716 1170">1</td> <td data-bbox="747 945 1367 1170">A capture mode event has occurred for Timer B. This interrupt asserts when the subtimer is configured in Input Edge-Time mode or when configured in PWM mode with the PWM interrupt enabled by setting the TBPWMIE bit in the GPTMTBMR (page 427) register.</td> </tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the CBECINT bit in the GPTMICR (page 444) register.</p>	Value	Description	0	The capture mode event for Timer B has not occurred.	1	A capture mode event has occurred for Timer B. This interrupt asserts when the subtimer is configured in Input Edge-Time mode or when configured in PWM mode with the PWM interrupt enabled by setting the TBPWMIE bit in the GPTMTBMR (page 427) register.
Value	Description									
0	The capture mode event for Timer B has not occurred.									
1	A capture mode event has occurred for Timer B. This interrupt asserts when the subtimer is configured in Input Edge-Time mode or when configured in PWM mode with the PWM interrupt enabled by setting the TBPWMIE bit in the GPTMTBMR (page 427) register.									
9	CBMRIS	RO	0	<p>GPTM Timer B Capture Mode Match Raw Interrupt</p> <table> <thead> <tr> <th data-bbox="684 1356 747 1390">Value</th> <th data-bbox="747 1356 938 1390">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="684 1401 716 1468">0</td> <td data-bbox="747 1401 1335 1468">The capture mode match for Timer B has not occurred.</td> </tr> <tr> <td data-bbox="684 1491 716 1760">1</td> <td data-bbox="747 1491 1367 1760">The capture mode match has occurred for Timer B. This interrupt asserts when the values in the GPTMTBR (page 437) and GPTMTBPR (page 433) match the values in the GPTMTBMATCHR (page 434) and GPTMTBPMR (page 435) when configured in Input Edge-Time mode.</td> </tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the CBMCINT bit in the GPTMICR (page 444) register.</p>	Value	Description	0	The capture mode match for Timer B has not occurred.	1	The capture mode match has occurred for Timer B. This interrupt asserts when the values in the GPTMTBR (page 437) and GPTMTBPR (page 433) match the values in the GPTMTBMATCHR (page 434) and GPTMTBPMR (page 435) when configured in Input Edge-Time mode.
Value	Description									
0	The capture mode match for Timer B has not occurred.									
1	The capture mode match has occurred for Timer B. This interrupt asserts when the values in the GPTMTBR (page 437) and GPTMTBPR (page 433) match the values in the GPTMTBMATCHR (page 434) and GPTMTBPMR (page 435) when configured in Input Edge-Time mode.									

Bit/Field	Name	Type	Reset	Description						
8	TBTORIS	RO	0	<p>GPTM Timer B Time-Out Raw Interrupt</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Timer B has not timed out.</td></tr> <tr> <td>1</td><td>Timer B has timed out. This interrupt is asserted when a one-shot or periodic mode timer reaches its count limit (0 or the value loaded into GPTMTBILR (page 436), depending on the count direction).</td></tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the TBT0CINT bit in the GPTMICR (page 444) register.</p>	Value	Description	0	Timer B has not timed out.	1	Timer B has timed out. This interrupt is asserted when a one-shot or periodic mode timer reaches its count limit (0 or the value loaded into GPTMTBILR (page 436), depending on the count direction).
Value	Description									
0	Timer B has not timed out.									
1	Timer B has timed out. This interrupt is asserted when a one-shot or periodic mode timer reaches its count limit (0 or the value loaded into GPTMTBILR (page 436), depending on the count direction).									
7:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
4	TAMRIS	RO	0	<p>GPTM Timer A Match Raw Interrupt</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The match value has not been reached.</td></tr> <tr> <td>1</td><td>The TAMIE bit is set in the GPTMTAMR (page 414) register, and the match value in the GPTMTAMATCHR (page 421) and (optionally) GPTMTAPMR (page 422) registers have been reached when configured in one-shot or periodic mode.</td></tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the TAMCINT bit in the GPTMICR (page 444) register.</p>	Value	Description	0	The match value has not been reached.	1	The TAMIE bit is set in the GPTMTAMR (page 414) register, and the match value in the GPTMTAMATCHR (page 421) and (optionally) GPTMTAPMR (page 422) registers have been reached when configured in one-shot or periodic mode.
Value	Description									
0	The match value has not been reached.									
1	The TAMIE bit is set in the GPTMTAMR (page 414) register, and the match value in the GPTMTAMATCHR (page 421) and (optionally) GPTMTAPMR (page 422) registers have been reached when configured in one-shot or periodic mode.									
3	RTCRIS	RO	0	<p>GPTM RTC Raw Interrupt</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The RTC event has not occurred.</td></tr> <tr> <td>1</td><td>The RTC event has occurred.</td></tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the RTCCINT bit in the GPTMICR (page 444) register.</p>	Value	Description	0	The RTC event has not occurred.	1	The RTC event has occurred.
Value	Description									
0	The RTC event has not occurred.									
1	The RTC event has occurred.									

Bit/Field	Name	Type	Reset	Description						
2	CAERIS	RO	0	<p>GPTM Timer A Capture Mode Event Raw Interrupt</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The capture mode event for Timer A has not occurred.</td></tr> <tr> <td>1</td><td>A capture mode event has occurred for Timer A. This interrupt asserts when the subtimer is configured in Input Edge-Time mode or when configured in PWM mode with the PWM interrupt enabled by setting the TAPWMIE bit in the GPTMTAMR (page 414) register.</td></tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the CAECINT bit in the GPTMICR (page 444) register.</p>	Value	Description	0	The capture mode event for Timer A has not occurred.	1	A capture mode event has occurred for Timer A. This interrupt asserts when the subtimer is configured in Input Edge-Time mode or when configured in PWM mode with the PWM interrupt enabled by setting the TAPWMIE bit in the GPTMTAMR (page 414) register.
Value	Description									
0	The capture mode event for Timer A has not occurred.									
1	A capture mode event has occurred for Timer A. This interrupt asserts when the subtimer is configured in Input Edge-Time mode or when configured in PWM mode with the PWM interrupt enabled by setting the TAPWMIE bit in the GPTMTAMR (page 414) register.									
1	CAMRIS	RO	0	<p>GPTM Timer A Capture Mode Match Raw Interrupt</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The capture mode match for Timer A has not occurred.</td></tr> <tr> <td>1</td><td>This interrupt asserts when the values in the GPTMTAR (page 424) and GPTMTAPR (page 420) match the values in the GPTMTAMATCHR (page 421) and GPTMTAPMR (page 422) when configured in Input Edge-Time mode.</td></tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the CAMCINT bit in the GPTMICR (page 444) register.</p>	Value	Description	0	The capture mode match for Timer A has not occurred.	1	This interrupt asserts when the values in the GPTMTAR (page 424) and GPTMTAPR (page 420) match the values in the GPTMTAMATCHR (page 421) and GPTMTAPMR (page 422) when configured in Input Edge-Time mode.
Value	Description									
0	The capture mode match for Timer A has not occurred.									
1	This interrupt asserts when the values in the GPTMTAR (page 424) and GPTMTAPR (page 420) match the values in the GPTMTAMATCHR (page 421) and GPTMTAPMR (page 422) when configured in Input Edge-Time mode.									
0	TATORIS	RO	0	<p>GPTM Timer A Time-Out Raw Interrupt</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Timer A has not timed out.</td></tr> <tr> <td>1</td><td>Timer A has timed out. This interrupt is asserted when a one-shot or periodic mode timer reaches its count limit (0 or the value loaded into GPTMTAILR (page 423), depending on the count direction).</td></tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the TAT0CINT bit in the GPTMICR (page 444) register.</p>	Value	Description	0	Timer A has not timed out.	1	Timer A has timed out. This interrupt is asserted when a one-shot or periodic mode timer reaches its count limit (0 or the value loaded into GPTMTAILR (page 423), depending on the count direction).
Value	Description									
0	Timer A has not timed out.									
1	Timer A has timed out. This interrupt is asserted when a one-shot or periodic mode timer reaches its count limit (0 or the value loaded into GPTMTAILR (page 423), depending on the count direction).									

24.2.10.26 GPTM Masked Interrupt Status (GPTMMIS)

- 16/32-bit Timer 0 base: 0x4003.0000
- 16/32-bit Timer 1 base: 0x4003.1000
- 16/32-bit Timer 2 base: 0x4003.2000
- 16/32-bit Timer 3 base: 0x4003.3000
- 16/32-bit Timer 4 base: 0x4003.4000
- 16/32-bit Timer 5 base: 0x4003.5000
- 32/64-bit Wide Timer 0 base: 0x4003.6000
- 32/64-bit Wide Timer 1 base: 0x4003.7000
- 32/64-bit Wide Timer 2 base: 0x4004.C000
- 32/64-bit Wide Timer 3 base: 0x4004.D000
- 32/64-bit Wide Timer 4 base: 0x4004.E000
- 32/64-bit Wide Timer 5 base: 0x4004.F000
- Offset 0x020
- Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description						
31:17	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
16	WUEMIS	RO	0	<p>32/64-Bit Wide GPTM Write Update Error Masked Interrupt Status</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>An unmasked Write Update Error has not occurred.</td> </tr> <tr> <td>1</td> <td>An unmasked Write Update Error has occurred.</td> </tr> </tbody> </table>	Value	Description	0	An unmasked Write Update Error has not occurred.	1	An unmasked Write Update Error has occurred.
Value	Description									
0	An unmasked Write Update Error has not occurred.									
1	An unmasked Write Update Error has occurred.									
15:12	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						

Bit/Field	Name	Type	Reset	Description						
11	TBMMIS	RO	0	<p>GPTM Timer B Match Masked Interrupt</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>A Timer B Mode Match interrupt has not occurred or is masked.</td> </tr> <tr> <td>1</td> <td>An unmasked Timer B Mode Match interrupt has occurred.</td> </tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the TBMCINT bit in the GPTMICR (page 444) register.</p>	Value	Description	0	A Timer B Mode Match interrupt has not occurred or is masked.	1	An unmasked Timer B Mode Match interrupt has occurred.
Value	Description									
0	A Timer B Mode Match interrupt has not occurred or is masked.									
1	An unmasked Timer B Mode Match interrupt has occurred.									
10	CBEMIS	RO	0	<p>GPTM Timer B Capture Mode Event Masked Interrupt</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>A Capture B event interrupt has not occurred or is masked.</td> </tr> <tr> <td>1</td> <td>An unmasked Capture B event interrupt has occurred.</td> </tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the CBECINT bit in the GPTMICR (page 444) register.</p>	Value	Description	0	A Capture B event interrupt has not occurred or is masked.	1	An unmasked Capture B event interrupt has occurred.
Value	Description									
0	A Capture B event interrupt has not occurred or is masked.									
1	An unmasked Capture B event interrupt has occurred.									
9	CBMMIS	RO	0	<p>GPTM Timer B Capture Mode Match Masked Interrupt</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>A Capture B Mode Match interrupt has not occurred or is masked.</td> </tr> <tr> <td>1</td> <td>An unmasked Capture B Match interrupt has occurred.</td> </tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the CBMCINT bit in the GPTMICR (page 444) register.</p>	Value	Description	0	A Capture B Mode Match interrupt has not occurred or is masked.	1	An unmasked Capture B Match interrupt has occurred.
Value	Description									
0	A Capture B Mode Match interrupt has not occurred or is masked.									
1	An unmasked Capture B Match interrupt has occurred.									
8	TBTOMIS	RO	0	<p>GPTM Timer B Time-Out Masked Interrupt</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>A Timer B Time-Out interrupt has not occurred or is masked.</td> </tr> <tr> <td>1</td> <td>An unmasked Timer B Time-Out interrupt has occurred.</td> </tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the TBT0CINT bit in the GPTMICR (page 444) register.</p>	Value	Description	0	A Timer B Time-Out interrupt has not occurred or is masked.	1	An unmasked Timer B Time-Out interrupt has occurred.
Value	Description									
0	A Timer B Time-Out interrupt has not occurred or is masked.									
1	An unmasked Timer B Time-Out interrupt has occurred.									
7:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						

Bit/Field	Name	Type	Reset	Description						
4	TAMMIS	RO	0	<p>GPTM Timer A Match Masked Interrupt</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>A Timer A Mode Match interrupt has not occurred or is masked.</td> </tr> <tr> <td>1</td> <td>An unmasked Timer A Mode Match interrupt has occurred.</td> </tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the TAMCINT bit in the GPTMICR (page 444) register.</p>	Value	Description	0	A Timer A Mode Match interrupt has not occurred or is masked.	1	An unmasked Timer A Mode Match interrupt has occurred.
Value	Description									
0	A Timer A Mode Match interrupt has not occurred or is masked.									
1	An unmasked Timer A Mode Match interrupt has occurred.									
3	RTCMIS	RO	0	<p>GPTM RTC Masked Interrupt</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>An RTC event interrupt has not occurred or is masked.</td> </tr> <tr> <td>1</td> <td>An unmasked RTC event interrupt has occurred.</td> </tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the RTCCINT bit in the GPTMICR (page 444) register.</p>	Value	Description	0	An RTC event interrupt has not occurred or is masked.	1	An unmasked RTC event interrupt has occurred.
Value	Description									
0	An RTC event interrupt has not occurred or is masked.									
1	An unmasked RTC event interrupt has occurred.									
2	CAEMIS	RO	0	<p>GPTM Timer A Capture Mode Event Masked Interrupt</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>A Capture A event interrupt has not occurred or is masked.</td> </tr> <tr> <td>1</td> <td>An unmasked Capture A event interrupt has occurred.</td> </tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the CAECINT bit in the GPTMICR (page 444) register.</p>	Value	Description	0	A Capture A event interrupt has not occurred or is masked.	1	An unmasked Capture A event interrupt has occurred.
Value	Description									
0	A Capture A event interrupt has not occurred or is masked.									
1	An unmasked Capture A event interrupt has occurred.									
1	CAMMIS	RO	0	<p>GPTM Timer A Capture Mode Match Masked Interrupt</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>A Capture A Mode Match interrupt has not occurred or is masked.</td> </tr> <tr> <td>1</td> <td>An unmasked Capture A Match interrupt has occurred.</td> </tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the CAMCINT bit in the GPTMICR (page 444) register.</p>	Value	Description	0	A Capture A Mode Match interrupt has not occurred or is masked.	1	An unmasked Capture A Match interrupt has occurred.
Value	Description									
0	A Capture A Mode Match interrupt has not occurred or is masked.									
1	An unmasked Capture A Match interrupt has occurred.									

Bit/Field	Name	Type	Reset	Description						
0	TATOMIS	RO	0	<p>GPTM Timer A Time-Out Masked Interrupt</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>A Timer A Time-Out interrupt has not occurred or is masked.</td> </tr> <tr> <td>1</td> <td>An unmasked Timer A Time-Out interrupt has occurred.</td> </tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the TATOCINT bit in the GPTMICR (page 444) register.</p>	Value	Description	0	A Timer A Time-Out interrupt has not occurred or is masked.	1	An unmasked Timer A Time-Out interrupt has occurred.
Value	Description									
0	A Timer A Time-Out interrupt has not occurred or is masked.									
1	An unmasked Timer A Time-Out interrupt has occurred.									

24.2.11 Initialisation and configuration

1. To use a GPTM, the appropriate TIMERn bit must be set in the **RCGCTIMER** ([page 440](#)) or **RCGCWTIMER** ([page 442](#)) register.
2. Enable the clock to the appropriate GPIO module via the **RCGCGPIO** ([page 133](#)) register.
3. Configure the PMCn fields in the **GPIOPCTL** ([page 158](#)) register to assign the CCP signals to the appropriate pins ([page 130](#)).

24.2.11.1 One-shot or periodic timer configuration

1. Ensure the timer is disabled, which means to ensure that the TnEN bit in the **GPTMCTL** ([page 409](#)) register is cleared before making any changes.
2. Write 0x0000.0000 to the **GPTM Configuration (GPTMCFG)** ([page 407](#)) register.
3. Configure the TnMR field in the **GPTM Timer n Mode (GPTMTnMR)** ([page 414](#)) register:
 - a. Write a value of 0x1 for One-Shot mode.
 - b. Write a value of 0x2 for Periodic mode.
4. Configure the TnSNAPS, TnWOT, TnMIE, and TnCDIR bits in the **GPTMTnMR** ([page 414](#)) register.
5. Load the start value into the **GPTM Timer n Interval Load (GPTMTnILR)** ([page 423](#)) register.
6. If interrupts are required, set the appropriate bits in the **GPTM Interrupt Mask (GPTMIMR)** ([page 446](#)).
7. Set the TnEN bit in the **GPTMCTL** ([page 409](#)) register to enable the timer and start counting.
8. Poll the **GPTMRIS** ([page 449](#)) register or wait for the interrupt to be generated (if enabled). In both cases, the status flags are cleared by writing a 1 to the appropriate bit of the **GPTM Interrupt Clear (GPTMICR)** ([page 444](#)) register.

In One-Shot mode, the timer stops counting after the time-out event. To re-enable the timer, repeat the sequence.

24.2.11.2 Real time clock mode

1. Ensure the timer is disabled, which means the TnEN bit is cleared, before making any changes.
2. If the timer has been operating in a different mode prior to this, clear any residual set bits in the **GPTM Timer n Mode (GPTMTnMR)** ([page 414](#)) register before reconfiguring.
3. Write 0x0000.0001 to the **GPTM Configuration Register (GPTMCFG)** ([page 407](#)) register.
4. Write the match value to the **GPTM Timer n Match (GPTMTnMATCHR)** ([page 421](#)) register.
5. Set or clear the RTCEN and TnSTALL bit in the **GPTM Control (GPTMCTL)** ([page 409](#)) register as needed.
6. If interrupts are required, set the RTCIM bit in the **GPTM Interrupt Mask (GPTMIMR)** ([page 446](#)) register.
7. Set the TAEN bit in the **GPTMCTL** ([page 409](#)) register to enable the timer and start counting.

When the timer count equals the value in the **GPTMTnMATCHR** ([page 421](#)) register, the GPTM asserts the RTCRIS bit in the **GPTMRIS** ([page 449](#)) register and continues counting until Timer A is disabled or a hardware reset occurs.

The interrupt is cleared by writing to the RTCCINT bit in the **GPTMICR** ([page 444](#)) register.

Note that if the **GPTMTnILR** ([page 423](#)) register is loaded with a new value, the timer begins counting at this new value and continues until it reaches 0xFFFF.FFFF, at which point it rolls over.

24.2.11.3 Input Edge Count mode

A timer is configured to the Input Edge Count mode by the follow sequence:

1. Ensure the timer is disabled, which means the TnEN bit is cleared, before making any changes.
2. Write 0x0000.0004 to the **GPTM Configuration (GPTMCFG)** (page 407) register.
3. In the **GPTM Timer n Mode (GPTMTnMR)** (page 414) register, write 0x0 to the TnCMR field and the 0x3 to the TnMR field.
4. Configure the type of events that the timer captures by writing to the TnEVENT field of the **GPTM Control (GPTMCTL)** (page 409) register.
5. Program registers according to the count direction:
 - In down-count mode, the **GPTMTnMATCHR** (page 421) and **GPTMTnPMR** (page 422) registers are configured so that the difference between the value in the **GPTMTnILR** (page 423) and the **GPTMTnPR** (page 420) registers and the **GPTMTnMATCHR** (page 421) and **GPTMTnPMR** (page 422) equals the number of edge events that must be counted.
 - In up-count mode, the timer counts from 0x0 to the value in the **GPTMTnMATCHR** (page 421) and **GPTMTnPMR** (page 422) registers. Note that when executing an up-count, the value of the **GPTMTnPR** (page 420) and **GPTMTnILR** (page 423) must be greater than the value of **GPTMTnPMR** (page 422) and **GPTMTnMATCHR** (page 421).
6. If interrupts are required, set the CnMIM bit in the **GPTM Interrupt Mask (GPTMIMR)** (page 446) register.
7. Set the TnEN bit in the **GPTMCTL** (page 409) register to enable the timer and begin waiting for edge events.
8. Poll the CnMRIS bit in the **GPTMRIS** (page 449) register or wait for the interrupt to be generated if enabled.

In both cases, the status flags are cleared by writing a 1 to the CnMCINT bit of the **GPTM Interrupt Clear (GPTMICR)** (page 444) register.

When counting down in Input Edge Count mode, the timer stops after the programmed number of edge events has been detected. To re-enable the timer, ensure that the TnEN bit is cleared and repeat steps 4 through 8.

24.2.11.4 Input Edge Time mode

A timer is configured to Input Edge Time mode by the following sequence:

1. Ensure the timer is disabled, which means the TnEN bit is cleared, before making any changes.
2. Write 0x0000.0004 to the **GPTM Configuration (GPTMCFG)** ([page 407](#)) register.
3. In the **GPTM Timer Mode (GPTMTnMR)** ([page 414](#)) register, write 0x1 to the TnCMR field and 0x3 to the TnMR field and select a count direction by programming the TnCDIR bit.
4. Configure the type of event that the timer captures by writing to the TnEVENT field of the **GPTM Control (GPTMCTL)** ([page 409](#)) register.
5. If a prescaler is to be used, write the prescale value to the **GPTM Timer n Interval Load (GPTMTnPR)** ([page 420](#)) register.
6. Load the timer start value into the **GPTM Timer n Interval Load (GPTMTnILR)** ([page 423](#)) register.
7. If interrupts are required, set the CnEIM bit in the **GPTM Interrupt Mask (GPTMIMR)** ([page 446](#)) register.
8. Set the TnEN bit in the **GPTM Control (GPTMCTL)** ([page 409](#)) register to enable the timer and start counting.
9. Poll the CnERIS bit in the **GPTMRIS** ([page 449](#)) register or wait for the interrupt to be generated (if enabled).

In both cases, the status flags are cleared by writing a 1 to the CnECINT bit of the **GPTM Interrupt Clear (GPTMICR)** ([page 444](#)) register. The time at which the event happened can be obtained by reading the **GPTM Timer n (GPTMTnR)** ([page 424](#)) register.

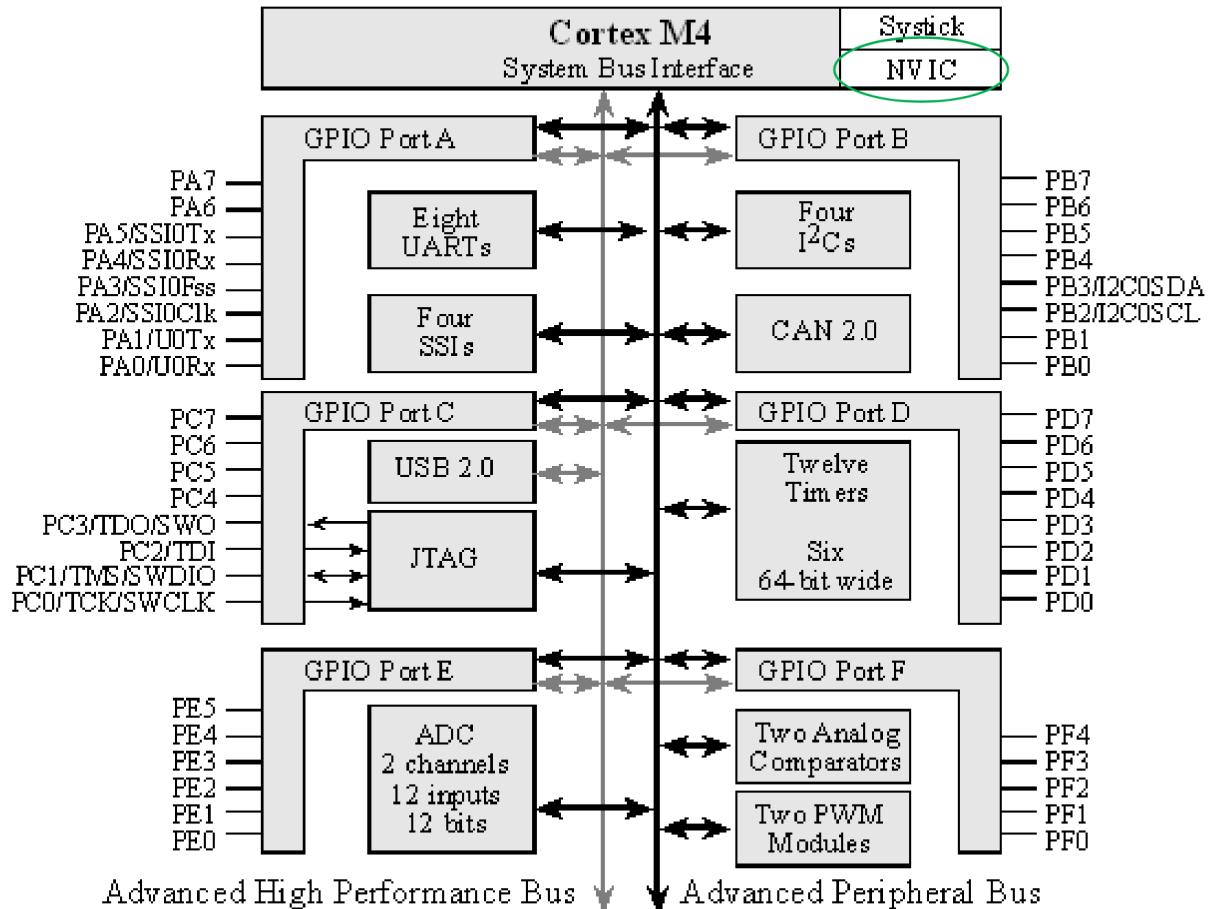
24.2.11.5 PWM mode

A timer is configured to PWM mode using the following sequence:

1. Ensure the timer is disabled, which means the TnEN bit is cleared, before making any changes.
2. Write 0x0000.0004 to the **GPTM Configuration (GPTMCFG)** ([page 407](#)) register.
3. In the **GPTM Timer n Mode (GPTMTnMR)** ([page 414](#)) register, set the TnAMS bit to 0x1, the TnCMR bit to 0x0, and the TnMR field to 0x2.
4. Configure the output state of the PWM signal, like whether or not it is inverted, in the TnPWML field of the **GPTM Control (GPTMCTL)** ([page 409](#)) register.
5. If a prescaler is to be used, write the prescale value to the **GPTM Timer n Prescale (GPTMTnPR)** ([page 420](#)) register.
6. If PWM interrupts are used, configure the interrupt condition in the TnEVENT field in the **GPTMCTL** ([page 409](#)) register and enable the interrupts by setting the TnPWMIE bit in the **GPTMTnMR** ([page 414](#)) register. Note that the edge detection interrupt behaviour is reversed when the PWM output is inverted.
7. Load the timer start value into the **GPTM Timer n Interval Load (GPTMTnILR)** ([page 423](#)) register.
8. Load the **GPTM Timer n Match (GPTMTnMATCHR)** ([page 421](#)) register with the match value.
9. Set the TnEN bit in the **GPTM Control (GPTMCTL)** ([page 409](#)) register to enable the timer and begin generation of the output PWM signal.

In PWM time mode, the timer continues running after the PWM signal has been generated. The PWM period can be adjusted at any time by writing to the **GPTMTnILR** ([page 423](#)) register, and the change takes effect on the next cycle after the write.

25 TM4C123GH6PM interrupts



- Controlled by the Nested Vectored Interrupt Controller (NVIC).
- Interrupt Request (IRQ) is an exception signaled by a peripheral or generated by a software request and fed through the NVIC. The latter is prioritised.
- All interrupts are asynchronous to instruction execution.
- In the system, peripherals use interrupts to communicate with the processor.

25.1 Interrupts

Vector Number	Interrupt Number (Bit in Interrupt Registers)	Vector Address or Offset	Description
0-15	-	0x0000.0000 - 0x0000.003C	Processor exceptions
16	0	0x0000.0040	GPIO Port A
17	1	0x0000.0044	GPIO Port B
18	2	0x0000.0048	GPIO Port C
19	3	0x0000.004C	GPIO Port D
20	4	0x0000.0050	GPIO Port E
21	5	0x0000.0054	UART0
22	6	0x0000.0058	UART1
23	7	0x0000.005C	SSIO
24	8	0x0000.0060	I ² C0
25	9	0x0000.0064	PWM0 Fault
26	10	0x0000.0068	PWM0 Generator 0
27	11	0x0000.006C	PWM0 Generator 1
28	12	0x0000.0070	PWM0 Generator 2
29	13	0x0000.0074	QEI0
30	14	0x0000.0078	ADC0 Sequence 0
31	15	0x0000.007C	ADC0 Sequence 1
32	16	0x0000.0080	ADC0 Sequence 2
33	17	0x0000.0084	ADC0 Sequence 3
34	18	0x0000.0088	Watchdog Timers 0 and 1
35	19	0x0000.008C	16/32-Bit Timer 0A
36	20	0x0000.0090	16/32-Bit Timer 0B
37	21	0x0000.0094	16/32-Bit Timer 1A
38	22	0x0000.0098	16/32-Bit Timer 1B
39	23	0x0000.009C	16/32-Bit Timer 2A
40	24	0x0000.00A0	16/32-Bit Timer 2B
41	25	0x0000.00A4	Analog Comparator 0
42	26	0x0000.00A8	Analog Comparator 1
43	27	-	Reserved
44	28	0x0000.00B0	System Control
45	29	0x0000.00B4	Flash Memory Control and EEPROM Control
46	30	0x0000.00B8	GPIO Port F
47-48	31-32	-	Reserved
49	33	0x0000.00C4	UART2
50	34	0x0000.00C8	SSI1

Vector Number	Interrupt Number (Bit in Interrupt Registers)	Vector Address or Offset	Description
51	35	0x0000.00CC	16/32-Bit Timer 3A
52	36	0x0000.00D0	16/32-Bit Timer 3B
53	37	0x0000.00D4	I ² C1
54	38	0x0000.00D8	QEI1
55	39	0x0000.00DC	CAN0
56	40	0x0000.00E0	CAN1
57-58	41-42	-	Reserved
59	43	0x0000.00EC	Hibernation Module
60	44	0x0000.00F0	USB
61	45	0x0000.00F4	PWM Generator 3
62	46	0x0000.00F8	μDMA Software
63	47	0x0000.00FC	μDMA Error
64	48	0x0000.0100	ADC1 Sequence 0
65	49	0x0000.0104	ADC1 Sequence 1
66	50	0x0000.0108	ADC1 Sequence 2
67	51	0x0000.010C	ADC1 Sequence 3
68-72	52-56	-	Reserved
73	57	0x0000.0124	SSI2
74	58	0x0000.0128	SSI3
75	59	0x0000.012C	UART3
76	60	0x0000.0130	UART4
77	61	0x0000.0134	UART5
78	62	0x0000.0138	UART6
79	63	0x0000.013C	UART7
80-83	64-67	0x0000.0140 - 0x0000.014C	Reserved
84	68	0x0000.0150	I ² C2
85	69	0x0000.0154	I ² C3
86	70	0x0000.0158	16/32-Bit Timer 4A
87	71	0x0000.015C	16/32-Bit Timer 4B
88-107	72-91	0x0000.0160 - 0x0000.01AC	Reserved
108	92	0x0000.01B0	16/32-Bit Timer 5A
109	93	0x0000.01B4	16/32-Bit Timer 5B
110	94	0x0000.01B8	32/64-Bit Timer 0A
111	95	0x0000.01BC	32/64-Bit Timer 0B
112	96	0x0000.01C0	32/64-Bit Timer 1A
113	97	0x0000.01C4	32/64-Bit Timer 1B
114	98	0x0000.01C8	32/64-Bit Timer 2A

Vector Number	Interrupt Number (Bit in Interrupt Registers)	Vector Address or Offset	Description
115	99	0x0000.01CC	32/64-Bit Timer 2B
116	100	0x0000.01D0	32/64-Bit Timer 3A
117	101	0x0000.01D4	32/64-Bit Timer 3B
118	102	0x0000.01D8	32/64-Bit Timer 4A
119	103	0x0000.01DC	32/64-Bit Timer 4B
120	104	0x0000.01E0	32/64-Bit Timer 5A
121	105	0x0000.01E4	32/64-Bit Timer 5B
122	106	0x0000.01E8	System Exception (imprecise)
123-149	107-133	-	Reserved
150	134	0x0000.0258	PWM1 Generator 0
151	135	0x0000.025C	PWM1 Generator 1
152	136	0x0000.0260	PWM1 Generator 2
153	137	0x0000.0264	PWM1 Generator 3
154	138	0x0000.0268	PWM1 Fault

25.2 Registers

25.2.1 Interrupt 0-31 Set Enable (EN0)

- Base 0xE000.E000
- Offset 0x100
- Type RW, reset 0x0000.0000

Registers:

- Register 4: Interrupt 0-31 Set Enable (EN0), offset 0x100
- Register 5: Interrupt 32-63 Set Enable (EN1), offset 0x104
- Register 6: Interrupt 64-95 Set Enable (EN2), offset 0x108
- Register 7: Interrupt 96-127 Set Enable (EN3), offset 0x10C

Note: This register can only be accessed from privileged mode.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

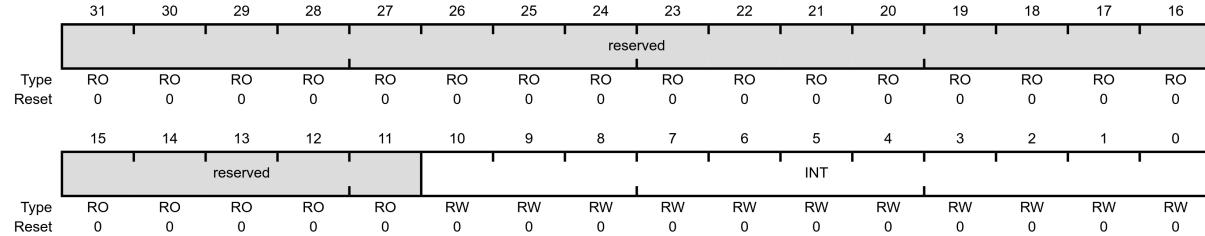
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description						
31:0	INT	RW	0x0000.0000	<p>Interrupt Enable</p> <table> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>On a read, indicates the interrupt is disabled. On a write, no effect.</td> </tr> <tr> <td>1</td> <td>On a read, indicates the interrupt is enabled. On a write, enables the interrupt.</td> </tr> </table> <p>A bit can only be cleared by setting the corresponding INT[n] bit in the DISn (page 466) register.</p>	Value	Description	0	On a read, indicates the interrupt is disabled. On a write, no effect.	1	On a read, indicates the interrupt is enabled. On a write, enables the interrupt.
Value	Description									
0	On a read, indicates the interrupt is disabled. On a write, no effect.									
1	On a read, indicates the interrupt is enabled. On a write, enables the interrupt.									

25.2.2 Interrupt 128-138 Set Enable (EN4)

- Base 0xE000.E000
- Offset 0x110
- Type RW, reset 0x0000.0000

Note: This register can only be accessed from privileged mode.



Bit/Field	Name	Type	Reset	Description						
31:11	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
10:0	INT	RW	0x0	<p>Interrupt Enable</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>On a read, indicates the interrupt is disabled. On a write, no effect.</td> </tr> <tr> <td>1</td> <td>On a read, indicates the interrupt is enabled. On a write, enables the interrupt.</td> </tr> </tbody> </table> <p>A bit can only be cleared by setting the corresponding INT[n] bit in the DIS4 (page 467) register.</p>	Value	Description	0	On a read, indicates the interrupt is disabled. On a write, no effect.	1	On a read, indicates the interrupt is enabled. On a write, enables the interrupt.
Value	Description									
0	On a read, indicates the interrupt is disabled. On a write, no effect.									
1	On a read, indicates the interrupt is enabled. On a write, enables the interrupt.									

25.2.3 Interrupt 0-31 Clear Enable (DIS0)

- Base 0xE000.E000
- Offset 0x180
- Type RW, reset 0x0000.0000

Registers:

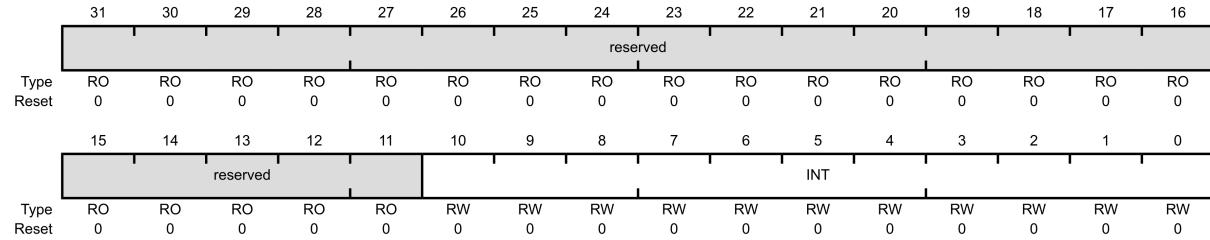
- Register 9: Interrupt 0-31 Clear Enable (DIS0), offset 0x180
- Register 10: Interrupt 32-63 Clear Enable (DIS1), offset 0x184
- Register 11: Interrupt 64-95 Clear Enable (DIS2), offset 0x188
- Register 12: Interrupt 96-127 Clear Enable (DIS3), offset 0x18C

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	RW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description						
31:0	INT	RW	0x0000.0000	<p>Interrupt Disable</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>On a read, indicates the interrupt is disabled.</td> </tr> <tr> <td>1</td> <td>On a read, indicates the interrupt is enabled. On a write, clears the corresponding INT[n] bit in the EN0 (page 464) register, disabling interrupt [n].</td> </tr> </tbody> </table>	Value	Description	0	On a read, indicates the interrupt is disabled.	1	On a read, indicates the interrupt is enabled. On a write, clears the corresponding INT[n] bit in the EN0 (page 464) register, disabling interrupt [n].
Value	Description									
0	On a read, indicates the interrupt is disabled.									
1	On a read, indicates the interrupt is enabled. On a write, clears the corresponding INT[n] bit in the EN0 (page 464) register, disabling interrupt [n].									

25.2.4 Interrupt 128-138 Clear Enable (DIS4)

- Base 0xE000.E000
- Offset 0x190
- Type RW, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description						
31:11	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
10:0	INT	RW	0x0	<p>Interrupt Disable</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>On a read, indicates the interrupt is disabled. On a write, no effect.</td> </tr> <tr> <td>1</td> <td>On a read, indicates the interrupt is enabled. On a write, clears the corresponding INT[n] bit in the EN4 (page 465) register, disabling interrupt [n].</td> </tr> </tbody> </table>	Value	Description	0	On a read, indicates the interrupt is disabled. On a write, no effect.	1	On a read, indicates the interrupt is enabled. On a write, clears the corresponding INT[n] bit in the EN4 (page 465) register, disabling interrupt [n].
Value	Description									
0	On a read, indicates the interrupt is disabled. On a write, no effect.									
1	On a read, indicates the interrupt is enabled. On a write, clears the corresponding INT[n] bit in the EN4 (page 465) register, disabling interrupt [n].									

25.2.5 Interrupt 0-3 Priority (PRI0)

- Base 0xE000.E000
- Offset 0x400
- Type RW, reset 0x0000.0000

Registers:

- Register 29: Interrupt 0-3 Priority (PRI0), offset 0x400
- Register 30: Interrupt 4-7 Priority (PRI1), offset 0x404
- Register 31: Interrupt 8-11 Priority (PRI2), offset 0x408
- Register 32: Interrupt 12-15 Priority (PRI3), offset 0x40C
- Register 33: Interrupt 16-19 Priority (PRI4), offset 0x410
- Register 34: Interrupt 20-23 Priority (PRI5), offset 0x414
- Register 35: Interrupt 24-27 Priority (PRI6), offset 0x418
- Register 36: Interrupt 28-31 Priority (PRI7), offset 0x41C
- Register 37: Interrupt 32-35 Priority (PRI8), offset 0x420
- Register 38: Interrupt 36-39 Priority (PRI9), offset 0x424
- Register 39: Interrupt 40-43 Priority (PRI10), offset 0x428
- Register 40: Interrupt 44-47 Priority (PRI11), offset 0x42C
- Register 41: Interrupt 48-51 Priority (PRI12), offset 0x430
- Register 42: Interrupt 52-55 Priority (PRI13), offset 0x434
- Register 43: Interrupt 56-59 Priority (PRI14), offset 0x438
- Register 44: Interrupt 60-63 Priority (PRI15), offset 0x43C
- Register 45: Interrupt 64-67 Priority (PRI16), offset 0x440
- Register 46: Interrupt 68-71 Priority (PRI17), offset 0x444
- Register 47: Interrupt 72-75 Priority (PRI18), offset 0x448
- Register 48: Interrupt 76-79 Priority (PRI19), offset 0x44C
- Register 49: Interrupt 80-83 Priority (PRI20), offset 0x450
- Register 50: Interrupt 84-87 Priority (PRI21), offset 0x454
- Register 51: Interrupt 88-91 Priority (PRI22), offset 0x458
- Register 52: Interrupt 92-95 Priority (PRI23), offset 0x45C
- Register 53: Interrupt 96-99 Priority (PRI24), offset 0x460
- Register 54: Interrupt 100-103 Priority (PRI25), offset 0x464
- Register 55: Interrupt 104-107 Priority (PRI26), offset 0x468
- Register 56: Interrupt 108-111 Priority (PRI27), offset 0x46C
- Register 57: Interrupt 112-115 Priority (PRI28), offset 0x470
- Register 58: Interrupt 116-119 Priority (PRI29), offset 0x474
- Register 59: Interrupt 120-123 Priority (PRI30), offset 0x478
- Register 60: Interrupt 124-127 Priority (PRI31), offset 0x47C
- Register 61: Interrupt 128-131 Priority (PRI32), offset 0x480
- Register 62: Interrupt 132-135 Priority (PRI33), offset 0x484
- Register 63: Interrupt 136-138 Priority (PRI34), offset 0x488

Note: This register can only be accessed from privileged mode.

The **PRIn** (page 468) registers provide 3-bit priority fields for each interrupt. These registers are byte accessible. Each register holds four priority fields that are assigned to interrupts as follows:

PRIn Register Bit Field	Interrupt
Bits 31:29 (INTD)	Interrupt [4n+3]
Bits 23:21 (INTC)	Interrupt [4n+2]
Bits 15:13 (INTB)	Interrupt [4n+1]
Bits 7:5 (INTA)	Interrupt [4n]

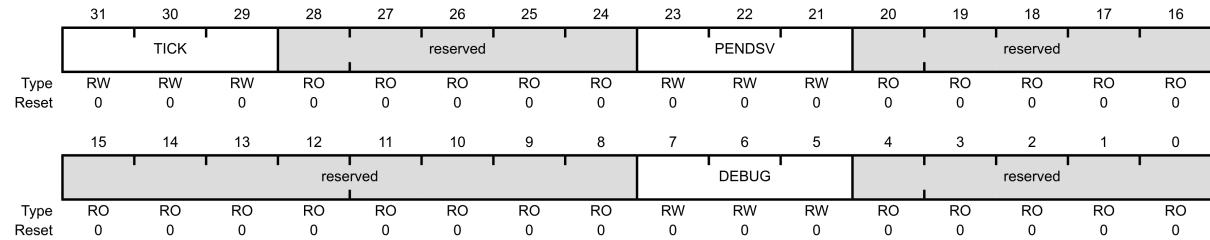
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Type				INTD			reserved				INTC			reserved		
Reset	RW	RW	RW	RO	RO	RO	RO	RO	RW	RW	RW	RO	RO	RO	RO	RO
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type				INTB			reserved			INTA			reserved			
Reset	RW	RW	RW	RO	RO	RO	RO	RO	RW	RW	RW	RO	RO	RO	RO	RO
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:29	INTD	RW	0x0	Interrupt Priority for Interrupt [4n+3] This field holds a priority value, 0-7, for the interrupt with the number [4n+3], where n is the number of the Interrupt Priority register (n=0 for PRI0 (page 468), and so on). The lower the value, the greater the priority of the corresponding interrupt.
28:24	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23:21	INTC	RW	0x0	Interrupt Priority for Interrupt [4n+2] This field holds a priority value, 0-7, for the interrupt with the number [4n+2], where n is the number of the Interrupt Priority register (n=0 for PRI0 (page 468), and so on). The lower the value, the greater the priority of the corresponding interrupt.
20:16	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:13	INTB	RW	0x0	Interrupt Priority for Interrupt [4n+1] This field holds a priority value, 0-7, for the interrupt with the number [4n+1], where n is the number of the Interrupt Priority register (n=0 for PRI0 (page 468), and so on). The lower the value, the greater the priority of the corresponding interrupt.

Bit/Field	Name	Type	Reset	Description
12:8	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:5	INTA	RW	0x0	<p>Interrupt Priority for Interrupt [4n]</p> <p>This field holds a priority value, 0-7, for the interrupt with the number [4n], where n is the number of the Interrupt Priority register (n=0 for PRI0 (page 468), and so on). The lower the value, the greater the priority of the corresponding interrupt.</p>
4:0	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

25.2.6 System Handler Priority 3 (SYSPRI3)

- Base 0xE000.E000
- Offset 0xD20
- Type RW, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:29	TICK	RW	0x0	SysTick Exception Priority This field configures the priority level of the SysTick exception. Configurable priority values are in the range 0-7, with lower values having higher priority.
28:24	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23:21	PENDSV	RW	0x0	PendSV Priority This field configures the priority level of PendSV. Configurable priority values are in the range 0-7, with lower values having higher priority.
20:8	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:5	DEBUG	RW	0x0	Debug Priority This field configures the priority level of Debug. Configurable priority values are in the range 0-7, with lower values having higher priority.
4:0	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

25.3 Working with interrupts

- There are 5 sets of enable (**EN_n** (page 464)) registers on the TM4C123GH6PM.
- They enable the interrupts and show which interrupts are enabled.
- Base: 0xE000.E000
 - Interrupt 0-31 Set Enable (EN0) (page 464), offset 0x100
 - Interrupt 32-63 Set Enable (EN1) (page 464), offset 0x104
 - Interrupt 64-95 Set Enable (EN2) (page 464), offset 0x108
 - Interrupt 96-127 Set Enable (EN3) (page 464), offset 0x10C
 - Interrupt 128-138 Set Enable (EN4) (page 465), offset 0x110
- Bit mappings for **EN_n** (page 464):
 - Bit 0 of **EN0** (page 464) corresponds to interrupt 0, and bit 31 corresponds to interrupt 31.
 - Bit 0 of **EN1** (page 464) corresponds to interrupt 32, and bit 31 corresponds to interrupt 63.
 - Bit 0 of **EN2** (page 464) corresponds to interrupt 64, and bit 31 corresponds to interrupt 95.
 - Bit 0 of **EN3** (page 464) corresponds to interrupt 96, and bit 31 corresponds to interrupt 127.
 - Bit 0 of **EN4** (page 465) corresponds to interrupt 128, and bit 10 corresponds to interrupt 138.
- If a pending interrupt is enabled, the NVIC activates the interrupt based on its priority.
- If an interrupt is not enabled, asserting its interrupt signal changes the interrupt state to pending, but the NVIC never activates the interrupt, regardless of its priority.
- To enable an interrupt, like SS1 (interrupt 34) for example.
 - Set bit 2 of **EN1** (page 464) to 1.
 - Note that **writing zeros** to the registers **EN0** (page 464) through **EN4** (page 465) **does not disable the interrupt**.
 - To disable the interrupt, write 1s to the corresponding **DIS0** (page 466) to **DIS4** (page 467) registers.
- Bit mappings for **DIS_n** (page 466):
 - Bit 0 of **DIS0** (page 466) corresponds to interrupt 0, and bit 31 corresponds to interrupt 31.
 - Bit 0 of **DIS1** (page 466) corresponds to interrupt 32, and bit 31 corresponds to interrupt 63.
 - Bit 0 of **DIS2** (page 466) corresponds to interrupt 64, and bit 31 corresponds to interrupt 95.
 - Bit 0 of **DIS3** (page 466) corresponds to interrupt 96, and bit 31 corresponds to interrupt 127.
 - Bit 0 of **DIS4** (page 467) corresponds to interrupt 128, and bit 10 corresponds to interrupt 138.

25.4 Interrupt priority

- There are 35 priority registers **PRIn** ([page 468](#)) on the TM4C123GH6PM.
- Each interrupt priority level can be specified by a 3-bit priority field giving a priority level of 0 to 7 for each interrupt. 0 is the highest priority, and 7 is the lowest priority.
- Each **PRIn** ([page 468](#)) register controls 4 priority fields. These 4 priority fields (INTA, INTB, INTC, and INTD) are assigned to interrupts as follows:

PRIn Register Bit Field	Interrupt
Bits 31:29 (INTD)	Interrupt [4n+3]
Bits 23:21 (INTC)	Interrupt [4n+2]
Bits 15:13 (INTB)	Interrupt [4n+1]
Bits 7:5 (INTA)	Interrupt [4n]

25.5 Initialising edge triggered interrupt on a pin

For example, making use of Port C bit 4 (PC4), interrupting on the rising edges:

1. Activate clock for Port C by writing 0x04 to the **RCGCGPIO** ([page 133](#)) register.
2. Enable PC4 by writing 0x10 to the **GPIODEN** ([page 135](#)) register.
3. Configure PC4 to be edge sensitive by writing 0x10 to the **GPIOIS** ([page 137](#)) register.
4. Configure PC4 to not interrupt on both edges by clearing the 0x10 bit in the **GPIOIBE** ([page 139](#)) register.
5. Configure PC4 to interrupt on the rising edge by writing 0x10 to the **GPIOIEV** ([page 138](#)) register.
6. Clear the interrupt on PC4 by writing 0x10 to the **GPIOICR** ([page 157](#)) register.
7. Turn on the interrupt on PC4 by writing 0x10 to the **GPIOIM** ([page 140](#)) register.
8. Set the interrupt priority of PC4 to 5 by writing 0x00A0.0000 to the **PRI0** ([page 468](#)) register to set bits 21 - 23.
9. Enable NVIC interrupt 2, which is the handler for Port C, by writing 0x04 to the **EN0** ([page 464](#)) register.

25.6 SysTick periodic interrupt

- The SysTick timer is a simple way to create periodic interrupts.
- A periodic interrupt is one that is generated on a fixed time basis.
- Useful when you need to do periodic polling.

The SysTick registers below are used to create a periodic interrupt.

Address	31-24	23-17	16	15-3	2	1	0	Name
\$E000E010	0	0	COUNT	0	CLK_SRC	INTEN	ENABLE	NVIC_ST_CTRL_R
\$E000E014	0			24-bit RELOAD value				NVIC_ST_RELOAD_R
\$E000E018	0			24-bit CURRENT value of SysTick counter				NVIC_ST_CURRENT_R

Address	31-29	28-24	23-21	20-8	7-5	4-0	Name
\$E000ED20	TICK	0	PENDSV	0	DEBUG	0	NVIC_SYS_PRI3_R

25.6.1 How to determine the frequency of the periodic interrupt

$$F_{\text{bus}} = \text{Frequency of the bus clock}$$

$$n = 24 \text{ bit reload value}$$

Hence the frequency of the periodic interrupt is $\frac{F_{\text{bus}}}{n+1}$.

25.6.2 SysTick periodic interrupt initialisation

- Clear the Enable bit in the **STCTRL** (page 395) register.
- Enter the 24 bit value into the **STRELOAD** (page 397) register.
- Write any value to the **STCURRENT** (page 398) register to clear the counter.
- Set **CLK_SRC** (bit 2) to 1 in the **STCTRL** (page 395) register.
- Enable **ITEN** to enable interrupts in the **STCTRL** (page 395) register.
- Set the priority level of the TICK interrupts (bits 29 - 31) on **SYSPRI3** (page 471) register.
- Set the Enable bit back in the **STCTRL** (page 395) register to turn on the counter.
- When the **CURRENT** value counts to 0, the **COUNT** flag is set.
- On the next clock, the **CURRENT** is loaded with the **RELOAD** value.
- If the **RELOAD** value is n, the **COUNT** flag will be configured to trigger an interrupt at every n+1 counts.

25.6.2.1 Example

- Activate the clock for Port D by writing 0x08 to the **RCGCGPIO** (page 133) register.
- Disable analogue mode for Port D by writing 0 to the **GPIOAMSEL** (page 152) register.
- Set Port D bit 0 (PD0) to output by writing 0x01 to the **GPIODIR** (page 136) register.
- Enable digital I/O on PD0 by writing 0x01 to the **GPIODEN** (page 135) register.
- Disable SysTick during setup by writing 0 to the **STCTRL** (page 395) register.
- Set the reload value by writing 0x00FFFF to the **STRELOAD** (page 397) register.
- Write any value, like a 0, to the **STCURRENT** (page 398) register to clear the register.
- Set **SYSPRI3** (page 471) register to priority 2 by writing 0x40000000 (bits 29 - 31) to it.
- Enable the core clock and interrupts by writing 0x00000007 to the **STCTRL** (page 395) register.

25.7 Timer period interrupt

To create a periodic interrupt:

- Set a prescale value, which can be any 8 bit number, loaded into the **GPTMTAPR** (page 420) register.
- Timer frequency = $\frac{\text{Bus frequency}}{\text{Prescale} + 1}$
- If using the default prescale of 0, then the timer frequency is equal to the bus frequency.
- Load the 32 bit counting value of the time, or the period, into the **GPTMTAILR** (page 423) register.

25.7.1 Initialisation

1. Enable the General Purpose Timer Module in the **RCGCTIMER** (page 440) register.
2. Clear the TAEN bit in the **GPTMCTL** (page 409) register to turn off the timer.
3. Write 0x00 to the **GPTMCFG** (page 407) register to set the timer to 32 bit mode.
4. Write 0x02 to the **GPTMTAMR** (page 414) register to configure for periodic mode.
5. Load the 32 bit PERIOD value into the **GPTMTAILR** (page 423) register.
6. Load PRESCALE into the **GPTMTAPR** (page 420) register.
7. Write a 1 to TAT0CINT in the **GPTMICR** (page 444) register to clear the time out flag.
8. Write a 1 to TATIM in the **GPTMIMR** (page 446) register to set the timeout interrupt.
9. Set the interrupt priority to the correct NVIC register.
10. Enable the correct interrupt in the NVIC interrupt enable register.
11. Set the TAEN bit back on to start the timer to begin counting down from the PERIOD value.

26 ARM reference

26.1 ARM's instruction format

	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		
Cond	0	0	1	Opcode				S	Rn				Rd				Operand2										Data processing/ PSR Transfer							
Cond	0	0	0	0	0	0	A	S	Rd				Rn				Rs				1	0	0	1	Multiply									
Cond	0	0	0	0	1	U	A	S	RdHi				RnLo				Rn				1	0	0	1	Multiply Long									
Cond	0	0	0	1	0	B	0	0	Rn				Rd				0	0	0	0	1	0	0	1	Single data swap									
Cond	0	0	0	1	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1	Rn	Branch and exchange							
Cond	0	0	0	P	U	0	W	L	Rn				Rd				0	0	0	0	1	S	H	1	Halfword data transfer: register offset									
Cond	0	0	0	P	U	1	W	L	Rn				Rd				Offset				1	S	H	1	Halfword data transfer: immediate offset									
Cond	0	1	1	P	U	B	W	L	Rn				Rd				Offset										Single data transfer							
Cond	0	1	1																1									Undefined						
Cond	1	0	0	P	U	S	W	L	Rn				Register List																		Block data transfer			
Cond	1	0	1	L					Offset																			Branch						
Cond	1	1	0	P	U	N	W	L	Rn				CRd				CP#				Offset										Coprocessor data transfer			
Cond	1	1	1	0	CP Opc				CRn				CRd				CP#				CP#				0	COPROCESSOR data operation								
Cond	1	1	1	0	CP Opc				L				CRn				Rd				CP#				1	COPROCESSOR register transfer								
Cond	1	1	1	1											Ignored by processor														Software Interrupt					

- The ARM's instruction format is a 4-byte instruction, consisting of the operation code (opcode), and the operands.
- Format 1: Opcode DestReg, Operand2
- Format 2: Opcode DestReg, SrcReg, Operand2
- Where:
 - Opcode is the operation code.
 - DestReg is the destination register to write the result to.
 - Operand2 is the 2nd operand or argument given to the operation.
 - SrcReg is the source register where the data is read from.
- Most, if not all instructions on the Tiva C are 8-bit instructions and can only take 8-bit values.

26.2 ARM's data formats

- Decimal numbers like 123.
- Hexadecimal numbers like 0x3F.
- n_xxx, where n is the base of the number from 2 to 9, and xxx is the number. For example, 8_12 is the octal number 12.
- A single character constant like 'A'.
- String constants like "string".

26.3 Syntax

26.3.1 Semicolon (;

; This is a comment.

Semicolons in ARM denote the start of a comment.

26.3.2 Hash (#)

#2

- The hash symbol (#) stands for a literal in ARM.
- Use the hash symbol to indicate numbers or strings.

26.3.2.1 Example

Copy the number 1 into the register R0.

`MOV R0, #1`

26.3.3 Square brackets ([])

[value{, optional_offset}]{!}
[value], {optional_offset} ; Post-indexed addressing

- Square brackets ([]) are the **value of** operator in ARM, which is equivalent to **dereferencing a pointer** in low-level programming languages like C or Rust.
- Use it to use the given value (value) as a memory address instead of a regular value.
- An optional offset (optional_offset) can be provided to add the offset to the given value to obtain the memory address.
- The ! represents write-back, and it is optional. Including the ! will update the value given to include the optional_offset, which means value becomes value + optional_offset after the instruction is done.
- The second form is for post-indexed addressing, which is when the value is used as the address for the memory access, then the optional_offset is applied to the address and written back to the register. Basically, value is used to access the memory, then once the memory is accessed, the optional_offset is added to value, so value becomes value + optional_offset.

26.3.3.1 Example 1

- Use the value in the R1 register as a memory address.
- Load into register R5 the data found at the memory address.

`LDR R5, [R1]`

26.3.3.2 Example 2

- Use the value in the R1 register as a memory address, and add 4 to the value.
- Load into register R6 the data found at the memory address.

`LDR R6, [R1, 4]`

26.3.3.3 Example 3

- Use the value in the R1 register as a memory address.
- Load into register R0 the data found at the memory address.
- Register R1 in this example is called the base address register.

`LDR R0, [R1] ; Equivalent to: LDR R0, [R1, #0]
; C code: R0 = memory[R1]`

26.3.3.4 Example 4

- Use the value in the R1 register as a memory address.
- Store into register R0 the data found at the memory address.
- Register R1 in this example is called the base address register.

```
STR R0, [R1]      ; Equivalent to: STR R0, [R1, #0]
; C code: memory[R1] = R0
```

26.3.3.5 Example 5

- Use the value in the R1 register as a memory address, offset by 4.
- Store into register R0 the data found at the memory address.
- Write the offset back into register R1, so that R1 becomes R1 + 4.

```
LDR R0, [R1, #4]!
```

26.3.3.6 Example 6

- Use the value in the R1 register as a memory address.
- Load into register R0 the data found at the memory address given by register R1.
- Offset the memory address in register R1 by 32 and save it back to register R1.

```
LDR R0, [R1], #32
```

26.4 Addressing modes

There are multiple addressing modes that can be used for loads and stores. The number in parentheses refers to the examples below:

- Register addressing, where the address is in a register (1).
- Pre-indexed addressing, where an offset to the base register is added before the memory access. The offset can be positive or negative and can be an immediate value of another register with an optional shift applied. (2), (3).
- Pre-indexed with write-back, which is indicated with an exclamation mark (!) added after the instruction. After the memory access has occurred, this updates the base register by adding the offset value (4).
- Post-index with write-back, where the offset is written after the square bracket. The address from the base register is only used for memory access, with the offset value added to the base register after the memory access (5).

; Address pointed to by R1
(1) LDR R0, [R1]

; Address pointed to by R1 + R2
(2) LDR R0, [R1, R2]

; Address is R1 + (R2 * 4)
(3) LDR R0, [R1, R2, LSL #2]

; Address pointed to by R1 + 32, then R1 = R1 + 32
(4) LDR R0, [R1, #32]!

; Read R0 from pointed to by R1, then R1 = R1 + 32
(5) LDR R0, [R1], #32

26.4.1 Examples with LDR

	Instruction	R0 =	R1 +=
Pre-index with write-back	LDR R0, [R1, #0x4]!	memory[R1 + 0x4]	0x4
	LDR R0, [R1, R2]!	memory[R1 + R2]	R2
	LDR R0, [R1, R2, LSR #0x4]!	memory[R1 + (R2 LSR 0x4)]	R2 LSR 0x4
Pre-index without write-back	LDR R0, [R1, #0x4]	memory[R1 + 0x4]	Not updated
	LDR R0, [R1, R2]	memory[R1 + R2]	Not updated
	LDR R0, [R1, -R2, LSR #0x4]	memory[R1 - (R2 LSR 0x4)]	Not updated
Post-index	LDR R0, [R1], #0x4	memory[R1]	0x4
	LDR R0, [R1, R2]	memory[R1]	R2
	LDR R0, [R1], R2 LSR #0x4	memory[R2]	R2 LSR 0x4

26.4.2 Syntax examples

The shift in the table below can be one of LSL, LSR, ASR or ROR, which are the bit shifting operations in ARM assembly.

Addressing mode and index method	Addressing syntax
Pre-index with immediate offset	[register #+/-offset]
Pre-index with register offset	[register_1 +/-register_2]
Pre-index with scaled register offset	[register_1 +/-register_2, shift #shift_amount]
Pre-index write-back with immediate offset	[register #+/-offset]!
Pre-index write-back with register offset	[register_1 +/-register_2]!
Pre-index write-back with scaled register offset	[register_1 +/-register_2, shift #shift_amount]!
Immediate post-indexed	[register], #+/-offset
Register post-indexed	[register_1], +/-register_2
Scaled register post-indexed	[register_1], +/-register_2, shift #shift_amount

26.5 Condition code suffixes

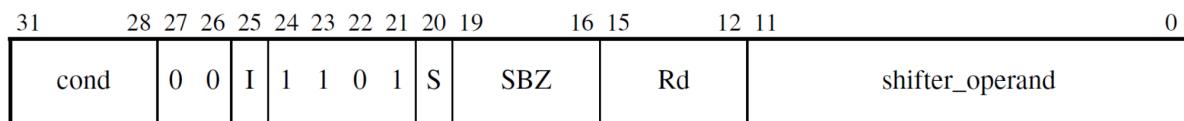
Opcode [31:28]	Suffix	Flags	Meaning
0000	EQ	Z = 1	Equal
0001	NE	Z = 0	Not equal
0010	CS or HS	C = 1	Higher or same, unsigned
0011	CC or LO	C = 0	Lower, unsigned
0100	MI	N = 1	Negative
0101	PL	N = 0	Positive or zero
0110	VS	V = 1	Overflow
0111	VC	V = 0	No overflow
1000	HI	C = 1 and Z = 0	Higher, unsigned
1001	LS	C = 0 or Z = 1	Lower or same, unsigned
1010	GE	N = V	Greater than or equal, signed
1011	LT	N \neq V	Less than, signed
1100	GT	Z = 0 and N = V	Greater than, signed
1101	LE	Z = 1 and N \neq V	Less than or equal, signed
1110	AL	Can have any value	Always, or unconditional. This is the default when no suffix is given.

26.6 ARM CPU instructions

ARM instructions are performed by the microprocessor when the program is being run, and are used to do useful things with the microprocessor, like read and modify data to perform a task.

26.7 MOV (Move)

`MOV{S}{condition} destination, data`



- The MOV instruction copies the data given into the destination register.
- Note that the MOV instruction only supports 8-bit values on the Tiva C.
- Larger values like 32-bit values need to be loaded in some other way, like the LDR pseudo-instruction ([page 482](#)).
- {S} is an optional suffix. If S is specified, the condition flags are updated on the result of the operation. In other words, the current program status register (CPSR) is updated when S is specified.
- {condition} is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.

26.7.1 Example

Copy the value 0x11 into the register R0.

`MOV R0, #0x11`

26.8 MVN (Move Not)

`MVN{S}{condition} destination, data`

- The MVN instruction performs a bitwise NOT on the data given and saves it into the destination register.
- It is the bitwise NOT operator on ARM.
- {S} is an optional suffix. If S is specified, the condition flags are updated on the result of the operation. In other words, the current program status register (CPSR) is updated when S is specified.
- {condition} is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.

26.8.1 Example

Invert the value of 4, which is 100 in binary, to obtain 011 in binary. Then copy the value of 011 into the register R2.

`MVN R2, #4`

26.9 MOVT (Move Top)

`MOVT{S}{condition} destination, data`

- The MOVT instruction is just like the MOV instruction ([page 480](#)), but copies the data given, a 16-bit half-word, into the **top** 16-bits of the register.
- {S} is an optional suffix. If S is specified, the condition flags are updated on the result of the operation. In other words, the current program status register (CPSR) is updated when S is specified.
- {condition} is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.

26.9.1 Example

Copy 0xFEED into the **top** 16-bits of register R3.

`MOVT R3, #0xFEED`

26.10 MOVW (Move Wide)

`MOVW{S}{condition} destination, data`

- The MOVW instruction is just like the MOV instruction ([page 480](#)), but copies the data given, a 16-bit half-word, into the **bottom** 16-bits of the register.
- {S} is an optional suffix. If S is specified, the condition flags are updated on the result of the operation. In other words, the current program status register (CPSR) is updated when S is specified.
- {condition} is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.

26.10.1 Example

Copy 0xC0DE into the **bottom** 16-bits of register R3.

`MOVW R3, #0xC0DE`

26.11 LDR (Load)

`LDR{condition} destination, source`

- The LDR instruction loads the value found at the given memory address (`source`), into the destination register.
- `{condition}` is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.

26.11.1 Example

- Use the value in the R1 register as a memory address.
- Load into register R5 the data found at the memory address.

`LDR R5, [R1]`

26.11.2 Pseudo-instruction

`LDR destination, =value`

The LDR pseudo-instruction either loads a 32-bit immediate value or a memory address (`value`) into the destination register. This instruction is used mostly used to load 32-bit values that cannot be loaded with MOV instruction ([page 480](#)) into a register with one single instruction.

26.12 LDRD (Load Double-Word)

`LDRD{condition} destination, source`

- The LDRD instruction loads a **double-word** of the given source register into the destination memory address.
- The difference between LDRD and the LDR instruction ([page 482](#)) is that LDRD only loads a **double-word**, which is **8 bytes** instead of a **word**, which is **4 bytes**.
- `{condition}` is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.

26.12.1 Example

- Use the value in register R1 as a memory address.
- loads a **double-word** from the memory address of R1, and save the contents in register R4.

`LDRD R4, [R1]`

26.13 LDRB (Load Byte)

`LDRB{condition} destination, source`

- The LDRB instruction loads a **byte** of the given source register into the destination memory address.
- The difference between LDRB and the LDR instruction ([page 482](#)) is that LDRB only loads a **byte**, which is **1 byte** instead of a **word**, which is **4 bytes**.
- `{condition}` is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.

26.13.1 Example

- Use the value in register R1 as a memory address.
- loads a **byte** from the memory address of R1, and save the contents in register R4.

`LDRB R4, [R1]`

26.14 LDRSB (Load Signed Byte)

LDRSB{condition} destination, source

- The LDRSB instruction loads a **signed byte** of the given source register into the destination memory address.
- The difference between LDRSB and the LDR instruction ([page 482](#)) is that LDRSB only loads a **signed byte**, which is **1 signed byte** instead of a **word**, which is **4 bytes**.
- {condition} is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.

26.14.1 Example

- Use the value in register R1 as a memory address.
- loads a **signed byte** from the memory address of R1, and save the contents in register R4.

LDRSB R4, [R1]

26.15 LDRH (Load Half-Word)

LDRH{condition} destination, source

- The LDRH instruction loads a **half-word** of the given source register into the destination memory address.
- The difference between LDRH and the LDR instruction ([page 482](#)) is that LDRH only loads a **half-word**, which is instead of a **word**, which is **4 bytes**.
- {condition} is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.

26.15.1 Example

- Use the value in register R1 as a memory address.
- loads a **half-word** from the memory address of R1, and save the contents in register R4.

LDRH R4, [R1]

26.16 LDRSH (Load Signed Half-Word)

LDRSH{condition} destination, source

- The LDRSH instruction loads a **signed half-word** of the given source register into the destination memory address.
- The difference between LDRSH and the LDR instruction ([page 482](#)) is that LDRSH only loads a **signed half-word**, which is **2 signed bytes** instead of a **word**, which is **4 bytes**.
- {condition} is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.

26.16.1 Example

- Use the value in register R1 as a memory address.
- loads a **signed half-word** from the memory address of R1, and save the contents in register R4.

LDRSH R4, [R1]

26.17 STR (Store)

`STR{condition} destination, source`

- The STR instruction stores the value of the given source register into the destination memory address.
- {condition} is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.

26.17.1 Example 1

- Use the value in the R1 register as a memory address.
- Store the contents of register R3 into the memory address.

`STR R3, [R1]`

26.18 STRD (Store Double-Word)

`STRD{condition} destination, source`

- The STRD instruction stores a **double-word** of the given source register into the destination memory address.
- The difference between STRD and the STR instruction ([page 484](#)) is that STRD only stores a **double-word**, which is **8 bytes** instead of a **word**, which is **4 bytes**.
- {condition} is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.

26.18.1 Example

- Use the value in register R1 as a memory address.
- stores a **double-word** from the contents of register R4 into the memory address.

`STRD R4, [R1]`

26.19 STRB (Store Byte)

`STRB{condition} destination, source`

- The STRB instruction stores a **byte** of the given source register into the destination memory address.
- The difference between STRB and the STR instruction ([page 484](#)) is that STRB only stores a **byte**, which is **1 byte** instead of a **word**, which is **4 bytes**.
- {condition} is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.

26.19.1 Example

- Use the value in register R1 as a memory address.
- stores a **byte** from the contents of register R4 into the memory address.

`STRB R4, [R1]`

26.20 STRSB (Store Signed Byte)

`STRSB{condition} destination, source`

- The STRSB instruction stores a **signed byte** of the given source register into the destination memory address.
- The difference between STRSB and the STR instruction ([page 484](#)) is that STRSB only stores a **signed byte**, which is **1 signed byte** instead of a **word**, which is **4 bytes**.
- {condition} is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.

26.20.1 Example

- Use the value in register R1 as a memory address.
- stores a **signed byte** from the contents of register R4 into the memory address.

`STRSB R4, [R1]`

26.21 STRH (Store Half-Word)

`STRH{condition} destination, source`

- The STRH instruction stores a **half-word** of the given source register into the destination memory address.
- The difference between STRH and the STR instruction ([page 484](#)) is that STRH only stores a **half-word**, which is instead of a **word**, which is **4 bytes**.
- {condition} is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.

26.21.1 Example

- Use the value in register R1 as a memory address.
- stores a **half-word** from the contents of register R4 into the memory address.

`STRH R4, [R1]`

26.22 STRSH (Store Signed Half-Word)

`STRSH{condition} destination, source`

- The STRSH instruction stores a **signed half-word** of the given source register into the destination memory address.
- The difference between STRSH and the STR instruction ([page 484](#)) is that STRSH only stores a **signed half-word**, which is **2 signed bytes** instead of a **word**, which is **4 bytes**.
- {condition} is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.

26.22.1 Example

- Use the value in register R1 as a memory address.
- stores a **signed half-word** from the contents of register R4 into the memory address.

`STRSH R4, [R1]`

26.23 ADD (Add)

ADD{S}{condition} {destination, }register, value

31	28	27	26	25	24	23	22	21	20	19	16	15	12	11	0
cond	0	0	I	0	1	0	0	S	Rn	Rd					shifter operand

- The ADD instruction adds the value in the register with the value given and stores the result in the destination register.
- If the destination register is not given, the result will be stored in the given register.
- {S} is an optional suffix. If S is specified, the condition flags are updated on the result of the operation. In other words, the current program status register (CPSR) is updated when S is specified.
- {condition} is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.

26.23.1 Example 1

Add the value of R0 twice and store the result in register R1

ADD R1, R0, R0

26.23.2 Example 2

Add the value of R0 and 2 and store the result in register R2.

ADD R2, R0, #2

26.23.3 Example 3

Add the value of R0 and R0 shifted left by 2, and store the result in register R2.

ADD R2, R0, R0, LSL #2

26.23.4 Example 4

Add 4 to the value of the program counter (PC), which is the address of the ADD instruction itself plus 8 bytes and store the result in register R2.

ADD R2, PC, #4

26.24 ADC (Add with Carry)

`ADC{S}{condition} {destination, }register, value`

31	28	27	26	25	24	23	22	21	20	19	16	15	12	11	0
cond	0	0	I	0	1	0	1	S	Rn	Rd					shifter_operand

- The ADC instruction adds the value in the given register with the given value and stores the result in the destination register, which is the same as the ADD instruction ([page 486](#)), but it also adds the carry flag to the result of the addition.
- If the destination register is not given, the result will be stored in the given register.
- This instruction is usually used to add the overflow from a previous ADD instruction ([page 486](#)).
- {S} is an optional suffix. If S is specified, the condition flags are updated on the result of the operation. In other words, the current program status register (CPSR) is updated when S is specified.
- {condition} is an optional condition code suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.

26.24.1 Example

- Use ADC to synthesise multi-word addition.
- If register pairs R0, R1 and R2, R3 hold 64-bit values, where R0 and R2 hold the least significant words, the following instructions leave the 64-bit sum in R4 and R5.

<code>ADDS R4, R0, R2</code>		
<code>ADC R5, R1, R3</code>		
	R1 R0	
	+ R3 R2	
	<hr/>	
	R5 R4	

If the second instruction is changed from:

`ADC R5, R1, R3`

To:

`ADCS R5, R1, R3`

The resulting values of the flags indicate:

- N - The 64-bit addition produced a negative result.
- C - An unsigned integer overflow occurred.
- V - A signed integer overflow occurred.
- Z - The most significant 32 bits are all zero.

26.25 SADD16 and SADD8 (Signed Add)

SADD16{condition} {destination, }register_1, register_2
SADD8{condition} {destination, }register_1, register_2

31	28	27	26	25	24	23	22	21	20	19	16	15	12	11	8	7	6	5	4	3	0
cond	0	1	1	0	0	0	0	1			Rn		Rd		SBO	0	0	0	1		Rm

- The SADD16 and SADD8 instructions are signed add operations.
- Both add the values in `register_1` and `register_2` together and store the result in the `destination` register.
- If the `destination` register is not given, the result will be stored in `register_1`.
- They also set the GE bits in the current program status register (CPSR) according to the results of the additions.
- SADD16 performs two 16-bit signed integer additions while SADD8 performs 4 8-bit signed integer additions.

26.25.1 Example

Use the SADD16 instruction to speed up operations on arrays of half-word data. For example:

```
LDR    R3, [R0], #4      ; Load 4 bytes from R0 and skip to the next 4 bytes
LDR    R5, [R1], #4      ; Load 4 bytes from R1 and skip to the next 4 bytes
SADD16 R3, R3, R5       ; Add R3 and R5 together and store the result in R3
STR    R3, [R2], #4      ; Store 4 bytes in R3 and skip to the next 4 bytes
```

The above performs the same operations as the following:

```
LDRH   R3, [R0], #2      ; Load 2 bytes from R0 and skip to the next 2 bytes
LDRH   R4, [R1], #2      ; Load 2 bytes from R1 and skip to the next 2 bytes
ADD    R3, R3, R4       ; Add R3 and R4 together and store the result in R3
STRH   R3, [R2], #2      ; Store 2 bytes in R2 and skip to the next 2 bytes
```

26.26 SUB (Subtract)

`SUB{S}{condition} {destination, }register, value`

31	28	27	26	25	24	23	22	21	20	19	16	15	12	11	0
cond	0	0	I	0	0	1	0	S	Rn	Rd					shifter_operand

- The SUB instruction subtracts the given value from the value in the given register and stores the result in the destination register.
- If the destination register is not given, the result will be stored in the given register.
- {S} is an optional suffix. If S is specified, the condition flags are updated on the result of the operation. In other words, the current program status register (CPSR) is updated when S is specified.
- Borrow = NOT(Carry) = C - 1
- If the S suffix is set, the C flag is set to:
 - ▶ 1 if no borrow occurs.
 - ▶ 0 if a borrow does occur.
- {condition} is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.

26.26.1 Example

Use SUB to subtract one value from another. To decrement the value of register R0, use:

`SUB R0, R0, #1`

SUBS is useful as a loop counter decrement, as the loop branch can test the flags for the appropriate termination condition, without the need for a separate compare instruction:

`SUBS R0, R0, #1`

The code above both decrements the loop counter in R0 and checks whether it has reached zero.

26.27 SBC (Subtract with Carry)

`SUB{S}{condition} {destination, }register, value`

31	28	27	26	25	24	23	22	21	20	19	16	15	12	11	0
cond	0	0	I	0	1	1	0	S	Rn	Rd					shifter_operand

- The SBC instruction subtracts the carry (C) flag from the given value, as well as the value in the given register, and stores the result in the destination register.
- If the destination register is not given, the result will be stored in the given register.
- {S} is an optional suffix. If S is specified, the condition flags are updated on the result of the operation. In other words, the current program status register (CPSR) is updated when S is specified.
- Borrow = NOT(Carry) = C - 1
- If the S suffix is set, the C flag is set to:
 - 1 if no borrow occurs.
 - 0 if a borrow does occur.
- {condition} is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.

26.27.1 Example

Use SBC to synthesise multi-word subtraction. For example, if register pairs R0 and R1 and R2 and R3 hold 64-bit values (R0 and R2 hold the least significant words), the following instructions will leave the 64-bit difference in R4 and R5.

`SUBS R4, R0, R2
SUBS R5, R1, R3`

$$\begin{array}{r}
 & & & & R1 & + & C & - & 1 & R0 \\
 & & & & - & & R3 & & R2 \\
 \hline
 & & & & & & & & R5 & R4
 \end{array}$$

26.28 RSB (Reverse Subtract)

`RSB{S}{condition} {destination, }register, value`

- The RSB instruction subtracts the value in the register from the given value and stores the result in the destination register.
- If the destination register is not given, the result will be stored in the given register.
- {S} is an optional suffix. If S is specified, the condition flags are updated on the result of the operation. In other words, the current program status register (CPSR) is updated when S is specified.
- {condition} is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.

26.28.1 Example

Subtract the value in register R0 from 5, i.e. 5 - R0.

`RSB R0, R0, #5`

26.29 RSC (Reverse Subtract with Carry)

RSC{S}{condition} {destination, }register, value

- The RSC instruction subtracts the value in the register from the given value and stores the result in the destination register, which is the same as the RSB instruction ([page 490](#)), but it adds the negation of the carry flag to the result of the subtraction.
- If the destination register is not given, the result will be stored in the given register.
- This instruction is usually used to add the underflow, or borrow, from a previous RSB instruction ([page 490](#)).
- {S} is an optional suffix. If S is specified, the condition flags are updated on the result of the operation. In other words, the current program status register (CPSR) is updated when S is specified.
- {condition} is an optional condition code suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.

26.29.1 Example

Add the negation of the carry flag to the value in register R2, then subtract the value in register R1 from R2, and store the result in register R0. Essentially, it is $R2 + C - 1$, where C is the carry flag.

RSC R0, R1, R2

26.30 SSUB16 and SSUB8 (Signed Subtract)

SSUB16{condition} {destination, }register_1, register_2

SSUB8{condition} {destination, }register_1, register_2

	31	28	27	26	25	24	23	22	21	20	19	16	15	12	11	8	7	6	5	4	3	0
cond	0	1	1	0	0	0	0	1				Rn	Rd	SBO		0	1	1	1		Rm	

- The SSUB16 and SSUB8 instructions are signed subtract operations.
- Both subtract the value in register_2 from the value in register_1 and store the result in the destination register.
- If the destination register is not given, the result will be stored in register_1.
- They also set the GE bits in the current program status register (CPSR) according to the results of the additions.
- SADD16 performs two 16-bit signed integer additions while SADD8 performs 4 8-bit signed integer additions.

26.30.1 Example

- Use SSUB16 to for operations on arrays of half-word data. This is similar to the way you can use SADD16.
- You can also use SSUB16 for operations on complex numbers that are held as pairs of 16-bit integers of Q15 numbers. If you hold the real and imaginary parts of a complex number in the bottom and top half of a register respectively, then the instruction below will perform the complex arithmetic operation $R0 = R1 - R2$.

SSUB16 R0, R1, R2

26.31 MUL (Multiply)

`MUL{S}{condition} {destination, }register_1, register_2`

31	28	27	26	25	24	23	22	21	20	19	16	15	12	11	8	7	6	5	4	3	0
cond	0	0	0	0	0	0	0	S		Rd		SBZ		Rs	1	0	0	1		Rm	

- The **MUL** instruction multiplies two values from the registers and stores the least significant 32 bits of the result in the **destination** register.
- {S} is an optional suffix. If S is specified, the condition flags are updated on the result of the operation. In other words, the current program status register (CPSR) is updated when S is specified.
- {condition} is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.

26.31.1 Example

Multiply the value of R0 with R1 and store the result in register R2.

`MUL R2, R0, R1`

26.32 MLA (Multiply Accumulate)

`MLA{S}{condition} destination, register_1, register_2, add_register`

31	28	27	26	25	24	23	22	21	20	19	16	15	12	11	8	7	6	5	4	3	0
cond	0	0	0	0	0	0	1	S		Rd		Rn		Rs	1	0	0	1		Rm	

- The **MLA** instruction multiplies two values from the registers, then adds the value from the **add_register**, then stores the least significant 32 bits of the result in the **destination** register.
- Essentially, it does **destination = register_1 * register_2 + add_register**
- {S} is an optional suffix. If S is specified, the condition flags are updated on the result of the operation. In other words, the current program status register (CPSR) is updated when S is specified.
- {condition} is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.

26.32.1 Example

Multiply the value of R0 with R1, then add the value of R3, and store the result in register R2, i.e. $R2 = R0 * R1 + R3$.

`MLA R2, R0, R1, R3`

26.33 MLS (Multiply Subtract)

`MLA{S}{condition} destination, register_1, register_2, subtract_register`

- The MLS instruction multiplies two values from the registers, then subtracts the value in `subtract_register` from the result of the multiplication, then stores the least significant 32 bits of the result in the `destination` register.
- Essentially, it does `destination = register_1 * register_2 - subtract_register`
- {S} is an optional suffix. If S is specified, the condition flags are updated on the result of the operation. In other words, the current program status register (CPSR) is updated when S is specified.
- {condition} is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.

26.33.1 Example

Multiply the value of R0 with R1, then subtract the value of R3 from the result, and store the result in register R2, i.e. $R2 = R0 * R1 - R3$.

`MLS R2, R0, R1, R3`

26.34 UMULL (Unsigned Multiply Long)

`UMULL{S}{condition} destination_low, destination_high, register_1, register_2`

	31	28	27	26	25	24	23	22	21	20	19	16	15	12	11	8	7	6	5	4	3	0
cond	0	0	0	0	1	0	0	S		RdHi		RdLo		Rs		1	0	0	1		Rm	

- The UMULL instruction takes the values from registers `register_1` and `register_2` as unsigned integers and multiplies these two integers. Then it places the least significant 32 bits of the result in `destination_low` and the most significant 32 bits of the result in `destination_high`.
- {S} is an optional suffix. If S is specified, the condition flags are updated on the result of the operation. In other words, the current program status register (CPSR) is updated when S is specified.
- {condition} is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.

26.34.1 Example

Multiply the values in R2 and R3 together, then places the least significant 32 bits of the result in R0, and the most significant 32 bits of the result in R1.

`UMULL R0, R1, R2, R3`

26.35 UMLAL (Unsigned Multiply Long with Accumulate)

`UMLAL{S}{condition} destination_low, destination_high, register_1, register_2`

- The UMLAL instruction takes the values from registers `register_1` and `register_2` as unsigned integers and multiplies these two integers. Then it adds the result to the 64-bit unsigned integer contained in `destination_low` and `destination_high`.
- {S} is an optional suffix. If S is specified, the condition flags are updated on the result of the operation. In other words, the current program status register (CPSR) is updated when S is specified.
- {condition} is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.

26.35.1 Example

Multiply the values in R2 and R3 together, then add the result to the 64-bit unsigned integer contained in R0 and R1.

`UMLAL R0, R1, R2, R3`

26.36 SMULL (Signed Multiply Long)

`SMULL{S}{condition} destination_low, destination_high, register_1, register_2`

- The SMULL instruction takes the values from registers `register_1` and `register_2` as two's complement signed integers and multiplies these two integers. Then it places the least significant 32 bits of the result in `destination_low` and the most significant 32 bits of the result in `destination_high`.
- {S} is an optional suffix. If S is specified, the condition flags are updated on the result of the operation. In other words, the current program status register (CPSR) is updated when S is specified.
- {condition} is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.

26.36.1 Example

Multiply the values in R2 and R3 together, then places the least significant 32 bits of the result in R0, and the most significant 32 bits of the result in R1.

`SMULL R0, R1, R2, R3`

26.37 SMLAL (Signed Multiply Long with Accumulate)

`SMLAL{S}{condition} destination_low, destination_high, register_1, register_2`

- The SMLAL instruction takes the values from registers `register_1` and `register_2` as two's complement signed integers and multiplies these two integers. Then adds the result to the 64-bit signed integer contained in `destination_low` and `destination_high`.
- {S} is an optional suffix. If S is specified, the condition flags are updated on the result of the operation. In other words, the current program status register (CPSR) is updated when S is specified.
- {condition} is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.

26.37.1 Example

Multiply the values in R2 and R3 together, then adds the result to the 64-bit signed integer contained in R0 and R1.

`SMLAL R0, R1, R2, R3`

26.38 UDIV (Unsigned Divide)

`UDIV{condition} {destination, }register_1, register_2`

- The UDIV instruction divides the 32-bit unsigned integer value in `register_1` by the 32-bit unsigned integer value in `register_2`, and stores the result in the `destination` register.
- If the `destination` register is not given, the result will be stored in `register_1`.
- `{condition}` is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.

26.38.1 Example

Perform unsigned integer division by dividing the value in R2 by the value in R0, and store the result in the register R3.

`UDIV R3, R2, R0`

26.39 AND (Bitwise AND)

`AND{S}{condition} destination, register, value`

31	28	27	26	25	24	23	22	21	20	19	16	15	12	11	0
cond	0	0	I	0	0	0	0	S	Rn	Rd					shifter_operand

0	0	1	1	1	1	0	0	
AND	1	0	1	0	1	0	1	0
<hr/>								
0	0	1	0	1	0	0	0	

- The AND instruction performs a bitwise AND on the value from the `register` and the given `value`, and stores the result in the `destination` register.
- `{S}` is an optional suffix. If S is specified, the condition flags are updated on the result of the operation. In other words, the current program status register (CPSR) is updated when S is specified.
- `{condition}` is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.

26.39.1 Example

Perform a bitwise AND on the value in R3 and R0 and store the result in register R4.

`AND R4, R3, R0`

26.39.2 Example of forcing bits to 0

- `R0 = 0000 0001 0001 0000`
- `~R0 = 1111 1110 1110 1111`
- `R1 = 0000 0000 0001 0001`
- `R3 = 1010 1011 1100 1101`

```
MOVW  R0, #0x0110      ; Bit mask
MVN   R0, R0          ; Invert the bit mask
AND   R2, R1, R0      ; Zero out the bits in R1 that are also in the mask
AND   R4, R3, R0      ; Zero out the bits in R3 that are also in the mask
```

Result:

- `R2 = 0000 0000 0000 0001`
- `R4 = 1010 1010 1100 1101`

26.40 ORR (Bitwise OR)

ORR{S}{condition} destination, register, value

31	28	27	26	25	24	23	22	21	20	19	16	15	12	11	0
cond	0	0	I	1	1	0	0	S	Rn	Rd					shifter_operand

0	0	1	1	1	1	0	0	
OR	1	0	1	0	1	0	1	0
	1	0	1	1	1	1	0	

- The ORR instruction performs a bitwise OR on the value from the register and the given value, and stores the result in the destination register.
- {S} is an optional suffix. If S is specified, the condition flags are updated on the result of the operation. In other words, the current program status register (CPSR) is updated when S is specified.
- {condition} is an optional condition code suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.

26.40.1 Example

Perform a bitwise OR on the value in R1 and R0 and store the result in register R2.

ORR R2, R1, R0

26.40.2 Example of forcing bits to 1

- R0 = 0000 0001 0001 0000
- R1 = 0000 0000 0001 0001
- R3 = 1010 1011 1100 1101

```
MOVW R0, #0x0110 ; Bit mask
ORR R2, R1, R0 ; Force the bits in R1 that are also in the mask to be 1
ORR R4, R3, R0 ; Force the bits in R3 that are also in the mask to be 1
```

Result:

- R2 = 0000 0001 0001 0001
- R4 = 1010 1011 1101 1101

26.41 EOR (Exclusive OR, XOR)

EOR{S}{condition} destination, register, value

31	28	27	26	25	24	23	22	21	20	19	16	15	12	11	0
cond	0	0	I	0	0	0	1	S	Rn		Rd		shifter_operand		

$$\begin{array}{r} 0 \ 0 \ 1 \ 1 \ 1 \ 0 \ 0 \\ \text{XOR} \ 1 \ 0 \ 1 \ 0 \ 1 \ 0 \ 0 \\ \hline 1 \ 0 \ 0 \ 1 \ 0 \ 1 \ 1 \ 0 \end{array}$$

- The EOR instruction performs a bitwise XOR, or exclusive OR on the value from the register and the given value, and stores the result in the destination register.
- {S} is an optional suffix. If S is specified, the condition flags are updated on the result of the operation. In other words, the current program status register (CPSR) is updated when S is specified.
- {condition} is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.

26.41.1 Example

Perform a bitwise XOR, or exclusive OR, on the value in R1 and R0 and store the result in register R2.

EOR R2, R1, R0

26.41.2 Example of flipping bits

- R0 = 0000 0001 0001 0000
- R1 = 0000 0000 0001 0001
- R3 = 1010 1011 1100 1101

```
MOVW R0, #0x0110 ; Bit mask
EOR R2, R1, R0 ; Flip the bits in R1 that are also in the mask
EOR R4, R3, R0 ; Flip the bits in R3 that are also in the mask
```

Result:

- R2 = 0000 0001 0000 0001
- R4 = 1010 1010 1101 1101

26.42 BIC (Bit Clear)

BIC{S}{condition} destination, source, bits

31	28	27	26	25	24	23	22	21	20	19	16	15	12	11	0
cond	0	0	I	1	1	1	0	S	Rn	Rd					shifter_operand

$$\begin{array}{r} 0 \ 0 \ 1 \ 1 \ 1 \ 1 \ 0 \ 0 \\ BIC \ 1 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1 \ 0 \\ \hline 0 \ 0 \ 0 \ 1 \ 0 \ 1 \ 0 \ 0 \end{array}$$

- The BIC (bit clear) instruction performs a bitwise AND ([page 495](#)) on the bits in the source register with the complements of the corresponding bits in the bits given, then copies the result to the destination register.
- Essentially, it does source AND NOT bits and copies the result to the destination register.
- If the documentation above still does not make sense, the instruction's name should give a good idea of what it does.
- Basically, to clear a bit (i.e. set the bit to 0) in the source register, the bits to clear should be indicated by a 1 in the binary value given in bits.
- To clear a byte instead, provide a hexadecimal value with the byte to clear indicated as F to clear a whole byte at once.
- {S} is an optional suffix. If S is specified, the condition flags are updated on the result of the operation. In other words, the current program status register (CPSR) is updated when S is specified.
- {condition} is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.

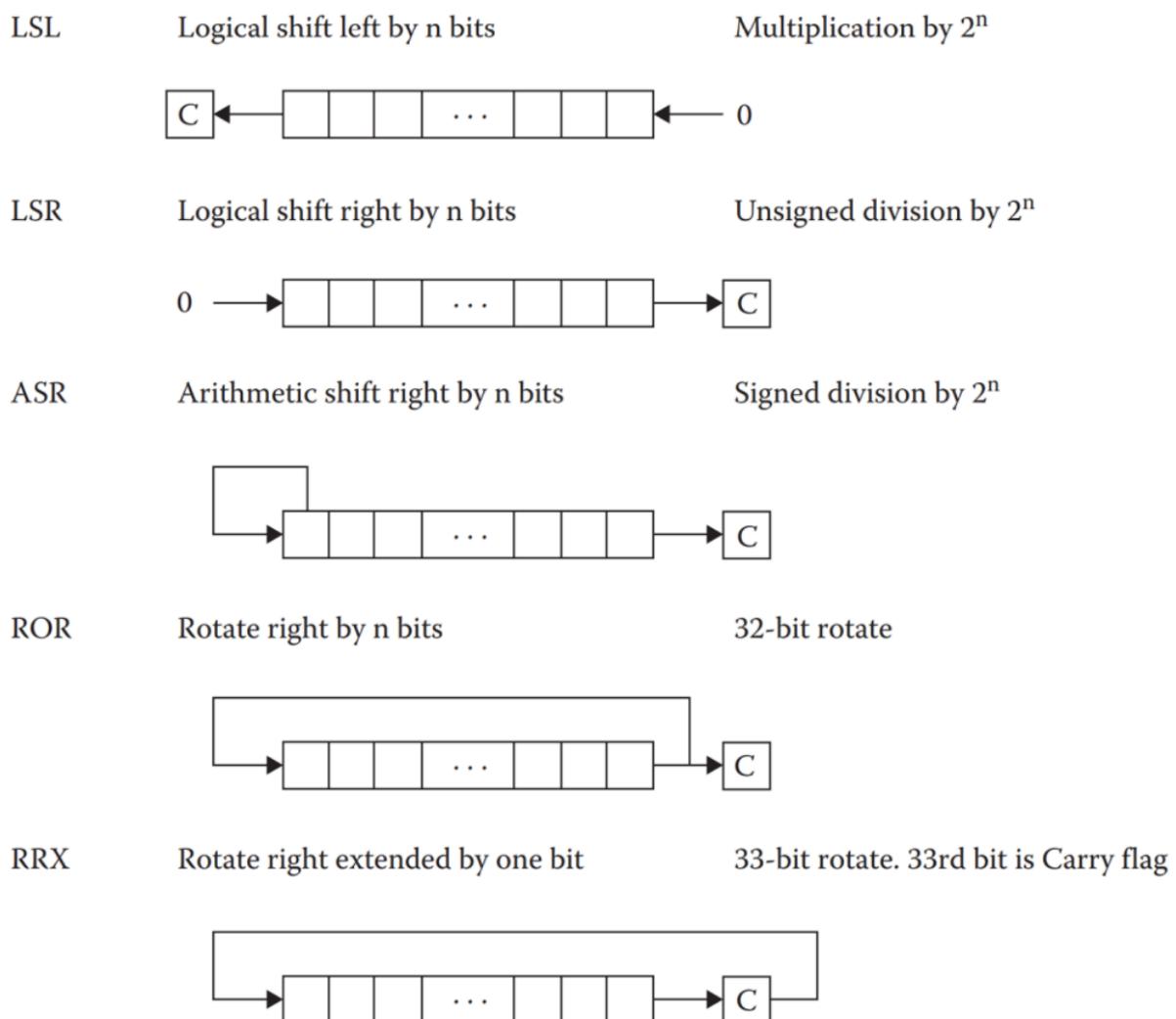
26.42.1 Example

Clear the last byte in the register R1 and copy the result to register R2.

BIC R2, R1, #0xF

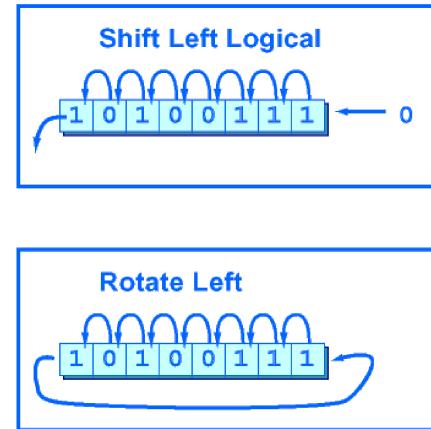
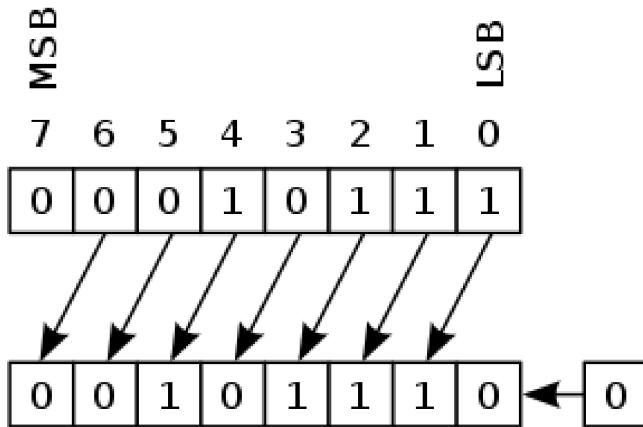
26.43 Shift operators

- LSL: Logical shift left
 - $x \ll y$ the least significant bits are filled with zeros.
- LSR: Logical shift right
 - $(\text{unsigned}) x \gg y$ the most significant bits are filled with zeros.
- ASR: Logical shift right
 - $(\text{signed}) x \gg y$, copy the sign bit to the most significant bit.
- ROR: Rotate right
 - $((\text{unsigned}) x \gg y) | (x \ll (32 - y))$.
- RRX: Rotate right extended
 - $C \ll 31 | ((\text{unsigned}) x \gg 1)$
 - Performs 33-bit rotate, with the current program status register's C bit being inserted above the sign bit of the word.



26.44 LSL (Logical Shift Left)

`LSL{S}{condition} destination, value, shift_amount`



- The LSL instruction performs a bitwise logical shift left on the given value by the given `shift_amount`.
- The least significant bits are filled with zeros.
- This is equivalent to C's value `<< shift_amount`.
- {S} is an optional suffix. If S is specified, the condition flags are updated on the result of the operation. In other words, the current program status register (CPSR) is updated when S is specified.
- {condition} is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.

26.44.1 Example 1

Perform a logical left shift by 2 bits on the value in register R1 and store the result in register R0.

`LSL R0, R1, #2`

26.44.2 Example 2

```

AREA    Prog1, CODE, READONLY
ENTRY
    MOV    R0, #0x11    ; Load initial value
    LSL    R1, R0, #1    ; Shift 1 bit left
    LSL    R2, R1, #1    ; Shift 1 bit left

stop   B    stop      ; Stop program
END

```

26.44.3 Example 3

Example of doing n! (factorial of n).

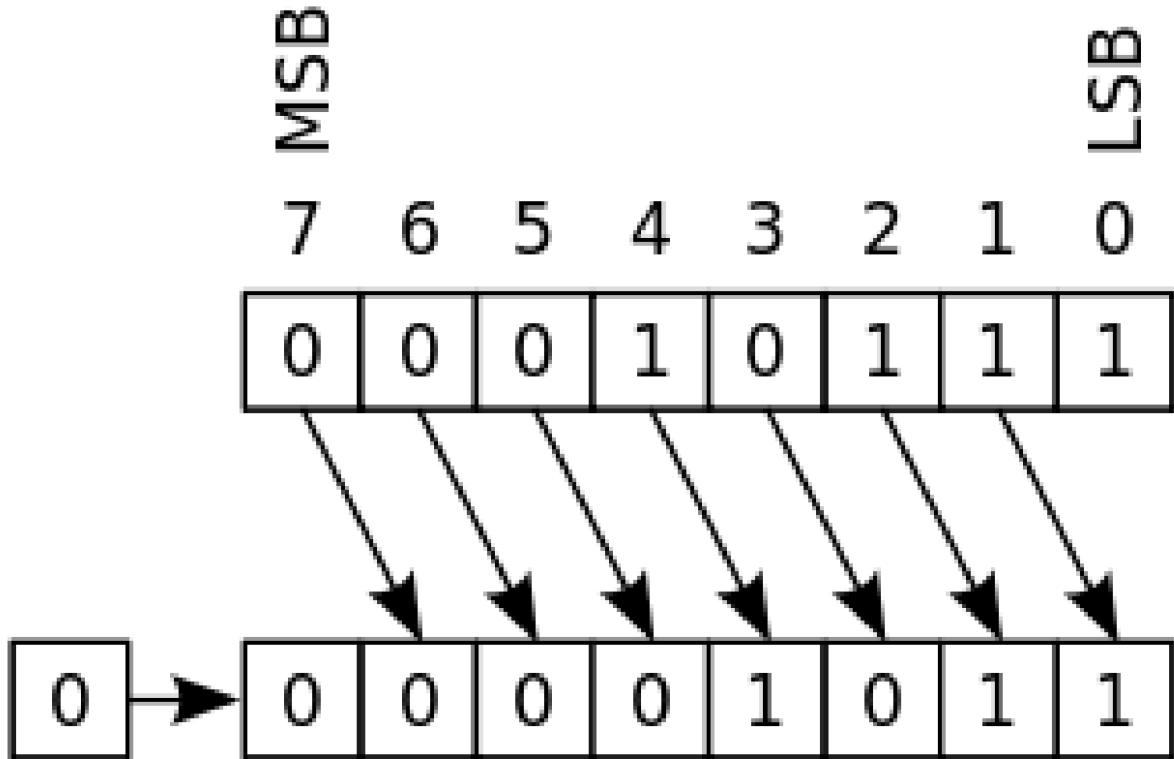
```

AREA    Prog2, CODE, READONLY
ENTRY
    MOV    R6, #10      ; Load n into R6
    MOV    R7, #1        ; Initial value of n! is 1, for the case of n = 0
loop   CMP    R7, #0      ; Compare R6 with 0
    MULGT R7, R6, R7    ; n(n - 1)(n - 2) * ... * 2 * 1
    SUBGT R6, R6, #1    ; Decrement n by 1
    BGT   loop          ; Multiply again if the counter isn't 0
stop   B    stop      ; Stop program
END

```

26.45 LSR (Logical Shift Right)

`LSR{S}{condition} destination, value, shift_amount`



- The LSR instruction performs a bitwise logical shift right on the given `value` by the given `shift_amount`.
- {S} is an optional suffix. If S is specified, the condition flags are updated on the result of the operation. In other words, the current program status register (CPSR) is updated when S is specified.
- {condition} is an optional condition code suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.

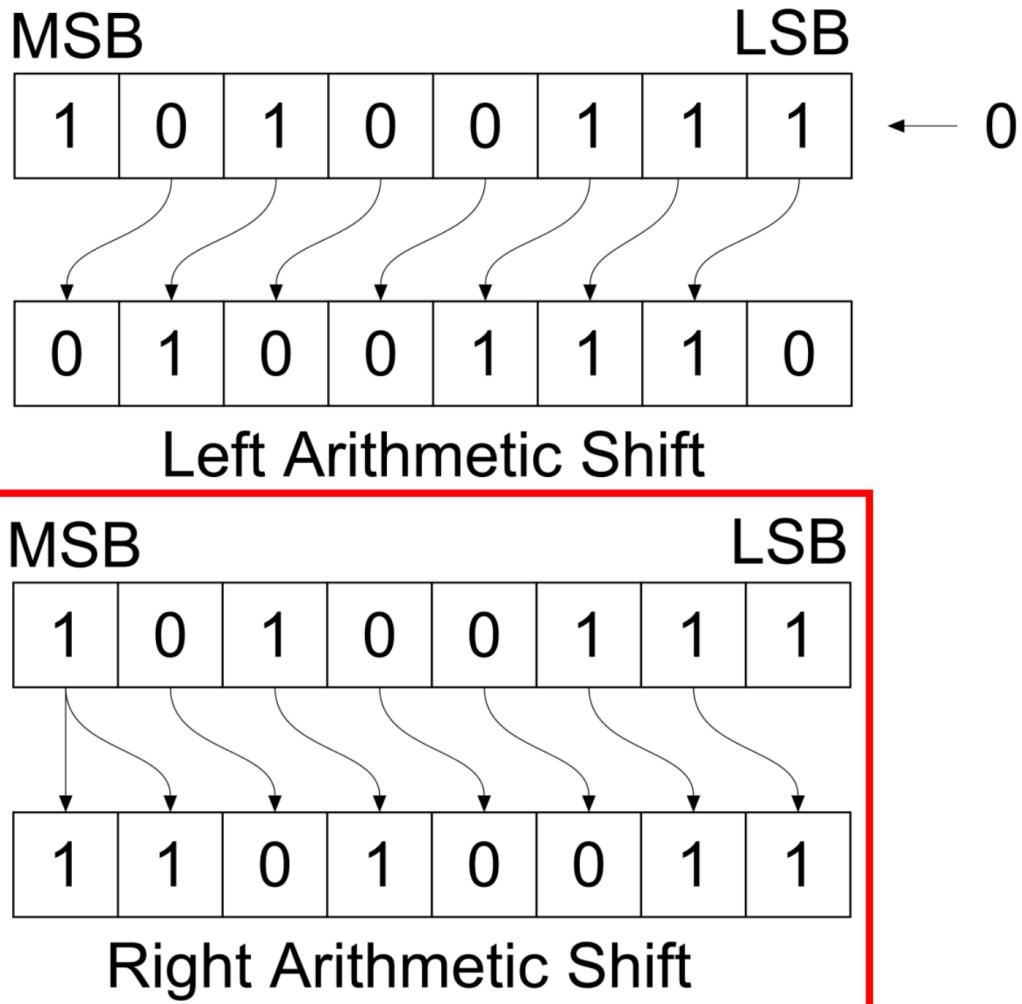
26.45.1 Example

Perform a logical right shift by 6 bits on the value in register R1 and store the result in register R0.

`LSR R0, R1, #6`

26.46 ASR (Arithmetic Shift Right)

`ASR{S}{condition} destination, value, shift_amount`



- The ASR instruction performs a bitwise arithmetic shift right on the given value by the given `shift_amount`.
- The only difference between ASR and LSR ([page 501](#)), or an arithmetic right shift and a logical right shift, is that the highest bit is preserved when doing an arithmetic right shift, while a logical right shift will just set the highest bit to 0. This is useful for calculating signed integers, as the arithmetic right shift will preserve the sign of the integer, i.e. a negative integer will remain negative, while you lose the sign when doing a logical right shift.
- {S} is an optional suffix. If S is specified, the condition flags are updated on the result of the operation. In other words, the current program status register (CPSR) is updated when S is specified.
- {condition} is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.

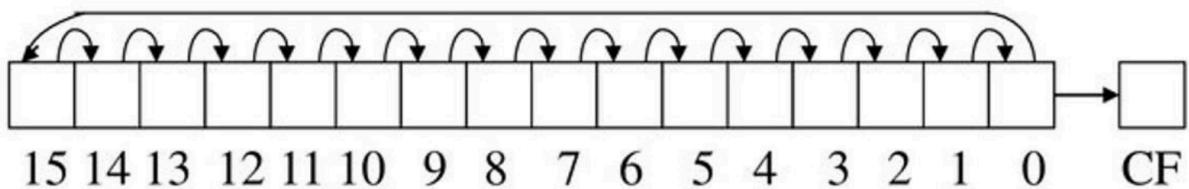
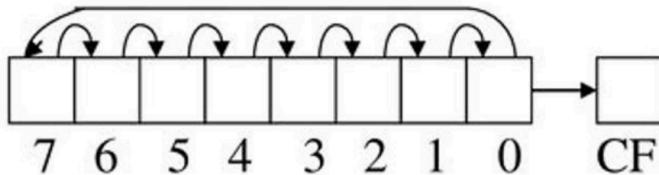
26.46.1 Example

Perform an arithmetic right shift by 4 bits on the value in register R1 and store the result in register R0.

`ASR R0, R1, #4`

26.47 ROR (Rotate Right)

ROR{S}{condition} destination, value, shift_amount



$x >> y; \quad y=7$



$$y = 7; 32 - y = 25; \quad 25 + 6 = 31$$

$$(0000000b31\dots b7) | (b6\dots b0 \dots 0)$$

- The ROR instruction rotates the given `value` right by the given `shift_amount`.
- Essentially, what this instruction does is the same as the LSR instruction ([page 501](#)), or logical shift right, but instead of replacing the highest bits with zeros, they are instead replaced with the lowest bits that are shifted off.
- {S} is an optional suffix. If S is specified, the condition flags are updated on the result of the operation. In other words, the current program status register (CPSR) is updated when S is specified.
- {condition} is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.

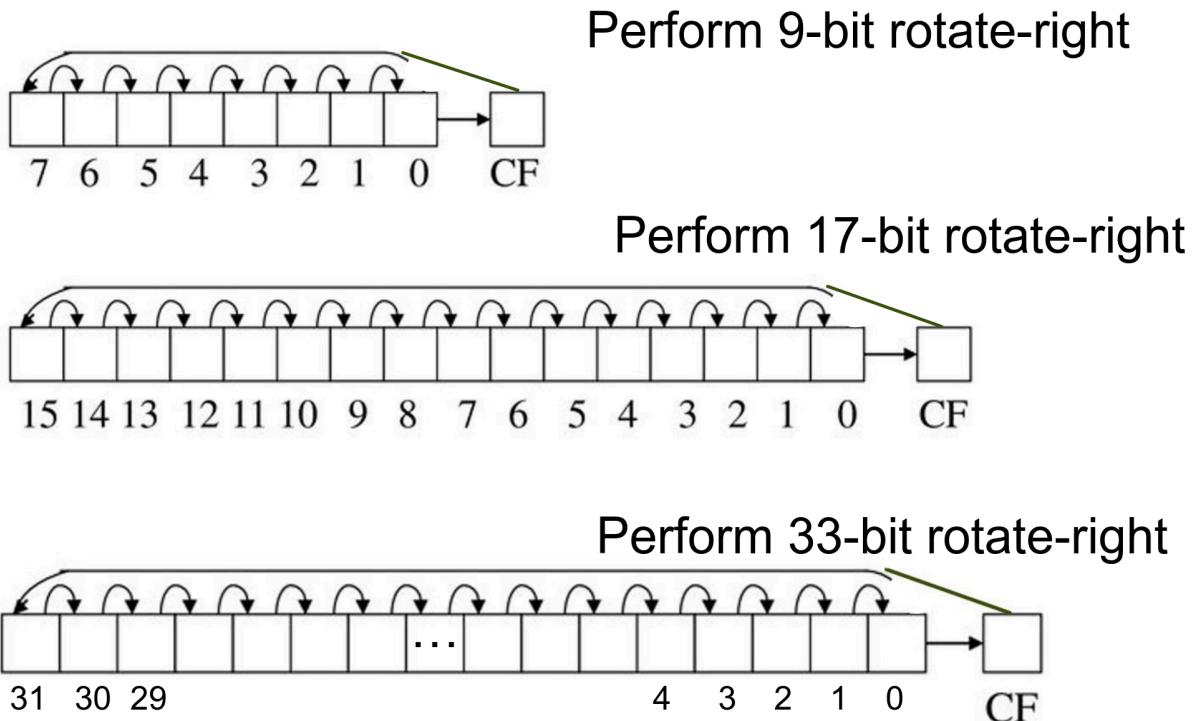
26.47.1 Example

Rotate the value in register R1 right by 2 bits and store the result in register R0.

ROR R0, R1, #2

26.48 RRX (Rotate Right Extended)

RRX{S}{condition} destination, value, shift_amount



- The ROR instruction rotates the given value right by the given shift_amount, but also writes to the carry (C) flag.
- Essentially, what this instruction does is the same as the LSR instruction ([page 503](#)), or rotate right, but instead of just having 32-bits to rotate around, there are 33-bits thanks to the carry (C) flag being included.
- {S} is an optional suffix. If S is specified, the condition flags are updated on the result of the operation. In other words, the current program status register (CPSR) is updated when S is specified.
- {condition} is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.

26.48.1 Example

Rotate the value in register R1 right by 10 bits, including the carry bit (C) and store the result in register R0.

RRX R0, R1, #10

26.49 B (Branch)

B{condition}{.W} label

31	28	27	26	25	24	23	0
cond	1	0	1	L			signed_immed_24

- The B instruction tells the microprocessor to branch to a different part of the code, which is indicated by the label.
- label can also be a register containing a memory address.
- {condition} is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.

26.49.1 Example 1

Go to the section of the code labelled loop.

B loop

26.49.2 Example 2

Go to the section of the code labelled loop if the carry flag is clear.

BCC loop

26.49.3 Example 3

Go to the section of the code labelled loop if the zero flag is set

BEQ loop

26.50 BL (Branch with Link)

BL{condition}{.W} label

- The BL instruction tells the microprocessor to branch to a different part of the code, which is indicated by the label, which is the same function as the B instruction ([page 505](#)), but also copies the memory address of the next instruction into the link register LR.
- This instruction is normally used with the BX instruction ([page 506](#)), where the BL instruction will jump to a section of code to perform a certain task, and the BX instruction will be used to jump back to the link register LR to continue executing from where it left off.
- label can also be a register containing a memory address.
- {condition} is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.

26.50.1 Example

Branch into the section of the code named function and save the memory address of the next instruction into the link register LR.

BL function

26.51 BX and BXNS (Branch and Exchange, and Branch and Exchange Non-Secure)

`BX{condition} register`
`BXNS{condition} register`

- The `BX` instruction tells the microprocessor to branch to the address contained in `register` and exchanges the instruction set if necessary.
- The `BXNS` instruction performs the same function as the `BX` instruction, but also transitions from the secure to the non-secure domain.
- If the first bit of the address contained in `register` is 0, the processor changes to, or remains in ARM state.
- If the first bit of the address contained in `register` is 1, the processor changes to, or remains in Thumb state.
- This instruction is normally used with the `BL` instruction ([page 505](#)), where the `BL` instruction will jump to a section of code to perform a certain task, and the `BX` instruction will be used to jump back to the link register `LR` to continue executing from where it left off.

26.51.1 Example 1

Branch to the link register, which is usually used to return from a function in ARM assembly.

`BX LR`

26.51.2 Example 2

```
; Subroutine call to function
BL func

func
    ; Function body

    ; Code
    ; Code
    ; Code

    ; R15 = R14, set the program counter to the value of the link register.
    ;
    ; This jumps out of the function
    ; and goes back to the instruction after the BL.
    MOV PC, LR

    ; Branch to the address in R12 and
    ; begin ARM execution if bit 0 of R12 is zero.
    ; Otherwise, continue executing thumb code.
    BX R12
```

26.52 ADR

`ADR destination, label`

- The `ADR` instruction loads a memory address to the `destination` register.
- The `label` is the label of the memory address to load. Note that it must be defined in the **same** code section.

26.52.1 Example

Get the memory address of `var_a` and save it in register `R0`.

`ADR R0, var_a`

26.53 CMP (Compare)

`CMP{condition} register, value`

- The `CMP` instruction compares whether the value in the `register` is the same as the given `value`, and updates the condition flags ([page 480](#)) on the result.
- This instruction does not place the result in any register.
- `CMP` subtracts the given `value` from the value in the `register`.
- `{condition}` is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.

26.53.1 Example

Check whether the value in register `R1` is equal to 1.

`CMP R1, #1`

26.54 CMN (Compare Negative)

`CMN{condition} register, value`

- The `CMN` instruction compares whether the value in the `register` is the same as the given `value`, and updates the condition flags ([page 480](#)) on the result.
- This instruction does not place the result in any register.
- `CMN` adds the given `value` to the value in the `register`.
- `{condition}` is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.

26.54.1 Example

Check whether the value in register `R0` is **not** equal to 1.

`CMN R0, #1`

26.55 TEQ (Test Equivalence)

TST{condition} register, value

31	28	27	26	25	24	23	22	21	20	19	16	15	12	11	0
cond	0	0	I	1	0	0	1	1	Rn	SBZ			shifter_operand		

- The TEQ instruction tests the value in the register for equality with the value given, and updates the condition flags ([page 480](#)).
- More specifically, the TEQ instruction performs a **bitwise XOR or Exclusive OR** operation on the value in register and the given value.
- It is the same as the E0RS instruction ([page 497](#)), but the result is discarded.
- This instruction is the same as `register == value` in higher level programming languages like C and Python.
- This instruction does not store the result in any register, and only updates the condition flags.
- {condition} is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.
- Use the TEQ instruction to test if two values are equal without affecting the V or C flags.
- TEQ is also useful for testing the sign of a value. After the comparison, the N flag is the logical Exclusive OR of the sign bits of the two given values.

26.55.1 Example

Conditionally test if the value in R10 is equal to the value in R9. The application program status register (APSR) is updated, but the result is discarded.

`TEQEQ R10, R9`

26.56 TST (Test bits)

TST{condition} register, value

- The TST instruction performs a **bitwise AND** operation on the value in register and the given value.
- It is the same as the ANDS instruction ([page 495](#)), but the result is discarded.
- This instruction does not store the result in any register, and only updates the condition flags.
- {condition} is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.
- To test whether a bit of the given register is 0 or 1, use the TST instruction with a constant that has that bit set to 1 and all other bits cleared to 0.

26.56.1 Example

Perform bitwise AND of the value in R0 to 0x3F8. The application program status register (APSR) is updated, but the result is discarded.

`TST R2, #1`

26.57 Vector floating point (VFP) operations

- Before performing any vector floating point (VFP) operations on the Tiva C microcontroller, the floating point unit (FPU) needs to be enabled.
- The processor must be in privileged mode to read from and write to the Coprocessor Access Control (CPAC) register.
- The code below enables the FPU in both privileged and user modes.

```
; CPAC register is located ad address 0xE000ED88,  
; so load the address into a register  
; using the LDR pseudo-instruction as it is 32-bit  
LDR R0, =0xE000ED88  
  
; Read from the CPAC register into another register  
LDR R1, [R0]  
  
; Set bits 20 - 23 to enable CP10 and CP11 coprocessors  
ORR R1, R1, #(0xF << 20)  
  
; Write the modified value back to the CPAC register  
STR R1, [R0]
```

- All VFP operations basically perform the same function as their non VFP counterparts, but manipulate data on either a single-precision or a double-precision floating point register.
- A single-precision floating point register is usually a register on the floating point unit (FPU) starting with S, like S0 and S1. A single-precision floating point register can hold 32-bits.
- A double-precision floating point register is usually a register on the floating point unit (FPU) starting with D, like D0 and D1. A double-precision floating point register can hold 64-bits.

26.58 VLDR (Vector Load)

```
VLDR{condition}.32 destination, value  
VLDR{condition}.64 destination, value
```

- VLDR performs the same function as the LDR instruction ([page 482](#))
- The main difference with VLDR and LDR is that the destination can either be a single-precision, like S0 and S1, or double-precision floating point register, like D0 and D1.
- {condition} is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.
- The .32 and .64 part of the instruction is optional.

26.58.1 Example

Load the value from the memory address in the register R6 offset by 8 into the single-precision floating point register S5.

```
VLDR S5, [R6, #08]
```

26.59 VSTR (Vector Load)

VSTR{condition}.32 destination, value
VSTR{condition}.64 destination, value

- VSTR performs the same function as the STR instruction ([page 484](#))
- The main difference with VSTR and STR is that the destination can either be a single-precision, like S0 and S1, or double-precision floating point register, like D0 and D1.
- {condition} is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.
- The .32 and .64 part of the instruction is optional.

26.59.1 Example

Store the value from the single-precision floating point register S5 to the memory address in the register R6.

VSTR R6, S5

26.60 VMOV (Vector Move)

VMOV{condition}.F32 destination, value
VMOV{condition}.F64 destination, value

- VMOV performs the same function as the MOV instruction ([page 480](#))
- The main difference with VMOV and MOV is that the destination can either be a single-precision, like S0 and S1, or double-precision floating point register, like D0 and D1.
- {condition} is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.
- The 32 and 64 part of the instruction is optional.

26.60.1 Example

Load the value from register R6 into single-precision register S12, and the value from register R11 into single-precision register S13.

VMOV S12, S13, R6, R11

26.61 VADD (Vector Add)

VADD{condition}.F32 destination, value
VADD{condition}.F64 destination, value

- VADD performs the same function as the ADD instruction ([page 486](#))
- The main difference with VADD and ADD is that the destination can either be a single-precision, like S0 and S1, or double-precision floating point register, like D0 and D1.
- {condition} is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.
- The 32 and 64 part of the instruction is optional.

26.61.1 Example

Add the value from single-precision registers S1 and S0 and store the result in single-precision register S2.

VADD.F S2, S1, S0

26.62 Directives

- Directives are **not** CPU instructions, as they are not executed by the CPU when running the program.
- Instead, directives are instructions to the assembler, which is the ARM assembler (`armasm`) if you are using the ARM Keil IDE.
- If you are using GCC or ArmClang, then the directives would differ from the ones listed below.

26.63 Allocating memory to house instructions or data

{label} {instruction | directive | pseudo_instruction}

- The `label` is the label for the memory allocation.
- `{instruction | directive | pseudo_instruction}` is the CPU instructions or assembler directives to execute.

26.63.1 Example

```
EXPORT Start      ; Export the start label of the program

          ; Assembler directive to allocate memory for code
AREA ARMEx, CODE, READONLY

Start    MOV R0, #10      ; Label and processor instruction
          MOV R1, #3       ; Another processor instruction
stop     B  stop        ; Label and another processor instruction
          END            ; Assembler directive signifying the end of the file
```

26.64 SETS

{name} SETS string

The SETS directive sets a string value to a variable given by `{name}`.

- `string` is the string the variable is set to.

26.64.1 Example

Set the variable `my_text` to the string, "This is my text".

```
my_text SETS "This is my text"
```

26.65 EQU

{name} EQU number{, type}

The EQU directive sets a number value to a variable given by `{name}`.

- `number` is the number the variable is set to.
- `type` is optional, and it refers to the type of the number given to the EQU directive.

26.65.1 Example

Set the variable `temperature` to the number 25.6.

```
temperature EQU 25.6
```

26.66 RN

name RN expression

The RN directive defines a name for a specified register.

- name is the name to be assigned to the register. name cannot be the same as any of the predefined names.
- expr is either a register like R0, or a register number from 0 to 15.

26.66.1 Example

Define the name regname for the register r11.

```
regname RN 11
```

26.67 MACRO and MEND

MACRO

```
{$label}      macro_name{$condition} {$parameter_1{, $parameter_2}...}
; Code for the macro
MEND
```

- The MACRO directive signifies the start of the definition of a macro.
- Likewise, the MEND directive signifies the end of the definition of a macro.
- Take note that the \$ characters are required in a macro definition.
- The label is the label for the start of the macro.
- {condition} is an optional condition suffix. Refer to the condition code suffixes ([page 480](#)) for more information on the available suffixes and what they mean.
- parameter_1, parameter_2, ... are the parameters the macro can take.

26.67.1 Example 1

MACRO

```
; Macro definition
; var_a = 8 * (var_b + var_c + 6)
$Label_1      AddMul $var_a, $var_b, $var_c

$Label_1
; Add var_a, var_b and var_c together
; and store the result in var_a
ADD $var_a, $var_b, $var_c

; Add 6 to var_a
ADD $var_a, $var_a, #6

; Bitwise shift var_a left by 3 bits.
; This basically multiplies var_a by 8,
; since 2^3 is 8.
LSL $var_a, $var_a, #3
```

MEND

When the above macro is invoked using:

```
AddMul R0, R1, R2
```

It expands into:

```
ADD R0, R1, R2
ADD R0, R1, #6
LSL R0, R1, #3
```

26.67.2 Example 2

Macro definition:

```
MACRO ; Start of macro definition
$label xmac $p1, $p2
; Code
$label.loop1 ; Code
; Code
BGE $label.loop1
$label.loop2 ; Code
BL $p1
BGT $label.loop2
; Code
ADR $p2
MEND ; End of macro definition
```

When the macro is invoked, it expands into:

```
abc xmac subr1,de ; Macro invocation
; Below this line is what the macro is expanded into
; Code
abcloop1 ; Code
; Code
BGE abcloop1
abcloop2 ; Code
BL subr1
BGT abcloop2
; Code
ADR de
; Code
```

26.68 AREA

```
AREA section_name{, attribute_1, attribute_2, ...}
```

The AREA directive instructs the assembler to assemble a new code or data section.

- `section_name` is the name of the section, which can be anything.
- `attribute_1, attribute_2, ...` are the section attributes in ARM.

26.68.1 Valid section attributes

Attribute	Description
ALIGN= <i>expr</i>	This aligns a section on a 2 <i>expr</i> -byte boundary. Note that this is different from the ALIGN directive. For example, if <i>expr</i> = 10, then the section is aligned to a 1 KB boundary.
CODE	This section is machine code (READONLY is the default).
DATA	This section is data (READWRITE is the default).
READONLY	This section can be placed in read-only memory (default for sections of CODE).
READWRITE	This section can be placed in read-write memory (default for sections of DATA).

26.68.2 Example

Defines new code (CODE) section called `main` which is read-only (READONLY).

```
AREA main, CODE, READONLY
```

26.69 ENTRY and END

ENTRY

```
; Code  
END
```

- The ENTRY directive specifies the entry point of the program.
- In simpler terms, it tells the computer which part of the code to start executing from when running the program.
- All programs must have an entry point.
- If the program contains multiple entry points, you must select one of them by exporting the symbol for the ENTRY directive that you want to use as the entry point.

26.69.1 Example

```
AREA      ARMEx, CODE, READONLY  
  
; Entry point for the application  
ENTRY  
  
; Export the symbol so the linker can find it  
; in the object file.  
EXPORT Start  
Start  
; Code  
END
```

26.70 DCB (Declare 8-bit bytes)

{label} DCB{U} value_1{, value_2, value_3, ...}

- The DCB directive allocates one or more bytes (8-bit value for ARM), aligned on 1-byte (8-bit) boundaries.
- It also initialises the memory with the value given to it, value_1.
- {, value_2, value_3, ...} are additional values to be initialised.
- {label} is an optional label to place the initialised memory in.
- {U} is an optional suffix that tells the assembler to **not** align the memory on 1-byte (8-bit) boundaries.

26.70.1 Example 1

Define 3 bytes (8-bit values) in the data1 section, the first containing the decimal value 1, the next containing the decimal value 5, and the last one containing the decimal value 20.

```
data1 DCB 1, 5, 20
```

26.70.2 Example 2

Define 1 byte (8-bit value) in the data2 section, containing 4 + the address of the label mem06. Essentially, the bytes initialised will contain the memory address in the label mem06 offset by 4.

```
data2 DCB mem06 + 4
```

26.70.3 Example 3

Define 3 bytes (8-bit values) in the data1 section, the first containing the decimal value 1, the next containing the decimal value 5, and the last one containing the decimal value 20. However, the memory is not aligned to bytes boundaries (1-byte or 8-bit boundaries).

```
data3 DCB 1, 5, 20
```

26.71 DCW (Declare 16-bit half-words)

```
{label} DCW{U} value_1{, value_2, value_3, ...}
```

- The DCW directive allocates one or more half-words (16-bit value for ARM), aligned on 2-byte (16-bit) boundaries.
- It also initialises the memory with the value given to it, value_1.
- {, value_2, value_3, ...} are additional values to be initialised.
- {label} is an optional label to place the initialised memory in.
- {U} is an optional suffix that tells the assembler to **not** align the memory on 2-byte (16-bit) boundaries.

26.71.1 Example 1

Define 3 half-words (16-bit values) in the data1 section, the first containing the decimal value 1, the next containing the decimal value 5, and the last one containing the decimal value 20.

```
data1 DCW 1, 5, 20
```

26.71.2 Example 2

Define 1 half-word (16-bit value) in the data2 section, containing 4 + the address of the label mem06. Essentially, the half-words initialised will contain the memory address in the label mem06 offset by 4.

```
data2 DCW mem06 + 4
```

26.71.3 Example 3

Define 3 half-words (16-bit values) in the data1 section, the first containing the decimal value 1, the next containing the decimal value 5, and the last one containing the decimal value 20. However, the memory is not aligned to half-words boundaries (2-byte or 16-bit boundaries).

```
data3 DCW 1, 5, 20
```

26.72 DCD (Declare 32-bit words)

```
{label} DCD{U} value_1{, value_2, value_3, ...}
```

- The DCD directive allocates one or more words (32-bit value for ARM), aligned on 4-byte (32-bit) boundaries.
- It also initialises the memory with the value given to it, value_1.
- {, value_2, value_3, ...} are additional values to be initialised.
- {label} is an optional label to place the initialised memory in.
- {U} is an optional suffix that tells the assembler to **not** align the memory on 4-byte (32-bit) boundaries.

26.72.1 Example 1

Define 3 words (32-bit values) in the `data1` section, the first containing the decimal value 1, the next containing the decimal value 5, and the last one containing the decimal value 20.

```
data1 DCD 1, 5, 20
```

26.72.2 Example 2

Define 1 word (32-bit value) in the `data2` section, containing `4 + the address of the label mem06`. Essentially, the words initialised will contain the memory address in the label `mem06` offset by 4.

```
data2 DCD mem06 + 4
```

26.72.3 Example 3

Define 3 words (32-bit values) in the `data1` section, the first containing the decimal value 1, the next containing the decimal value 5, and the last one containing the decimal value 20. However, the memory is not aligned to words boundaries (4-byte or 32-bit boundaries).

```
data3 DCD 1, 5, 20
```

26.73 ALIGN

```
ALIGN {byte_boundary_size{, offset}}
```

The ALIGN directive aligns the current location within the code to a word (4-byte or 32-bit) boundary.

- byte_boundary_size is the size of the next byte boundary to align the current location on. It can be any power of 2 from 2 to 2^{31} , and the current location will be aligned to the next 2^n -byte boundary. If this parameter is not given, the instruction location will be set to the next word boundary.
- offset is optional, and is the byte offset from the alignment given in byte_boundary_size.
- Note that the current location in bytes is given by ($\text{offset} + n \times \text{byte_boundary_size}$), where n is the constant needed to get to the next byte_boundary_size boundary

26.73.1 Example 1

```
AREA cacheable, CODE, ALIGN=3
rout1 ; Code      ; Aligned on 8-byte boundary
       ; Code
       MOV pc, lr ; Aligned only on 4-byte boundary
       ALIGN 8    ; Now aligned on 8-byte boundary
rout2 ; Code
```

26.73.2 Example 2

```
AREA OffsetExample, CODE

; Place 1 byte at the start.
; Let's call this byte 1.
DCB 1

; Offset 3 bytes from the current location,
; then align on the next 4-byte boundary.
ALIGN 4, 3

; Current location is now 3 bytes away
; from the very first byte or byte 1.
;
; The first 3 bytes come from the offset given
; to the ALIGN directive, which is 3-bytes away
; from the boundary, so the current location is at byte 4.
;
; Byte 4 is the start of a 4-byte boundary,
; so it is already at the next word boundary to align on,
; and there is nothing else to do.
; There is no need to move to the next 4-byte boundary.
;
; Hence, the second byte from the second DCB directive
; is placed at byte 4, or 3 bytes away from the very first byte.
DCB 1
```

Below is an image to illustrate the alignment.



26.74 SPACE

`{label} SPACE bytes`

The SPACE directive reserves a zeroed block of memory, which is basically just setting the block of memory to 0.

- `{label}` is an optional label to place the memory in.
- `bytes` is the number of bytes of memory to zero.

26.74.1 Example

Fill 255 bytes of memory with 0.

`SPACE 255`

26.75 FILL

`{label} FILL bytes{, value{, value_size_in_bytes}}`

The FILL directive is similar to the SPACE directive ([page 518](#)), but it fills the memory with the given value instead of zero.

- `{label}` is an optional label to place the memory in.
- `bytes` is the number of bytes of memory to fill up with the `value`.
- `value` is the value used to fill the memory with, and it is optional as it will default to 0 if not given.
- `value_size_in_bytes` is the size of the `value` given in bytes, and is optional.

26.75.1 Example

Fill 50 bytes of memory with the hexadecimal value 0xAB, which is 1 byte in size.

`FILL 50, 0xAB, 1`

26.76 LTORG (Lookup table organised)

LTORG

- The LTORG directive instructs the assembler to assemble the current literal pool immediately.
- The assembler assembles the current literal pool at the end of every code section.
- The end of a code section is determined by the AREA ([page 513](#)) directive at the beginning of the following section, or the end of the assembly.
- A literal pool is a lookup table used to hold literals during assembly and execution.
- The assembler uses literal pools to store some constant data in code sections.
- The literal pool is part of the working memory of a function, but only the constant data in the function.
- The purpose of this function is because some code sections are very long, so the memory addresses to those constants might be out of range of the LDR instruction ([page 482](#)), so the LTORG directive is used to keep those memory addresses in range.

26.76.1 Example

```
AREA    Example, CODE, READONLY
start   BL      func1
func1
; Function body
; Code

; Load into register R1, the memory address of the first literal pool
; LDR R1, [pc, #offset to Literal Pool 1]
LDR     R1,=0x55555555
; Code
MOV     pc,lr      ; End function

; Literal Pool 1 contains literal &55555555.
LTORG

; Clears 4200 bytes of memory starting at current location.
data   SPACE 4200

; Default literal pool is empty.
END
```

26.77 EXPORT or GLOBAL

`EXPORT symbol {[qualifier{}, qualifier}{, qualifier}[]}`

- The `EXPORT` directive declares a symbol that can be used by the linker to resolve symbol references in separate object and library files.
- `GLOBAL` is a synonym for `EXPORT`.
- `symbol` is the symbol name to export, which is case-sensitive.
- `qualifier` can be any of:
 - ▶ `FPREGARGS`, which means that `symbol` refers to a function that expects floating-point arguments to be passed in floating-point registers.
 - ▶ `DATA`, which means that `symbol` refers to a data location rather than a function or a procedure entry point.
 - ▶ `LEAF`, which denotes that the exported function is a leaf function that calls no other functions.
This qualifier is obsolete.

26.77.1 Example

Export the function name `DoAdd` to be used by external modules.

```
AREA   Example, CODE, READONLY
EXPORT DoAdd
DoAdd ADD    R0, R1, R2
```

26.78 IMPORT or EXTERN

`IMPORT symbol {[qualifier{}, qualifier}[]}`

- The `IMPORT` directive provides the assembler with a name that is not defined in the current assembly.
- `EXTERN` is a synonym for `IMPORT`.
- `symbol` is a symbol name defined in a separately assembled source file, object file, or library. The symbol name is case-sensitive.
- `qualifier` can be any of:
 - ▶ `FPREGARGS`, which means that `symbol` defines a function that expects floating-point arguments passed in floating-point registers.
 - ▶ `WEAK`, which prevents the linker from generating an error message if the symbol is not defined elsewhere. It also prevents the linker from searching libraries that are not already included.

26.78.1 Example

Test if a C++ library has been linked, and branches conditionally on the result.

```
AREA   Example, CODE, READONLY
IMPORT __CPP_INITIALISE[WEAK]

LDR R0, __CPP_INITIALISE

CMP R0, #0
BEQ nocplusplus
```

26.79 GET or INCLUDE

GET filename

- The GET directive includes a file within the file being assembled. The included file is assembled.
- INCLUDE is a synonym for GET.
- filename is the name of the file to be included in the assembly. The assembler accepts path names in either UNIX or MS-DOS format.

26.79.1 Example

Includes file_1.s if it exists in the current place, and also include file_2.s from the path given.

```
AREA    Example, CODE, READONLY
GET      file_1.s

GET      C:\Project\file_2.s
```

26.80 Other directives offered by ARM Keil

Directive	Description
A:MOD:B	A modulo B or A % B.
A:ROL:B	Rotate A left by B bits.
A:ROR:B	Rotate A right by B bits.
A:SHL:B or A << B	Shift A left by B bits.
A:SHR:B or A >> B	Shift A right by B bits.
A + B	Add A to B
A - B	Subtract B from A
A:AND:B	Bitwise AND of A and B
A:EOR:B	Bitwise Exclusive OR of A and B
A:OR:B	Bitwise OR of A and B