JN5189(T)/JN5188(T) User Manual

UM11138

Rev. 1.4 — June 2020 **User manual**

Document information

Info	Content
Keywords	Arm Cortex-M4, microcontroller, Zigbee, Thread
Abstract	JN5189 User Manual

Revision history

Rev	Date	Description
1.4	202006	• Added to support Thread as well as Zigbee across the whole book. • Corrected the Tag of JN5189T/JN5188T to be NT3H2211. • Updated SPIFI feature in the Section 1.2.1 “Microcontroller features”. • Updated to “1.9 V to 3.6 V supply voltage” in the Section 1.2.2 “Radio features”. • Updated Table 3 “Pin descriptions” to correct typos and make the descriptions aligned with the Figure 3 “Pinout diagram”. • Updated the Range of MAIN_CLK to be “12-48 MHz” in the Table 56 “Base clock sources for DMIC interface peripheral”. • Updated SPIFI features in the Section 34.2 “Features”, the Remark in the Table 64 “Available pins and configuration registers”, and Section 34.8.4 “SPIFI intermediate data register”, Section 34.9.2 “Software requirements and capabilities”. • Updated the ISP usage restrictions in the Section 38.8 “Usage restrictions”. • Updated the Section 41.7.6.2 “Memory mapping”, descriptions in the Section 41.9 “Antenna Diversity”. • Updated Section 42.5 “General description”. • Updated to reserve the CT32B_IR[7:6].
1.3	202002	Initial public release

Contact information

For more information, please visit: **http://www.nxp.com**

For sales office addresses, please send an email to: **salesaddresses@nxp.com**

1.1 Introduction

The JN5189(T) and JN5188(T) (called JN5189(T)/JN5188(T) throughout this document) are ultra-low power, high performance Arm® Cortex®-M4 based wireless microcontrollers supporting Zigbee 3.0 and Thread networking stacks to facilitate the development of home automation, smart lighting and wireless sensor network applications.

The JN5189(T)/JN5188(T) includes a 2.4 GHz IEEE 802.15.4 and a comprehensive mix of analog and digital peripherals. Ultra-low current consumption in radio receive and radio transmit modes allows use of coin cell batteries.

JN5189(T) has 640 KB embedded Flash, 152 KB RAM and 128 KB ROM memory. The embedded flash can support Over The Air (OTA) code download to applications. The devices include 10−channel PWM, two timers, one RTC/alarm timer, a Windowed Watchdog Timer (WWDT), two USARTs, two SPI interfaces, two I^2C interfaces, a DMIC subsystem with dual-channel PDM microphone interface with voice activity detector, one 12-bit ADC, temperature sensor and comparator.

The JN5189T/JN5188T variant has an internal NTAG I^2C plus NFC tag and connections for the external NFC antenna. The tag is an NXP device NT3H2211.

The JN5188 variant has the same functionality as the JN5189 except for reduced memory sizes of 320 KB embedded Flash, 88 KB RAM. The JN5188T variant is that it has the functionality of the JN5188 with the addition of an embedded NTAG I^2C plus NFC tag.

The Arm Cortex-M4 is a 32-bit core that offers system enhancements such as low-power consumption and enhanced debug features. The Arm Cortex-M4 CPU, operating at up to 48 MHz, incorporates a 3-stage pipeline, uses a Harvard architecture with separate local instruction and data buses as well as a third bus for peripherals, and includes an internal prefetch unit that supports speculative branching. The Arm Cortex-M4 supports single-cycle digital signal processing and SIMD instructions. Debug is supported using the Serial Wire Debug.

Refer to the data sheet for complete details on specific products and configurations.

1.2 Features

1.2.1 Microcontroller features

- Application CPU, Arm Cortex-M4 CPU:
  - Arm Cortex-M4 processor, running at a frequency of up to 48 MHz.
  - Arm built-in Nested Vectored Interrupt Controller (NVIC)
  - Memory Protection Unit (MPU)
  - Non-maskable Interrupt (NMI) with a selection of sources
  - Serial Wire Debug (SWD) with 8 breakpoints and 4 watchpoints
  - System tick timer

  - Includes Serial Wire Output for enhanced debug capabilities.
- On-Chip memory:
  - 640 KB flash (320 KB for JN5188)
  - 152 KB SRAM (88 KB for JN5188)
  - 128 KB ROM
- 12 MHz to 48 MHz system clock speed for low-power
- 2 x I^2C-bus interface, operate as either master or slave
- 10 x PWM
- 2 x Low-power timers
- 2 x USART, one with flow control
- 2 x SPI-bus, master or slave
- 1 x PDM digital audio interface with a hardware based voice activity detector to reduce power consumption in voice applications. Support for dual-channel microphone interface, flexible decimators, 16 entry FIFOs and optional DC blocking
- 19-channel DMA engine for efficient data transfer between peripherals and SRAM, or SRAM to SRAM. DMA can operate with fixed or incrementing addresses. Operations can be chained together to provide complex functionality with low CPU overhead
- Up to four GPIOs can be selected as pin interrupts (PINT), triggered by rising, falling or both input edges
- Two GPIO grouped interrupts (GINT) enable an interrupt based on a logical (AND/OR) combination of input states.
- 32-bit Real Time clock (RTC) with 1 s resolution. A timer in the RTC can be used to wake from Sleep, Deep-sleep and Power-down, with 1 ms resolution
- Voltage Brown Out with 8 programmable thresholds
- 8-input 12-bit ADC, 190 kS/sec. HW support for continuous operation or single conversions, single or multiple inputs can be sampled within a sequence. DMA operation can be linked to achieve low overhead operation.
- 1 x analog comparator
- Battery voltage measurement
- Temperature sensor
- Watchdog timer and POR
- Standby power controller
- Up to 22 Digital IOs (DIO)
- 1 x Quad SPIFI for reading or writing to external flash device
- NTAG NFC Forum Type 2 on JN5189T and JN5188T only
- Random Number Generator engine
- AES engine
- Hash hardware accelerator
- EFuse :
  - 128-bit random AES key
  - configuration modes

- Trimming

1.2.2 Radio features

- 2.4 GHz IEEE 802.15.4 compliant
- Receive current 4.3 mA
- IEEE 802.15.4 receiver sensitivity -100 dBm
- Improved co-existence with WiFi
- Flexible output power up to 11 dBm, programmable with 46 dB range
- Transmit power +10 dBm current 20.28 mA
- Transmit power +3 dBm current 9.44 mA
- Transmit power 0 dBm current 7.36 mA
- 1.9 V to 3.6 V supply voltage
- 32 MHz XTAL cell with internal capacitors, able with suitable external XTAL, to meet the required accuracy for radio operation over the operating conditions
- Antenna Diversity control
- Integrated RF balun
- Integrated ultra Low-power sleep oscillator
- Deep Power-down current 350 nA (with wake-up from IO)
- 128-bit or 256-bit AES security processor
- MAC accelerator with packet formatting, CRCs, address check, auto-acks, timers

1.2.3 Low-power features

- Sleep mode supported, CPU in low-power state waiting for interrupt
- Deep-sleep mode supported, CPU in low-power state waiting for interrupt, but extra functionality disabled or in low-power state compared to sleep mode
- Power Down mode, main functionality powered down, wakeup possible from IOs, wakeup possible from some peripherals (I^2C, USART, SPI) in a limited function mode and low-power timers
- Deep power down, very low-power state with option of reset triggered by IOs, 350 nA
- 41-bit and 28-bit Low power timers can run in power down mode, clocked by 32 kHz FRO or 32 kHz XTAL. Timers can run for over one year or 2 days

1.3 Block diagram

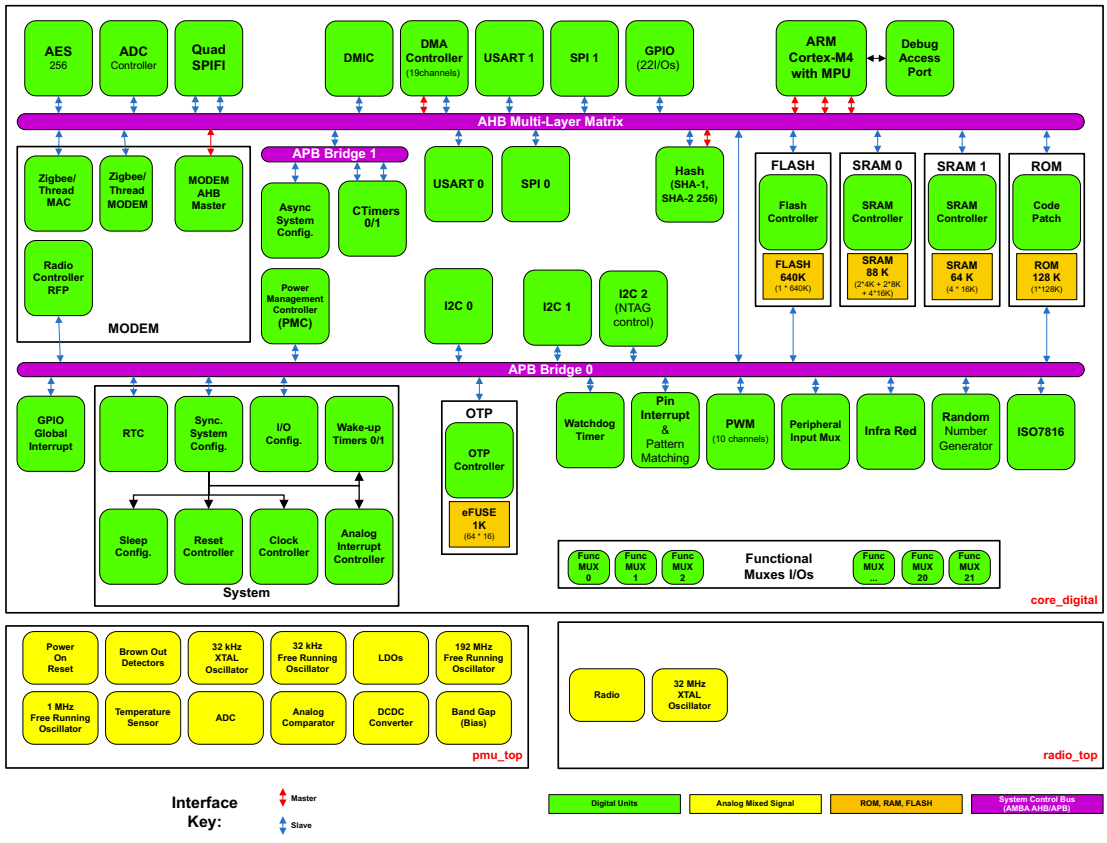


Fig 1. Chip block diagram

1.4 Architectural overview

The Arm Cortex-M4 includes three AHB-Lite buses, one system bus and the I-code and D-code buses. One bus is dedicated for instruction fetch (I-code), and one bus is dedicated for data access (D-code). The use of two core buses allows for simultaneous operations if concurrent operations target different devices.

A multi-layer AHB matrix connects the CPU buses and other bus masters to peripherals in a flexible manner that optimizes performance by allowing peripherals on different slaves ports of the matrix to be accessed simultaneously by different bus masters. More information on the multilayer matrix can be found in Section 2.1.3 “AHB multilayer matrix”. Connections in the multilayer matrix are shown in Figure 1. Note that while the AHB bus itself supports word, halfword, and byte accesses, not all AHB peripherals need or provide that support.

APB peripherals are connected to the AHB matrix via two APB buses using separate slave ports from the multilayer AHB matrix. This allows for better performance by reducing collisions between the CPU and the DMA controller, and also for peripherals on the asynchronous bridge to have a fixed clock that does not track the system clock. Note that APB, by definition, does not directly support byte or halfword accesses.

The multi-layer has several master interfaces and slave interfaces. The following table shows which slaves the master interfaces are able to access; indicated with a 'x'. Where a block has '-REG' in the name, this shows that it is the register interface of the block. SPIFI-MEM is the direct memory access function of the block and not the register interface.

Table 1. AHB master interface accessibility to the slaves

AHB slave		AHB master					
		CPU			DMA	HASH	MODEM
		I-code	D-code	System			
Slave port 0	Flash	x	x		x	x	
Slave port 1	ROM	x	x		x		
Slave port 2	SRAM0	x	x		x	x	x
Slave port 3	SRAM1	x	x		x	x	x
Slave port 4	SPIFI-MEM		x		x		
Slave port 5	APB bridge 0			x	x		
Slave port 6	APB bridge 1			x			
Slave port 7	SPIFI-REG			x	x		
	GPIO			x	x		
	DMA-REG			x	x		
	AES			x	x		
	ADC			x	x		
	DMIC			x	x		
	USART0			x	x		
	USART1			x	x		
	SPI0			x	x		
	SPI1			x	x		
	HASH			x	x		
Slave port 8	Zigbee/Thread MODEM			x	x		
	Zigbee/Thread MAC			x	x		

1.5 Arm Cortex-M4 processor

The Cortex-M4 is a general purpose 32-bit microprocessor, which offers high performance and very low-power consumption. The Cortex-M4 offers a Thumb-2 instruction set, low interrupt latency, interruptible/continuable multiple load and store instructions, automatic state save and restore for interrupts, tightly integrated interrupt controller, and multiple core buses capable of simultaneous accesses.

A 3-stage pipeline is employed so that all parts of the processing and memory systems can operate continuously. Typically, while one instruction is being executed, its successor is being decoded, and a third instruction is being fetched from memory.

Information about Cortex-M4 configuration options can be found in Chapter 43 “Arm Cortex-M4 Appendix”.

2.1 General description

The JN5189(T)/JN5188(T) incorporates several distinct memory regions. Figure 2 shows the overall map of the entire address space from the user program viewpoint following reset.

The APB peripheral area (detailed in Figure 2) is divided into fixed 4 KB slots to simplify addressing.

The registers incorporated into the CPU, such as NVIC, SysTick, and sleep mode control, are located on the private peripheral bus.

2.1.1 Main SRAM

The main SRAM is composed of 152 KB of on-chip static RAM memory. The SRAM is accessed through two controllers SRAM-CTRL0 and SRAM-CTRL1. SRAM-CTRL0 gives access to the first 88 KB and SRAM-CTRL1 gives access to the remaining 64 KB. In a JN5188 device, SRAM-CTRL1 is held in reset to prevent access to this memory. The memory is contiguous within the two separate regions but not contiguous from SRAM-CTRL0 to SRAM-CTRL1. Each SRAM has a separate clock control and power switch.

Table 2. SRAM configuration

SRAM Controller	SRAM-CTRL0	SRAM-CTRL1
(total main SRAM = up to 152 KB)		
Size	88 KB	Up to 64 KB
Address range	0x0400 0000 to 0x0401 5FFF	0x0402 0000 to 0x0402 FFFF

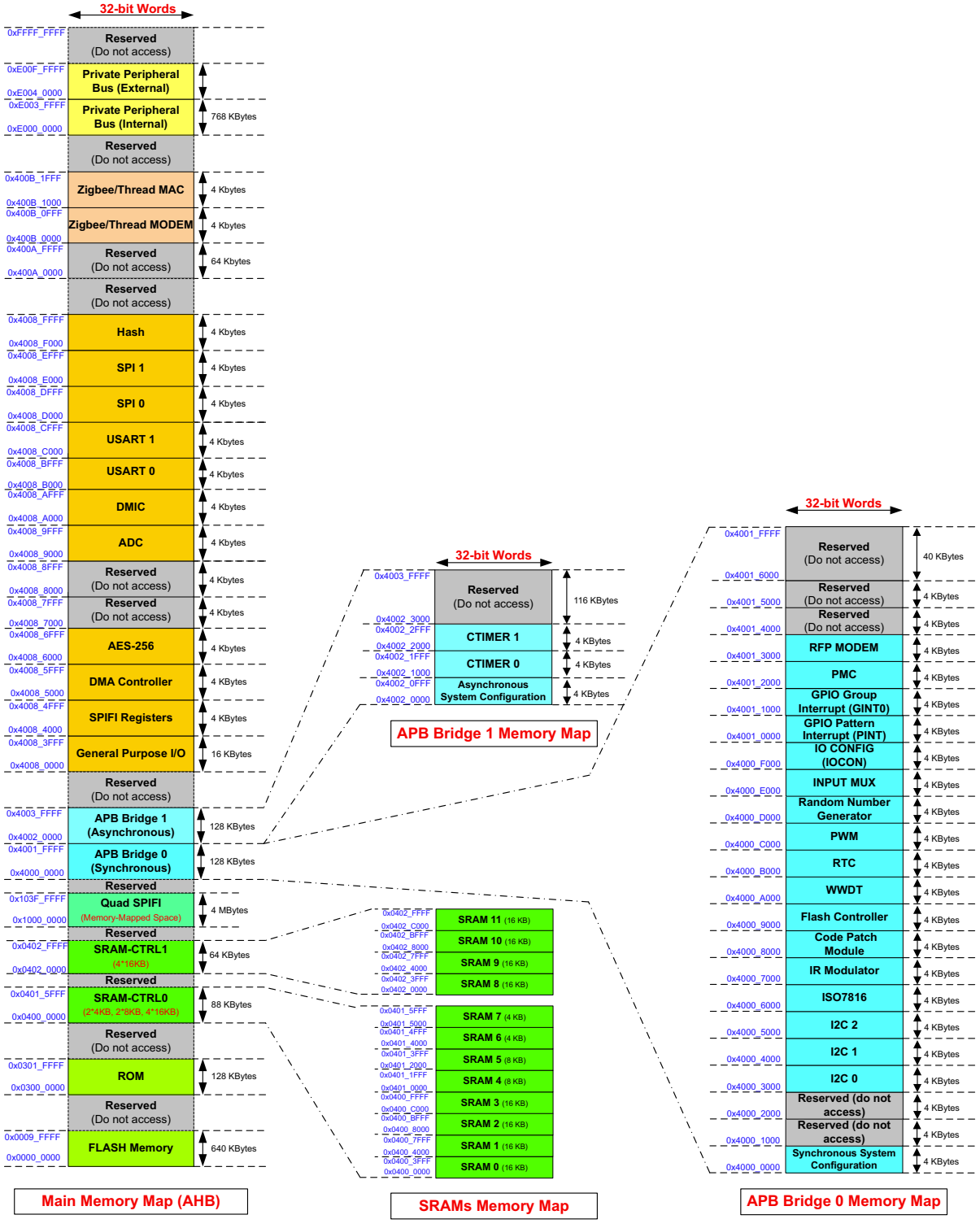
2.1.1.1 SRAM usage notes

The SRAM controllers, SRAM-CTRL0 and SRAM-CTRL1, are placed on different AHB matrix ports. This allows user programs to potentially obtain better performance by dividing RAM usage among the ports. For example, simultaneous access to SRAM0 by the CPU and SRAM1 by the system DMA controller does not result in any bus stalls for either master.

Generally, data being communicated via peripherals will be accessed by the CPU at some point, even when peripheral data is mainly being transferred via DMA. So, in order to minimizing data read/write stalls, data buffers may be placed in RAMs on different AHB matrix ports. For instance, if DMA is writing to one buffer on a specific AHB matrix port while the CPU is reading data from a buffer on a different AHB matrix port, there is no stall for either the CPU or the DMA. Sequences of data from the same peripheral could be alternated between RAM on each port. This could be helpful if DMA fills or empties a RAM buffer, then signals the CPU before proceeding on to a second buffer. The CPU would then tend to access the data while the DMA is using the other RAM.

2.1.2 Memory mapping

The overall memory map and also details of the APB peripheral mapping are shown in Figure 2 “Main memory map”.



1) The private peripheral bus includes CPU peripherals such as the NVIC, SysTick, and the core control registers.

2) The total size of flash and SRAM is part dependent, see the ordering information in the specific device data sheet for details.

3) Memory region 0x00000000 to 0x000001FF can be mapped to flash, as shown here, or ROM or RAM depending upon the Vector Table Remapping setting.

Fig 2. Main memory map

2.1.3 AHB multilayer matrix

The JN5189(T)/JN5188(T) uses a multi-layer AHB matrix to connect the CPU buses and other bus masters to peripherals in a flexible manner that optimizes performance by allowing peripherals that are on different slave ports of the matrix to be accessed simultaneously by different bus masters. Figure 2 and shows details of the potential matrix connections.

2.1.4 Memory Protection Unit (MPU)

The Cortex-M4 processor has a memory protection unit (MPU) that provides fine grain memory control, enabling applications to implement security privilege levels, separating code, data and stack on a task-by-task basis. Such requirements are critical in many embedded applications.

The MPU register interface is located on the private peripheral bus and is described in detail in Ref. 1 “Cortex-M4 TRM”.

2.1.5 Vector table remapping

The Cortex boot address is 0x00000000. The memory map in Figure 2 shows flash at this address. However, the first 512 bytes in the memory map are treated specially to give flexibility to the location of the vector table. This region will be referred to here as the Vector Table Region.

The MEMORYREMAP MAP field is used to indicate if this Vector Table Region is located in ROM, Flash or RAM. Depending upon this setting, cortex address 0x00000000 to 0x000001FF will either access the bottom of the ROM, the bottom of the Flash or the bottom of SRAM0.

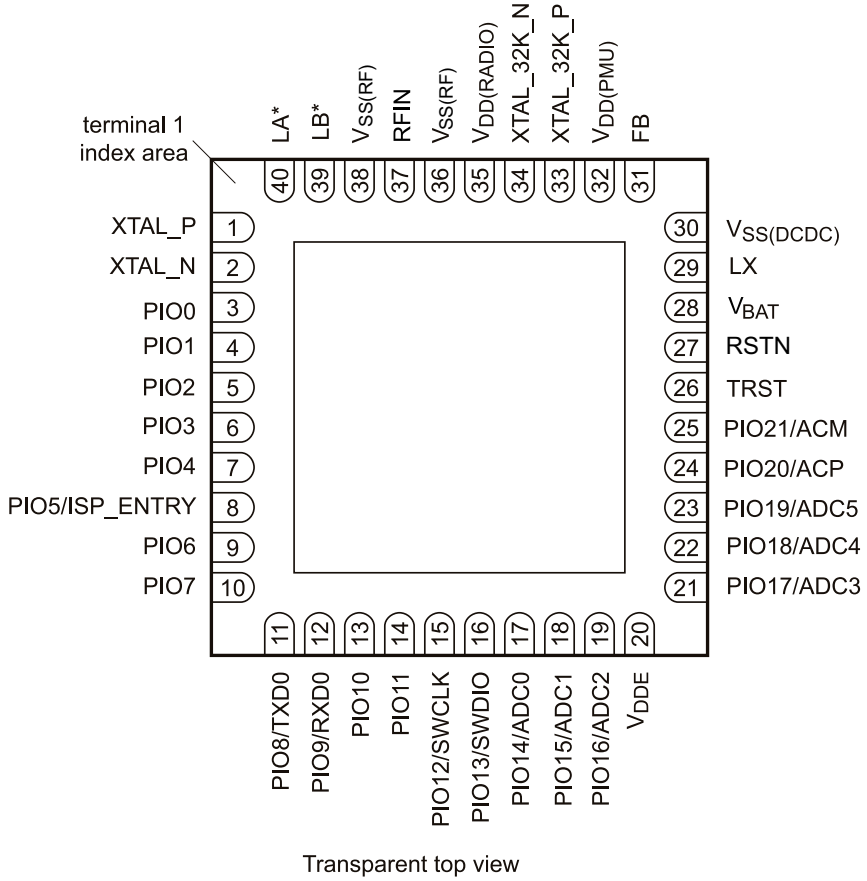
The default / reset condition is to use ROM code so that a known boot sequence is followed. Typically, during the boot sequence, this setting is changed so that the vector table region is in Flash. This allows application specific interrupt vectors to be used.

3.1 How to read this chapter

JN5189(T)/JN5188(T) package is a HVQFN40 (6x6 mm). The pinout and IO cells are consistent across all product types except for the NTAG antenna connections.

There are some functional differences between the IO cells used for different digital pins and these are presented in this chapter.

3.2 Pinout diagram



Transparent top view

* For JN5188HN and JN5189HN (without NTAG), it is N.C.

Fig 3. Pinout diagram

3.3 Pinout signaling descriptions

Table 3. Pin descriptions

Symbol	Pin	Type	Default at reset	Description
XTAL_P	1			System crystal oscillator 32 MHz
XTAL_N	2			System crystal oscillator 32 MHz
PIO0	3	IO	GPIO0[1]	**GPIO0 —** General Purpose digital Input/Output 0
				USART0_SCK — Universal Synchronous/Asynchronous Receiver/Transmitter 0 - synchronous clock
				USART1_TXD — Universal Synchronous/Asynchronous Receiver/Transmitter 1 - transmit data output
				PWM0 — Pulse Width Modulator output 0
				SPI1_SCK — Serial Peripheral Interface-bus 1 clock input/output
				PDM0_DATA — Pulse Density Modulation Data input from digital microphone (channel 0)
PIO1	4	IO	GPIO1[1]	**GPIO1 —** General Purpose digital Input/Output 1
				USART1_RXD — Universal Synchronous/Asynchronous Receiver/Transmitter 1 - receive data input
				PWM1 — Pulse Width Modulator output 1
				SPI1_MISO — Serial Peripheral Interface-bus 1 master data input
				PDM0_CLK — Pulse Density Modulation Clock output to digital microphone (channel 0)
PIO2	5	IO	GPIO2[1]	**GPIO2 —** General Purpose digital Input/Output 2
				SPI0_SCK — Serial Peripheral Interface-bus 0 clock input/output
				PWM2 — Pulse Width Modulator output 2
				SPI1_MOSI — Serial Peripheral Interface-bus 1 master output slave input
				USART0_RXD — Universal Synchronous/Asynchronous Receiver/Transmitter 0 - receive data input
				ISO7816_RST — RST signal, output, for ISO7816 interface
				MCLK — External clock, can be provided to DMIC IP
PIO3	6	IO	GPIO3[1]	**GPIO3 —** General Purpose digital Input/Output 3
				SPI0_MISO — Serial Peripheral Interface-bus 0 master input
				PWM3 — Pulse Width Modulator output 3
				SPI1_SSELN0 — Serial Peripheral Interface-bus 1 slave select not 0
				USART0_TXD — Universal Synchronous/Asynchronous Receiver/Transmitter 0 - transmit data output
				ISO7816_CLK — Clock output for ISO7816 interface

Table 3. Pin descriptions

Symbol	Pin	Type	Default at reset	Description
PIO4	7	IO	GPIO4[1][2]	**GPIO4 —** General Purpose digital Input/Output 4
				SPI0_MOSI — Serial Peripheral Interface-bus 0 master output slave input
				PWM4 — Pulse Width Modulator output 4
				SPI1_SSELN1 — Serial Peripheral Interface-bus 1 slave select not 1
				USART0_CTS — Universal Synchronous/Asynchronous Receiver/Transmitter 0 - Clear To Send input
				ISO7816_IO — IO of ISO7816 interface
				RFTX — Radio Transmit Control Output
				ISP_SEL — In-System Programming Mode Selection
PIO5/ISP_ENTRY	8	IO	GPIO5/ISP_ENTRY[1][3]	**GPIO5/ISP_ENTRY —** General Purpose digital Input/Output 5; In-System Programming Entry
				SPI0_SSELN — Serial Peripheral Interface-bus 0 slave select not
				SPI1_MISO — Serial Peripheral Interface-bus 1 master data input
				SPI1_SSELN2 — Serial Peripheral Interface-bus 1 slave select not 2
				USART0_RTS — Universal Synchronous/Asynchronous Receiver/Transmitter 0 - Request To Send output
				RFRX — Radio Receiver Control Output
PIO6	9	IO	GPIO6[1]	**GPIO6 —** General Purpose digital Input/Output 6
				USART0_RTS — Universal Synchronous/Asynchronous Receiver/Transmitter 0 - Request to Send output
				CT32B1_MAT0 — 32-bit CT32B1 match output 0
				PWM6 — Pulse Width Modulator output 6
				I2C1_SCL — I^2C-bus 1 master/slave SCL input/output
				USART1_TXD — Universal Synchronous/Asynchronous Receiver/Transmitter 1 - transmit data output
				ADE — Antenna Diversity Even output
				SPI0_SCK — Serial Peripheral Interface 0- synchronous clock
PIO7	10	IO	GPIO7[1]	**GPIO7 —** General Purpose digital Input/Output 7
				USART0_CTS — Universal Synchronous/Asynchronous Receiver/Transmitter 0 - Clear to Send input
				CT32B1_MAT1 — 32-bit CT32B1 match output 1
				PWM7 — Pulse Width Modulator output 7
				I2C1_SDA — I^2C-bus 1 master/slave SDA input/output
				USART1_RXD — Universal Synchronous/Asynchronous Receiver/Transmitter 1 - receive data input
				ADO — Antenna Diversity Odd Output
				SPI0_MISO — Serial Peripheral Interface-bus 0 master input

Table 3. Pin descriptions

Symbol	Pin	Type	Default at reset	Description
PIO8/TXD0	11	IO	GPIO8[1][4]	**GPIO8 —** General Purpose digital Input/Output 8
				USART0_TXD — Universal Synchronous/Asynchronous Receiver/Transmitter 0 - transmit data output
				CT32B0_MAT0 — 32-bit CT32B0 match output 0
				PWM8 — Pulse Width Modulator output 8
				ANA_COMP_OUT — Analog Comparator digital output
				PDM1_DATA — Pulse Density Modulation Data input from digital microphone (channel 1)
				SPI0_MOSI — Serial Peripheral Interface-bus 0 master output slave input
				RFTX — Radio Transmit Control Output
PIO9/RXD0	12	IO	GPIO9[1][5]	**GPIO9 —** General Purpose digital Input/Output 9
				USART0_RXD — Universal Synchronous/Asynchronous Receiver/Transmitter 0 - receive data input
				CT32B1_CAP1 — 32-bit CT32B1 capture input 1
				PWM9 — Pulse Width Modulator output 9
				USART1_SCK — Universal Synchronous/Asynchronous Receiver/Transmitter 1 - synchronous clock
				PDM1_CLK — Pulse Density Modulation Clock output to digital microphone (channel 1)
				SPI0_SSELN — Serial Peripheral Interface-bus 0 slave select not
				ADO — Antenna Diversity Odd Output
PIO10	13	IO	GPIO10[1]	**GPIO10 —** General Purpose digital Input/Output 10
				CT32B0_CAP0 — 32-bit CT32B0 capture input 0
				USART1_TXD — Universal Synchronous/Asynchronous Receiver/Transmitter 1 - transmit data output
				RFTX — Radio Transmit Control Output
				I2C0_SCL — I^2C-bus 0 master/slave SCL input/output (open drain)
				SPI0_SCK — Serial Peripheral Interface-bus 0 clock input/output
				PDM0_DATA — Pulse Density Modulation Data input from digital microphone (channel 0)
PIO11	14	IO	GPIO11[1]	**GPIO11 —** General Purpose digital Input/Output 11
				CT32B1_CAP0 — 32-bit CT32B1 capture input 0
				USART1_RXD — Universal Synchronous/Asynchronous Receiver/Transmitter 1 - receive data input
				RFRX — Radio Receiver Control Output
				I2C0_SDA — I^2C-bus 0 master/slave SDA input/output (open drain)
				SPI0_MISO — Serial Peripheral Interface-bus 0 master input slave output
				PDM0_CLK — Pulse Density Modulation Clock output to digital microphone (channel 0)

Table 3. Pin descriptions

Symbol	Pin	Type	Default at reset	Description
PIO12/SWCLK	15	IO	SWCLK	**GPIO12 —** General Purpose digital Input/Output 12
				SWCLK — Serial Wire Debug Clock
				PWM0 — Pulse Width Modulator output 0
				I2C1_SCL — I^2C-bus 1 master/slave SCL input/output (open drain)
				SPI0_MOSI — Serial Peripheral Interface-bus 0 master output slave input
				ANA_COMP_OUT — Analog Comparator digital output
				IR_BLASTER — Infra-Red Modulator output
PIO13/SWDIO	16	IO	SWDIO	**GPIO13 —** General Purpose digital Input/Output 13
				SPI1_SSELN2 — Serial Peripheral Interface-bus 1, slave select not 2
				SWDIO — Serial Wire Debug Input/Output
				PWM2 — Pulse Width Modulator output 2
				I2C1_SDA — I^2C-bus 1 master/slave SDA input/output (open drain)
				SPI0_SSELN — Serial Peripheral Interface-bus 0, slave select not
PIO14/ADC0	17	IO	GPIO14[1]	**ADC0 —** ADC input 0
				GPIO14 — General Purpose digital Input/Output 14
				SPI1_SSELN1 — Serial Peripheral Interface-bus 1, slave select not 1
				CT32B0_CAP1 — 32-bit CT32B0 capture input 1
				PWM1 — Pulse Width Modulator output 1
				SWO — Serial Wire Output
				USART0_SCK — Universal Synchronous/Asynchronous Receiver/Transmitter 0 - synchronous clock
				MCLK — External clock, can be provided to DMIC IP
				RFTX — Radio Transmit Control Output
PIO15/ADC1	18	IO	GPIO15[1]	**ADC1 —** ADC input 1
				GPIO15 — General Purpose digital Input/Output 15
				SPI1_SCK — Serial Peripheral Interface-bus 1, clock input/output
				ANA_COMP_OUT — Analog Comparator digital output
				PWM3 — Pulse Width Modulator output 3
				PDM1_DATA — Pulse Density Modulation Data input from digital microphone (channel 1)
				I2C0_SCL — I2C-bus 0 master/slave SCL input/output (open drain)
				RFRX — Radio Receiver Control Output
PIO16/ADC2	19	IO	GPIO16[1]	**ADC2 —** ADC input 2
				GPIO16 — General Purpose digital Input/Output 16
				SPI1_SSELN0 — Serial Peripheral Interface-bus 1, slave select not 0
				PWM5 — Pulse Width Modulator output 5
				PDM1_CLK — Pulse Density Modulation Clock output to digital microphone (channel 1)
				SPIFI_CSN — Quad-SPI Chip Select Not, output
				ISO7816_RST — RST signal, output, for ISO7816 interface
				I2C0_SDA — I2C-bus 0 master/slave SDA input/output (open drain)

Table 3. Pin descriptions

Symbol	Pin	Type	Default at reset	Description
V_{DDE}	20	P		**V_{DDE} —** Supply voltage for IO
PIO17/ADC3	21	IO	GPIO17[1]	**ADC3 —** ADC input 3
				GPIO17 — General Purpose digital Input/Output 17
				SPI1_MOSI — Serial Peripheral Interface-bus 1, master output slave input
				SWO — Serial Wire Output
				PWM6 — Pulse Width Modulator output 6
				SPIFI_IO3 — Quad-SPI Input/Output 3
				ISO7816_CLK — Clock output for ISO7816 interface
				CLK_OUT — Clock out
PIO18/ADC4	22	IO	GPIO18[1]	**ADC4 —** ADC input 4
				GPIO18 — General Purpose digital Input/Output 18
				SPI1_MISO — Serial Peripheral Interface-bus 1, master data input
				CT32B0_MAT1 — 32-bit CT32B0 match output 1
				PWM7 — Pulse Width Modulator output 7
				SPIFI_CLK — Quad-SPI Clock output
				ISO7816_IO — IO of ISO7816 interface
				USART0_TXD — Universal Synchronous/Asynchronous Receiver/Transmitter 0 - transmit data output
PIO19/ADC5	23	IO	GPIO19[1]	**ADC5 —** ADC input 5
				GPIO19 — General Purpose digital Input/Output 19
				ADO — Antenna Diversity Odd Output
				PWM4 — Pulse Width Modulator output 4
				SPIFI_IO0 — Quad-SPI Input/Output 0
				USART1_RXD — Universal Synchronous/Asynchronous Receiver/Transmitter 1 - receive data input
				CLK_IN — External clock
				USART0_RXD — Universal Synchronous/Asynchronous Receiver/Transmitter 0 - receive data input
PIO20/ACP	24	IO	GPIO20[1]	**ACP —** Analog Comparator Positive input
				GPIO20 — General Purpose digital Input/Output 20
				IR_BLASTER — Infra-Red Modulator output
				PWM8 — Pulse Width Modulator output 8
				RFTX — Radio Transmit control Output
				SPIFI_IO2 — Quad-SPI Input/Output 2
				USART1_TXD — Universal Synchronous/Asynchronous Receiver/Transmitter 1 - transmit data output

Table 3. Pin descriptions

Symbol	Pin	Type	Default at reset	Description
PIO21/ACM	25	IO	GPIO21[1]	**ACM —** Analog Comparator Negative input
				GPIO21 — General Purpose digital Input/Output 21
				IR_BLASTER — Infra-Red Modulator output
				PWM9 — Pulse Width Modulator output 9
				RFRX — Radio Receiver Control Output
				SWO — Serial Wire Output
				SPIFI_IO1 — Quad-SPI Input/Output 1
				USART1_SCK — Universal Synchronous/Asynchronous Receiver/Transmitter 1 - synchronous clock
TRST	26	G		**TRST —** must be connected to GND
RSTN	27	I		**RSTN —** Reset Not input
V_{BAT}	28	P		**V_{BAT} —** Supply voltage DCDC input
LX	29			**LX —** DCDC filter
$V_{SS(DCDC)}$	30	G		**$V_{SS(DCDC)}$ —** ground for DCDC section
FB	31			**FB —** DCDC Feedback input
$V_{DD(PMU)}$	32	P		**$V_{DD(PMU)}$ —** supply voltage for PMU section
XTAL_32K_P	33			crystal oscillator 32.768 kHz
XTAL_32K_N	34			crystal oscillator 32.768 kHz
$V_{DD(RADIO)}$	35	P		**$V_{DD(RADIO)}$ —** supply voltage for radio section
$V_{SS(RF)}$	36	G		**$V_{SS(RF)}$ —** RF ground
RF_IO	37	IO		**RF_IO —** RF antenna, RF pin which can be considered as RF Input/output. The radio transceiver is connected here.
$V_{SS(RF)}$	38	G		**$V_{SS(RF)}$ —** RF ground
LB	39			NFC tag antenna input B
LA	40			NFC tag antenna input A
exposed die pad		G		must be connected to RF ground plane

[1] I: input at reset.

[2] For standard operation (normal boot or ISP programming mode), this pin should be high during the release of reset. If there is no external driver to this pin, then the internal pull-up will keep this pin high.

[3] ISP programming mode: leave pin floating high during reset to avoid entering UART programming mode or hold it low to program.

[4] In ISP mode, it is configured to USART0_TXD.

[5] In ISP mode, it is configured to USART0_RXD.

3.4 Pin properties

Table 4 presents the different functionality and default states of the pins. PIO10 and PIO11 have IO cells that support true I2C operation and also general purpose digital modes. The reset and test reset pins support a narrow range of functionality. All other digital IOs are standard GPIO IO cells.

Table 4. Pin properties

Pin No.	Pin Name	Default status after POR	Pullup/ Pulldown enable after POR	Pullup/ pulldown selection after POR	Slew rate after POR	Passive pin filter after POR	Open drain enable at reset	Open drain enable control	Pin interrupt	Fast capability
1	XTAL_P	—	—	—	—	—	—	—	—	—
2	XTAL_N	—	—	—	—	—	—	—	—	—
3	PIO0	Hi-Z	Y	PU	SS	N	N	N	Y	N
4	PIO1	Hi-Z	Y	PD	SS	N	N	N	Y	N
5	PIO2	Hi-Z	Y	PD	SS	N	N	N	Y	N
6	PIO3	Hi-Z	Y	PU	SS	N	N	N	Y	N
7	PIO4	Hi-Z	Y	PU	SS	N	N	N	Y	N
8	PIO5/ISP_ENTRY	Hi-Z	Y	PU	SS	N	N	N	Y	N
9	PIO6	Hi-Z	Y	PD	SS	N	N	N	Y	N
10	PIO7	Hi-Z	Y	PD	SS	N	N	N	Y	N
11	PIO8/TXD0	Hi-Z	Y	PU	SS	N	N	N	Y	N
12	PIO9/RXD0	Hi-Z	Y	PU	SS	N	N	N	Y	N
13	PIO10	Hi-Z	N[1]	—	SS	N	N	Y	Y	Y
14	PIO11	Hi-Z	N[1]	—	SS	N	N	Y	Y	Y
15	PIO12/SWCLK	Hi-Z	Y	PU	SS	N	N	N	Y	N
16	PIO13/SWDIO	Hi-Z	Y	PU	SS	N	N	N	Y	N
17	PIO14/ADC0	Hi-Z	Y	PU	SS	N	N	N	Y	N
18	PIO15/ADC1	Hi-Z	Y	PU	SS	N	N	N	Y	N
19	PIO16/ADC2	Hi-Z	Y	PU	SS	N	N	N	Y	N
20	V_{DDE}	—	—	—	—	—	—	—	—	—
21	PIO17/ADC3	Hi-Z	Y	PD	SS	N	N	N	Y	N
22	PIO18/ADC4	Hi-Z	Y	PD	SS	N	N	N	Y	N
23	PIO19/ADC5	Hi-Z	Y	PD	SS	N	N	N	Y	N
24	PIO20/ACP	Hi-Z	Y	PD	SS	N	N	N	Y	N
25	PIO21/ACM	Hi-Z	Y	PU	SS	N	N	N	Y	N
26	TRST[2]	Hi-Z	N	—	—	—	—	—	N	—
27	RSTN	H	Y	PU	—	—	—	—	N	—
28	V_{BAT}	—	—	—	—	—	—	—	—	—
29	LX	—	—	—	—	—	—	—	—	—
30	$V_{SS(DCDC)}$	—	—	—	—	—	—	—	—	—
31	FB	—	—	—	—	—	—	—	—	—

Table 4. Pin properties

Pin No.	Pin Name	Default status after POR	Pullup/ Pulldown enable after POR	Pullup/ pulldown selection after POR	Slew rate after POR	Passive pin filter after POR	Open drain enable at reset	Open drain enable control	Pin interrupt	Fast capability
32	$V_{DD(PMU)}$	—	—	—	—	—	—	—	—	—
33	XTAL_32K_P	—	—	—	—	—	—	—	—	—
34	XTAL_32K_N	—	—	—	—	—	—	—	—	—
35	$V_{DD(RADIO)}$	—	—	—	—	—	—	—	—	—
36	$V_{SS(RF)}$	—	—	—	—	—	—	—	—	—
37	RF_IO	—	—	—	—	—	—	—	—	—
38	$V_{SS(RF)}$	—	—	—	—	—	—	—	—	—
39	LB	—	—	—	—	—	—	—	—	—
40	LA	—	—	—	—	—	—	—	—	—

[1] External pullup required

[2] Tie to ground for functional mode

Table 5: Abbreviation used in the Table 4

Properties	Abbreviation	Descriptions
Default status after POR	Hi-Z	High impedance
	H	High level
	L	Low level
Pullup/pulldown Enable after POR	Y	Enabled
	N	Disabled
Pullup/pulldown selection after POR	PU	Pullup (ie reg ICON.MODE = 0x0)
	PD	Pulldown (ie reg ICON.MODE = 0x3)
Slew rate after POR	FS	Fast slew rate • For MFIO pads ie all except PIO10&11: IOCON.SLEW = 0 or 1, IOCON.SLEW1 = 1 • For IICFPGPIO pads ie PIO10&11: IOCON.SLEW = 1
	SS	Slow slew rate • For MFIO pads ie all except PIO10&11: IOCON.SLEW = 0, IOCON.SLEW1 = 0 • For IICFPGPIO pads ie PIO10&11: IOCON.SLEW = 0
Passive Pin Filter after POR	N	Disabled (ie reg IOCON.FILTEROFF = 1)
	Y	Enabled (ie reg IOCON.FILTEROFF = 0)

Table 5: Abbreviation used in the Table 4

Properties	Abbreviation	Descriptions
Open drain enable after reset	N	Disabled (ie reg IOCON.OD = 0)
	Y	Enabled (ie reg IOCON.OD = 1)
Open drain enable control	N	Disabled[1]
	Y	Enabled
Pin interrupt	N	No
	Y	Yes
Fast capability	N	Not support fast capability
	Y	Support fast capability

[1] All PIO except PIO10 and PIO11 can be configured to operate in a pseudo-open drain mode

3.5 General information for handling PIOs

3.5.1 IO clamping

IO clamping can be used in power down mode to maintain digital outputs when necessary (except if driven by SPI0, USART0, I^2C0). It is necessary to activate this function just before going into power down.

It should only be enabled for power down mode and for IO cells being used as outputs where the application requires the pin state to be held during the power down cycle. IO cells that are used as inputs must not be clamped because the level on the pin depends on the external driver; a clamp could create a conflict.

Settings to freeze the IO:

- Assert SYSCON_RETENTIONCTRL register, to enable the option of clamping
- Assert SYSCON_RETENTIONCTRL[IOCLAMP] field for all IOs to be maintained or clamped:
  - IOCON_PIOx[11] for MFIO pads (ie. All PIOs except PIO10 and 11)
  - IOCON_PIOx[12] for combo I^2C/GPIO pads (ie. PIO10 and 11).

After wake-up from power-down, the clamp will still be enabled for all the IPs that are clamped. The application is responsible for releasing the clamps to allow for normal IO operation. Hence, the clamp must be released for each clamped IO cell by clearing the SYSCON_RETENTIONCTRL[IOCLAMP] control bits.

3.5.2 IOs and power modes

PIOs of PIO_0 to PIO_21 can be used as GPIO. You can use them as general-purpose inputs and outputs or for specific functions, like I^2C, USART, ISO7816, etc.

By default, at reset and during boot, GPIO mode is often the selected mode, except for IOs 8/9/12/13 (see Table 19 “IOMUX functions”).

If boot fails, ISP mode is activated, see more in Chapter 38 “In-System Programming (ISP)”.

Then in ACTIVE, SLEEP, DEEPSLEEP modes, user has complete control over PADs: speed, inversion, filter, open drain, pull-up, pull-down, bus keeper, disable input. These settings are configured independently of whether the pin is used as GPIO or for a specific function.

Note here that PIOs 10/11 are using special PADs that make them behave differently from the other ones. For example, pull-down is not supported (see Table 19 “IOMUX functions”).

The maximum supported frequency is not only affected by the type of PAD being used but also is limited by construction to either 10 MHz (PIOs from 0 to 15) or 33 MHz (PIOs from 16 to 21). All these restrictions are checked by the reference SW API joined below.

Selection of input or output mode (direction), and value to output are controlled dynamically by the functional module itself: GPIO, I^2C, USART, ISO7816, etc.

In POWERDOWN modes, most functional modules are powered off, which means they lose control over direction and value to output. If it is required to maintain the output value of an IO during power-down, then two steps are required:

1. Firstly, it is necessary to enable the IOCLAMP feature, giving the ability for any IO to freeze and maintain an output value during the power-down phase. To do this refer to the SYSCON_RETENTIONCTRL[IOCLAMP] control bit.
2. Secondly, for any IO that is required to clamp, there is a dedicated control bit in the respective IOCON_PIOx register (See Section 3.5.1 “IO clamping”). IOs that are operating as inputs do not require this because they are driven by an external source.

Note that in some POWERDOWN modes, power domain Comm0 is maintained, which means USART0, I^2C0, SPI0 can still have control over PADs so no need in that case to use IOCLAMP feature.

In DEEPPOWERDOWN mode, it is not possible to have any IOs as an output or clamping. There are two options on managing the IO during DEEPPOWERDOWN mode:

- IOs are powered off. The IC is waiting for a PAD reset or an event on NTAG (when available)
- IOs are kept alive to wake the IC with for example an event on PIO. In that case, internal isolation forces PIOs as input with pre-defined configuration (pull-up/pull-down), the same as the one defined after reset (see Table 19 “IOMUX functions”)

See Chapter 5 “Power Management” for more details.

3.5.3 IOs speed and configuration

Summary of possible configurations on PIOs

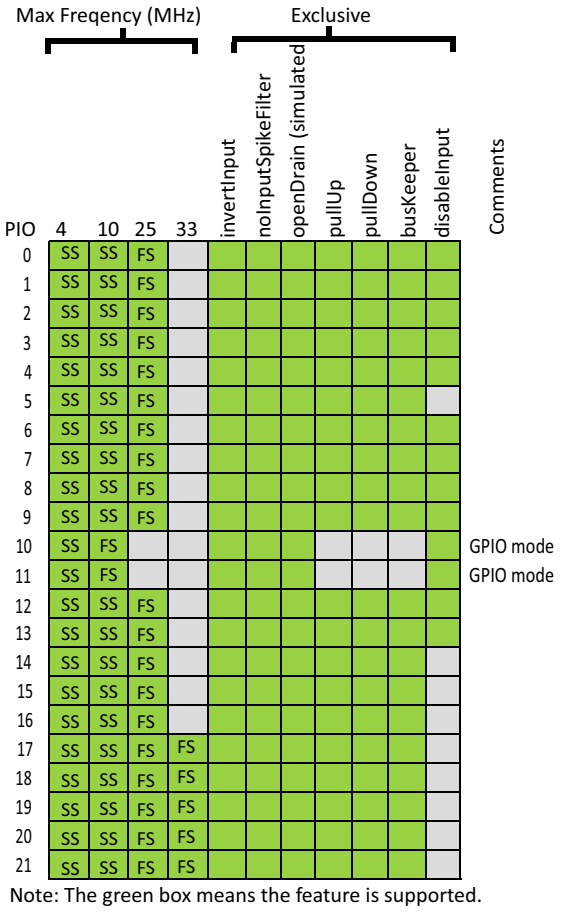


Fig 4. Possible configurations on PIOs

Figure 4 shows the maximum speed of operation of the PIO outputs and also the slew setting required for this operation, SS or FS. For definition of the slew settings, see Table 5.

The figure also shows the modes supported on the PIOs.

For PIO10 and PIO11, it is necessary to configure the IO for GPIO mode in order to achieve the stated output frequencies.

3.5.3.1 IO application mode

The table below lists the 40 IO application modes available on JN5189(T)/JN5188(T):

Table 6. IO application mode

IO Application mode	Description of IO functional mode
IO in strong push pull:	
0	IOx: Strong output 0, receiver enabled, no pull-up, no pull-down, low speed
1	IOx: Strong output 1, receiver enabled, no pull-up, no pull-down, low speed
2	IOx: Strong output 0, receiver enabled, no pull-up, no pull-down, high speed
3	IOx: Strong output 1, receiver enabled, no pull-up, no pull-down, high speed

Table 6. IO application mode

IO Application mode	Description of IO functional mode
4	IOx: Strong output 0, receiver disabled, no pull-up, no pull-down, low speed
5	IOx: Strong output 1, receiver disabled, no pull-up, no pull-down, low speed
6	IOx: Strong output 0, receiver disabled, no pull-up, no pull-down, high speed
7	IOx: Strong output 1, receiver disabled, no pull-up, no pull-down, high speed
IO in open drain mode (external pull-up):	
8	IOx: Strong output 0, receiver enabled, no pull-up, no pull-down, low speed
9	IOx: Output disabled , receiver enabled, no pull-up, no pull-down, low speed (pull-up at application level)
10	IOx: Strong output 0, receiver enabled, no pull-up, no pull-down, high speed
11	IOx: Output disabled, receiver enabled, no pull-up, no pull-down, high speed (pull-up at application level)
12	IOx: Strong output 0, receiver disabled, no pull-up, no pull-down, low speed
13	IOx: Output disabled, receiver enabled, no pull-up, no pull-down, low speed (pull-up at application level)
14	IOx: Strong output 0, receiver disabled, no pull-up, no pull-down, high speed
15	IOx: Output disabled, receiver enabled, no pull-up, no pull-down, high speed (pull-up at application level)
IO in open drain mode (Internal pull-up)[1]:	
16	IOx: Strong output 0, receiver enabled, pull-up enabled, no pull-down, low speed
17	IOx: Output disabled, receiver enabled, pull-up enabled, no pull-down, low speed
18	IOx: Strong output 0, receiver enabled, pull-up enabled, no pull-down, high speed
19	IOx: Output disabled, receiver enabled, pull-up enabled, no pull-down, high speed
IO in input mode only:	
20	IOx: Output disabled, receiver enabled, no pull-up, no pull-down, low speed (input filtered)
21	IOx: Output disabled, receiver enabled, no pull-up, no pull-down, high speed (input not filtered)
IO in Input mode with pull-up or pull-down[1]:	
22	IOx: Output disabled, receiver enabled, pull-up enabled, no pull-down, low speed (input filtered)
23	IOx: Output disabled, receiver enabled, pull-up enabled, no pull-down, high speed (input not filtered)
24	IOx: Output disabled, receiver enabled, no pull-up, pull-down enabled, low speed (input filtered)
25	IOx: Output disabled, receiver enabled, no pull-up, pull-down enabled, high speed (input not filtered)
IO in pull-up or pull-down mode only[1]:	
26	IOx: Output disabled, receiver disabled, pull-up enabled, no pull-down, low speed
27	IOx: Output disabled, receiver disabled, pull-up enabled, no pull-down, high speed
28	IOx: Output disabled, receiver disabled, no pull-up, pull down enabled, low speed
29	IOx: Output disabled, receiver disabled, no pull-up, pull-down enabled, high speed
IO high impedance /floating:	
30	IOx: Output disabled, receiver disabled, no pull-up, no pull-down, low speed
31	IOx: Output disabled, receiver disabled, no pull-up, no pull-down, high speed

Table 6. IO application mode

IO Application mode	Description of IO functional mode
IO with repeater mode[1]:	
32	IOx: Strong output 0, receiver enabled, pull-up enabled, pull-down enabled, low speed
33	IOx: Strong output 0, receiver enabled, pull-up enabled, pull-down enabled, high speed
34	IOx: Strong output 1, receiver enabled, pull-up enabled, pull-down enabled, low speed
35	IOx: Strong output 1, receiver enabled, pull-up enabled, pull-down enabled, high speed
36	IOx: Output disabled, receiver enabled, pull-up enabled, pull-down enabled, low speed
37	IOx: Output disabled, receiver enabled, pull-up enabled, pull-down enabled, high speed
IO in Analogue mode:	
38	IOx: Output disabled, receiver disabled, no pull-up, no pull-down, low speed
IO in Switch OFF mode (DPD0 => CPD asserted):	
39	IOx: High impedance /floating

[1] Not valid for PIO10 and PIO11.

To configure these modes, the PIO or PIO_I2C registers in IOCON must be set correctly. The exception to this is option 39; this option occurs when the device is put into deep power-down mode, without the option of IO wake-up.

4.1 General information

The Analog Power Management Unit (PMU) contains the analog blocks for generating clocks, creating and managing the internal power domains. For reliable start-up and device operation, the power-on reset (POR) and brownout detect (BOD) blocks are necessary and also form part of the PMU. These blocks are outlined in this section.

Overall, these blocks are needed to support the device functionality and to achieve low-power consumption in functional and low-power modes.

The control and configuration of most of these features are managed either by dedicated hardware state machines, to manage the device start-up or power-down modes, or the Power Management API library.

4.2 Power supplies

The JN5189(T)/JN5188(T) has three power supplies: the main power connection, V_{BAT}, and two lower voltage supplies, V_{DD}(RADIO) and V_{DD}(PMU). Internal to the JN5189(T)/JN5188(T), there is a DCDC converter which can generate the V_{DD} supplies, see Section 4.2.1 “DCDC”.

From these three power supplies, there are further internal power domains which are used to support a range of operating modes. The power domains are presented in Section 4.3 “Power domains”.

4.2.1 DCDC

The JN5189(T)/JN5188(T) has an internal DCDC module. It is a buck converter which efficiently converts an input supply voltage to a fixed output voltage. The configuration of the DCDC module can be optimized to suit the load current of the application; there are settings for 10 mA, 20 mA, 40 mA and 60 mA. The default configuration is 40 mA; software APIs are available to configure the other settings.

The DCDC converter is connected to V_{BAT} and generates the supply voltage required for V_{DD}(RADIO) and V_{DD}(PMU). The external configuration for this is shown in the following diagram.

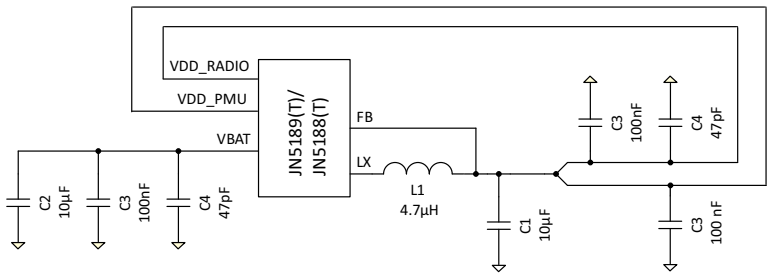


Fig 5. DCDC system diagram

4.3 Power domains

The JN5189(T)/JN5188(T) has many power domains; these are created in various ways such as internal LDOs, power switches or connections to DCDC output. The voltage of the LDOs may be changed to reduce power in certain modes and the domain may be switched off as well. The APIs of the Low-power library, fsl_power library, manage the configurations of the power domains. The main domains are listed here:

- Always On: This domain is used to control initial device start-up and the deep power-down states with their wake-up triggers
- System: System control features are in this domain such as the sleep controller and low-power wake-up, IO configuration and clock control. This domain is active in most operating modes.
- Comm0: In power-down mode, some digital communication modules can still operate in a reduced capacity. The Comm0 domain ensures these are powered when required.
- Mem: The SRAMs can all be powered separately and, if memory retention is required in power-down state, then voltage scaling is used to reduce power consumption
- Core: The majority of the digital logic is in the core domain, used during main device operation
- ADC: The general purpose ADC has its own supply
- Flash: The flash memory has its own supply
- IO domain: To achieve the very low deep power-down current consumption, it is necessary to remove power from IO cells and hence a separate domain is necessary
- Retention in Radio Controller: When digital logic is unpowered, it will lose its state, a small number of registers in the radio controller have been managed specially in order to keep their state in power-down mode, so that on wake-up their values will still be valid.

4.4 Clocks

There are several clock sources in the JN5189(T)/JN5188(T) to support a range of operating conditions.

4.4.1 FRO1M

The FRO1M clock is an internal, very low-power 1 MHz FRO used for the main system controller state machines. It can be trimmed to improve its accuracy. During device production, trimming values are determined and these are applied, by boot software, before the application starts. Once trimmed, the FRO accuracy is within ±15% across operating temperature and voltage.

4.4.2 High speed FRO

The high speed FRO clock module is an internal module that generates several clocks: 48 MHz, 32 MHz, 12 MHz. It can be trimmed to improve its accuracy. During device production, trimming values are determined and these are applied by hardware before the device starts executing the boot code. Once trimmed, the FRO accuracy is within ±2% across operating temperature and voltage.

4.4.3 FRO32K

The FRO32K clock is an internal, ultra low-power 32 kHz FRO used for the RTC and low-power wake-timers. It can be trimmed to improve its accuracy. During device production, trimming values are determined and these are applied, by software APIs, before the clock is used. Once trimmed, the FRO accuracy is within ±2% across operating temperature and voltage.

4.4.4 XTAL32K

The XTAL32K clock is generated from the internal clock module and an external 32 kHz XTAL. The accuracy is determined by the XTAL selected, and very high precision devices are available. This clock can be used, instead of the internal FRO32K, as the source for the 32 kHz clock.

Generally, an XTAL requires a capacitor connected between each pin of the XTAL and ground. The capacitance required is set by the specification of the crystal. The JN5189(T)/JN5188(T) has internal configurable capacitors that removes the need for external capacitors in most applications.

4.4.5 XTAL32M

For radio operation, a very precise 32 MHz clock source is required. To support this, a 32 MHz XTAL is supported in the radio module. The accuracy is determined by the XTAL selected, and very high precision XTALs are available. For IEEE 802.15.4 operation, it is necessary to meet ±40 ppm accuracy across temperature/voltage and lifetime of the product, accounting for XTAL aging.

Generally, an XTAL requires a capacitor connected between each pin of the XTAL and ground. The capacitance required is set by the specification of the crystal. The JN5189(T)/JN5188(T) has internal configurable capacitors that removes the need for external capacitors in most applications.

The internal 32 MHz XTAL module requires biasing which can be sourced from the PMU or within the radio module. For radio operation, it is necessary to use the radio biasing because the PMU biasing is not stable enough to meet the radio specification. The configuration of the biasing and XTAL module is managed by the CLOCK_EnableClock(kCLOCK_Xtal32M) and also the radio biasing function in the Radio Controller API.

4.4.6 XTAL cap bank management

The two XTAL cells both have capacitor banks connected to the two XTAL pins; this is configured to select the capacitance to match the external XTAL and hence to give the required frequency. Each capacitor bank for the 32 MHz XTAL has a maximum capacitance of approximately 25 pF. For the 32 kHz XTAL, the maximum is approximately 24 pF.

During production test, the capacitor banks are tested and calibrated; the setting required for two specific capacitance values is stored in flash.

The software API that configures the XTAL capacitor bank setting uses this calibration data to achieve accurate operation of the XTAL.

Ideally the external XTAL is stable across the whole temperature range of the device. In this case once the capacitor bank setting has been applied, it does not need modifying if the device remains powered. However, if the XTAL is not stable across the whole temperature range of the device, it may be necessary to adjust the capacitor bank setting based on the device temperature.

4.5 Support Functions

Two functional blocks are included in the Analog Power Management Unit for safe operation of the device. These are Power On Reset (POR) and Brown-out Detect (BOD).

4.5.1 POR

When the device is powered up, the POR module keeps the device in reset until the power supply, VBAT, reaches the release/ trip_high threshold; then the start-up sequence begins. If the supply drops too low, it is necessary to put the device safely into reset again. For this purpose, the POR block creates the reset signal, which is the voltage dropping below the reapply/trip_low threshold where this threshold is slightly lower than the trip_high threshold. The trip levels are typically 0.8 V.

4.5.2 BOD

The device must be supplied by VBAT within the specified operating limits. There is a large gap between the lowest allowed voltage and the point at which the POR block would cause the device to enter the reset state. To allow for detection of the supply being within this region, it supports for Brown-out detection. This block supports a configurable threshold and will create an interrupt if a low level is detected. This allows application software to complete any critical operation before shutting down the device.

5.1 Introduction

The JN5189(T)/JN5188(T) supports several low-power modes that can be used by the application to reduce average power consumption. This chapter provides an overview of these modes, what can be enabled in these modes and what events can be used to trigger a return to the normal active state.

The Power Management Controller has state machines and configuration settings to manage the low-power states and these are explained in Chapter 8 “Power Management Controller and SLEEPCON”. To allow easy use of these power modes there is a software API provided, this is introduced in Chapter 11 “Power Control API”.

Many of the features required to achieve these low-power modes use the blocks within the Power Management Unit, this is described in Chapter 4 “Analog Power Management Unit”.

5.2 General description

There is a primary power input to the device V_{BAT}. From this a DCDC converter, programmable LDOs and power switches are used to create numerous power domains. A key power domain for the power modes is the always-on power domain, this domain is always active as long as sufficient voltage is supplied to V_{BAT}. This domain controls the device start-up and the lowest power modes when all other power domains may be off.

In addition to controlling power domains, the other parts are also carefully controlled in power down states to achieve correct functionality and low power.

The following modes are supported in order from highest to lowest power consumption:

1. Active mode:

   The part is in active mode after a Power-On Reset (POR) and when it is fully powered and operational after booting.

2. Sleep mode:

   This is the same as Active mode except that the processor is inactive and is waiting for an interrupt/event to cause it to restart operation. There are no changes to power domains compared to active state.

3. Deep-sleep mode:

   Deep-sleep mode allows some functional blocks and clocks to be disabled to save power. In addition, the core voltage is reduced to save power; this restricts the maximum operating frequency to be 12 MHz. DMA operation is not permitted while the device is in Deep Sleep mode.

   Wake-up from deep-sleep mode takes more time than from sleep mode due to the need to alter system voltages and also to re-enable blocks which had been disabled in the deep-sleep mode.

   The CPU clock is switched off in deep-sleep mode.

In deep-sleep mode, SRAM access is not possible and the SRAM will be in one of three states: powered off; in low-power state to retain contents; in normal state. The SRAM containing data necessary for the application must not be powered off.

Other blocks that may be switched off are: Flash, ADC, analog comparator, temperature sensor, brown out detectors, XTAL32M, XTAL32K, FRO192M, FRO32K.

Peripherals can be left running provided that they have the required clock and are not using DMAs (DMA is not available in deep-sleep mode).

On wake-up the processor will continue code execution from where it was before entering deep-sleep mode.

4. Power-down mode:

   Power-down mode switches off further functionality to save even more power. The main digital domain is switched off, flash is off and the SRAM is either off or in a low-power state to retain contents. The only peripherals that can operate are the I^2C0, USART0 and SPI0, but in a limited functionality mode. A 32 kHz clock can be still active, either FRO32K or XTAL32K.

   After a wake-up, the processor will start executing the boot code to determine how to reinitialize the device.

5. Deep power-down mode:

   Deep power-down mode shuts down virtually all on-chip power consumption, and requires a significantly longer wake-up time. For maximal power savings, the entire system (CPU and all peripherals) is shut down except for the PMU. On wake-up, the device reboots.

   The device can be woken by the reset pin or an IO event unless the IOs are disabled to reduce current consumption further. For a JN5189T or JN5188Tdevice, NTAG FD (Field Detect) interrupt, from the internal NTAG device can also wake up the device.

Table 7. Peripheral configuration in reduced power modes

Peripheral	Power mode			
	Active or sleep	Deep-sleep	Power-down	Deep power-down
PD_MCU	On	On	Retained/ Off	Off
PD_SYSTEM	On	On	On	Off
PD_COMM0	On	On	Optional	Off
PD_AON	On	On	On	On
PD_IO	On	On	On	Optional
PD_MEM	On	On / Retained/ Off	Retained/ Off	Off
Flash	On	Optional	Off	Off
GP ADC	Optional	Optional	Off	Off
Comparator	Optional	Optional	Optional	Off
DCDC converter	On	On	Off	Off
Temperature Sensor	Optional	Optional	Off	Off
Power on reset	On	On	On	On
BODVBAT	Optional	Optional	Optional	Optional
XTAL32M	Optional	Optional	Off	Off
XTAL32K	Optional	Optional	Optional	Off
FRO1M	On	On	Off	Off

Table 7. Peripheral configuration in reduced power modes

Peripheral	Power mode			
	Active or sleep	Deep-sleep	Power-down	Deep power-down
High speed FRO	On	Optional	Off	Off
FRO32K	Optional	Optional	Optional	Off
Radio	Optional	Off	Off	Off
CPU	On or Halted in Sleep	Halted	Off	Off
I2C0	Optional	Optional	Optional (with limited functionality)	Off
SPI0	Optional	Optional	Optional (with limited functionality)	Off
USART0	Optional	Optional	Optional (with limited functionality)	Off
Other digital peripherals	Optional	Optional	Off	Off
DMA	Optional	Off	Off	Off

5.2.1 Wake-up process

The part always wakes up to the active mode. To wake up from the reduced power modes, the user must configure the wake-up source. Each reduced power mode supports its own wake-up sources and needs to be configured accordingly as shown in Table 8.

Table 8. Wake-up sources for reduced power modes

Power mode	Wake-up source	Conditions
Sleep	Any interrupt	Enable interrupt in NVIC.

Table 8. Wake-up sources for reduced power modes

Power mode	Wake-up source	Conditions
Deep-sleep	Pin interrupts	Enable pin interrupts in NVIC (see more details in Chapter 9 “Nested Vectored Interrupt Controller (NVIC)”) and SYSCON_STARTER1 registers.
	BOD interrupt	• Enable interrupt in NVIC, see more details in Chapter 9 “Nested Vectored Interrupt Controller (NVIC)” • Enable interrupt using SYSCON_STARTER0[WDT_BOD]. • Configure the BOD to keep running in this mode with the power API.
	Watchdog interrupt	• Enable the watchdog oscillator using the software APIs, see Section 6.4 “Clock control software functions”. • Enable the watchdog interrupt in NVIC (see more details in Chapter 9 “Nested Vectored Interrupt Controller (NVIC)”) and using SYSCON_STARTER0[WDT_BOD]. • Enable the watchdog in the WWDT_MOD and WWDT_FEED registers. • Enable interrupt in WWDT_MOD register. • Configure the selected watchdog oscillator source to keep running in this mode with the power API.
	Watchdog reset	• Enable the watchdog oscillator using the software APIs, see Section 6.4 “Clock control software functions”. • Enable the watchdog and watchdog reset in the WWDT_MOD and WWDT_FEED registers. • Configure the selected watchdog oscillator source to keep running in this mode with the power API
	Reset pin	Always available.
	RTC 1 Hz alarm timer	• Enable the RTC 1 Hz oscillator using the software APIs, see Section 6.4 “Clock control software functions”. • Enable the RTC bus clock in the SYSCON_AHBCLKCTRL0 register. • Start RTC alarm timer by writing a time-out value to the RTC_COUNT register. • Enable the RTCALARM interrupt by using SYSCON_STARTER0[RTC].
	RTC 1 kHz timer time-out and alarm	• Enable the RTC 1Hz and 1kHz clocks using the software APIs, see Section 6.4 “Clock control software functions”. • Start RTC 1 kHz timer by writing a value to the WAKE register of the RTC. • Enable the RTC wake-up interrupt by using the SYSCON_STARTER0[RTC].
	I2C interrupt	Interrupt from I2C in slave mode. See Chapter 25 “Inter-Integrated Circuit (I^2C)”.
	SPI interrupt	Interrupt from SPI in slave mode. See Chapter 24 “Serial Peripheral Interfaces (SPI)”.
	USART interrupt	Interrupt from USART in slave or 32 kHz mode. See Chapter 23 “Universal Synchronous/Asynchronous Receiver/Transmitter (USART)”.
	DMIC	• Enable the DMIC module • Enable DMIC wake-up in the SYSCON_STARTER1 register
	NTAG FD interrupt	Enable interrupt in NVIC and SYSCON_STARTER1 registers.
	Comparator	• Enable ANA_COMP to cause wake-up in SYSCON_STARTER1[ANA_COMP] • Configure comparator to operate in power down, using the two comparator input pins
	ISO7816	• Enable the ISO7816 interface • Enable ISO7816 wake-up in the SYSCON_STARTER1[ISO7816]
	ADC	• Enable the ADC module • Enable ADC wake-up in the SYSCON_STARTER1[ADC_THCMP_OVR]

Table 8. Wake-up sources for reduced power modes

Power mode	Wake-up source	Conditions
Deep-sleep	IR modulator	• Enable the IR modulator • Enable IR modulator wake-up in the SYSCON_STARTER0[IRBLASTER]
	SPIFI	• Enable the SPIFI module • Enable SPIFI wake-up in the SYSCON_STARTER0[SPIFI]
	Wakeup timer	• Enable the wake-up timer • Enable wake-up timer wake-up in the SYSCON_STARTER1[WAKE_UP_TIMER1] and SYSCON_STARTER1[WAKE_UP_TIMER0]
	PWM	• Enable the PWM module • Enable PWM wake-up in the SYSCON_STARTER0[PWMx]
	NFCTAG	• Enable the NFCTAG module • Enable NFCTAG wake-up in the SYSCON_STARTER0[NFCTAG]

Table 8. Wake-up sources for reduced power modes

Power mode	Wake-up source	Conditions
Power-down	IO	• Enable required IO to be able to cause wakeup in PMC_DPDWKSRC[PIOx] • Configure IO to be an input in GPIO_DIR[DIRP_PIOn]
	NTAG FD	• Enable NTAG FD to be able to cause wakeup in PMC_DPDWKSRC[NTAG_FD]
	RTC 1 Hz alarm timer	• Enable the RTC 1 Hz oscillator in the RTC_CTRL[ALARM1HZ]. • Enable the RTC bus clock in the SYSCON_AHBCLKCTRL0[RTC]. • Start RTC alarm timer by writing a time-out value to the RTC_COUNT register. • Enable the RTCALARM interrupt in the SYSCON_STARTER0[RTC].
	RTC 1 kHz timer time-out and alarm	• Enable the RTC 1 Hz oscillator and the RTC 1 kHz oscillator in the RTC_CTRL[RTC1KHZ_EN]. • Enable the RTC bus clock in the SYSCON_AHBCLKCTRL0 register. • Start RTC 1 kHz timer by writing a value to the RTC_WAKE. • Enable the 1 kHz timer by setting the RTC_CTRL[RTC1KHZ_EN] • Enable the RTC wake-up interrupt in the SYSCON_STARTER0[RTC].
	Low-power wake-up timers	• Enable wake up timer 0 and/or 1 to cause wake-up in SYSCON_STARTER1[WAKE_UP_TIMER1] and/or SYSCON_STARTER1[WAKE_UP_TIMER0] • Enable the 32 kHz clock, FRO or XTAL using the software APIs, see Section 6.4 “Clock control software functions” • Configure and enable the wake-up timer using SYSCON_WKT_CTRL and the relevant WKT_LOAD_ registers. Software APIs are provided to perform this configuration.
	Comparator	• Enable ANA_COMP to cause wake-up in SYSCON_STARTER1[ANA_COMP] • Configure comparator to operate in power down, using the two comparator input pins
	I2C0	• Enable CPD_COMM0 to stay active in power-down using the power control APIs • Interrupt from I2C in slave mode. See Chapter 25 “Inter-Integrated Circuit (I^2C)”
	SPIO0	• Enable CPD_COMM0 to stay active in power-down using the power control APIs. • Interrupt from SPIO in slave mode. See Chapter 24 “Serial Peripheral Interfaces (SPI)”
	USART0	• Enable CPD_COMM0 to stay active in power-down using thepower control APIs. • Interrupt from USART in slave mode. See Chapter 23 “Universal Synchronous/Asynchronous Receiver/Transmitter (USART)”
	BODVBAT	• Enable BOD VBAT to wake the device in power-down mode using the power control APIs. • The BOD bias must be enabled during the power-down cycle, this is managed by the power control APIs • Configure trigger threshold for BOD VBAT using PMC_BODVBAT[TRIGLVL]. This is managed by the power control APIs.

Table 8. Wake-up sources for reduced power modes

Power mode	Wake-up source	Conditions
Deep power-down	IO	• Enable IO domain in deep power-down with PMC_CTRL[WAKUPRESETENABLE]
	NTAG FD	• Enable NTAG FD wake-up with PMC_CTRL[NTAGWAKUPRESETENABLE]
	BODVBAT	• Enable BOD VBAT to be enabled in deep power-dwon mode using the power control APIs. • The BOD bias must be enabled during the deep power-down cycle, this is managed by the power control APIs • Configure trigger threshold for BOD VBAT using PMC_BODVBAT[TRIGLVL]. This is managed by the power control APIs.

5.3 Functional description

5.3.1 Power management

The JN5189(T)/JN5188(T) supports a variety of power control features. In Active mode, when the chip is running, power and clocks to selected peripherals can be optimized for power consumption. In addition, there are three special modes of processor power reduction with different peripherals running: sleep mode, deep-sleep mode, and deep power-down mode, activated by the power mode configuration API (see Chapter 11 “Power Control API”).

Remark: The Debug mode is not supported in sleep, deep-sleep, or deep power-down modes.

5.3.2 Active mode

In Active mode, the CPU, memories, and peripherals are clocked by the AHB/CPU clock.

The chip is in Active mode after reset and the default power configuration is determined by the reset values of the registers, such as the SYSCON_AHBCLKCTRL0 and SYSCON_AHBCLKCTRL1 registers. The power configuration can be changed during run time by functional configuration changes such as enabling clock sources, functional blocks and changing the CPU clock speed.

5.3.2.1 Power configuration in Active mode

Power consumption in Active mode is determined by the following configuration choices:

- The AHBCLKCTRL registers control which memories and peripherals are running. In order to save power, the user should turn off the functions that are not needed by the application. If certain functions are not needed in specific time, they can be turned off temporarily and turned back on when they are needed.
- The power to various analog blocks (RAMs, PLL, oscillators, and the BOD circuit) can be controlled individually. As with clock controls, these blocks should generally be turned off if not needed by the application. If turned off, it takes time to make these blocks functional after being turned on. Software APIs are provided to configure and enable or disable these blocks.

- The system clock frequency, 48 MHz to 12 MHz, and source can be selected (See Section 6.3 “Clock generation (CLK_GEN) module”). In general, the device uses less power at lower frequencies, so running the CPU and other device features at a frequency sufficient for the application (plus some margin) will save power. In some cases, a faster CPU frequency is better so that code can be executed quickly to move to a power-down state.
- Several peripherals use individual peripheral clocks with their own clock dividers. The peripheral clocks can be shut down through the corresponding clock divider registers if the base clock is still needed for another function.
- The power library provides an easy way to optimize power consumption depending on CPU load and performance requirements. See Chapter 11 “Power Control API”.

5.3.3 Sleep mode

In sleep mode, the system clock to the CPU is stopped and execution of instructions is suspended until either a reset or an interrupt occurs.

Peripheral functions, if selected to be clocked in the SYSCON_AHBCLKCTRL0 and/or SYSCON_AHBCLKCTRL1 registers, continue operation during sleep mode and may generate interrupts to cause the processor to resume execution. Sleep mode eliminates dynamic power used by the processor itself, memory systems and related controllers, and internal buses. The processor state and registers, peripheral registers, and internal SRAM values are maintained, and the logic levels of the pins remain static.

As in Active mode, the power API provides an easy way to optimize power consumption depending on CPU load and performance requirements in sleep mode. See Chapter 11 “Power Control API”.

5.3.3.1 Power configuration in sleep mode

Power consumption in sleep mode is configured by the same settings as in Active mode:

- Enabled clocks remain running.
- The system clock frequency remains the same as in Active mode, but the processor is not clocked.
- Analog and digital peripherals are powered and selected as in Active mode through the registers such as AHBCLKCTRL0 and AHBCLKCTRL1 if other functional blocks, such as clocks, have been enabled..

5.3.3.2 Programming sleep mode

Generally, sleep mode is used when the application has no further processing to perform until a functional event occurs or a timer event occurs. Therefore it is unlikely the configuration changes needed for sleep mode.The following steps must be performed to enter sleep mode:

1. In the NVIC, enable all interrupts that are needed to wake up the part, see Chapter 9 “Nested Vectored Interrupt Controller (NVIC)” for more details.
2. Ensure that an event will occur in the future to end the sleep mode.
3. Execute the WFI instruction to enter sleep mode

5.3.3.3 Wake-up from sleep mode

Sleep mode is exited automatically when an interrupt enabled by the NVIC arrives at the processor or a reset occurs. After wake-up caused by an interrupt, the device returns to its original power configuration as the processor clock will be restarted and no other changes occurred due to being in sleep mode. If a reset occurs, the microcontroller enters the default configuration in Active mode.

5.3.4 Deep-sleep mode

In deep-sleep mode, the system clock to the processor is disabled as in sleep mode. Analog blocks are powered down by default but can be selected to keep running through the power API if needed as wake-up sources. Table 7 shows the state of different blocks in deep sleep and indicates which ones the user has options on. The main clock and all peripheral clocks are disabled.

Deep-sleep mode eliminates power used by analog peripherals and all dynamic power used by the processor itself, memory systems and related controllers, and internal buses. The processor state and registers, peripheral registers, and internal SRAM values are maintained, and the logic levels of the pins remain static.

GPIO Pin Interrupts, GPIO Group Interrupts, and selected peripherals such as DMIC, SPI, I^2C, USART, WWDT, RTC, and BOD can be left running in deep-sleep mode. The FRO, RTC oscillator, and the watchdog oscillator can be left running.

5.3.4.1 Power configuration in deep-sleep mode

Power consumption in deep-sleep mode is determined primarily by which analog wake-up sources remain enabled. Serial peripherals and pin interrupts configured to wake up the part contribute to the power consumption only to the extent that they are clocked by external sources. All wake-up events (other than reset) must be enabled in the SYSCON_STARTER registers and in the NVIC. In addition, any related analog block (for example, the RTC oscillator or the watchdog oscillator) must be explicitly enabled through a power API function See Table 8 and Chapter 11 “Power Control API”.

5.3.4.2 Programming deep-sleep mode

The following steps must be performed to enter deep-sleep mode:

1. Select wake-up sources and enable all selected wake-up events in the SYSCON_STARTER0 and SYSCON_STARTER1 registers and in the NVIC.
2. Select the FRO 12 MHz as the main clock.
3. Ensure that an event will occur in the future to end the deep sleep mode.
4. Call the power API with the peripheral parameter to enable the digital/analog peripherals as wake-up sources (see Chapter 11 “Power Control API”).

5.3.4.3 Wake-up from deep-sleep mode

The part can wake up from deep-sleep mode in the following ways:

- Using a signal on one of the four pin interrupts selected in INPUTMUX_PINTSEL register. Each pin interrupt must also be enabled in the SYSCON_STARTER0 register and in the NVIC.

- Using an interrupt from a block such as the watchdog interrupt or RTC interrupt, when enabled during the reduced power mode via the power API. Also enable the wake-up sources in the SYSCON_STARTER registers and the NVIC.
- Using a reset from the $\overline{\text{RESET}}$ pin, or the WWDT (if enabled in the power API).
- Using a wake-up signal from any of the serial peripherals that are operating in deep-sleep mode. Also enable the wake-up sources in the SYSCON_STARTER registers and the NVIC.
- GPIO group interrupt signal. The interrupt must also be enabled in the SYSCON_STARTER1 register and in the NVIC.
- RTC alarm signal or wake-up signal. See Chapter 21 “Real-Time Clock (RTC)”. Interrupts must also be enabled in the SYSCON_STARTER1 register and in the NVIC.

5.3.5 Power down mode

In power-down mode, the always on logic, IO cells and, if enabled, low-power wake timers are powered. During power-down mode, the contents of the SRAM can be optionally retained.

5.3.5.1 Power configuration in power down mode

Power-down mode has configuration options to decide:

- which of the possible wake-up sources will be enabled
- if the RAM contents are to be retained
- if the IO cell values are to be held during power down
- if the 32 MHz XTAL will be restarted automatically on wake-up

5.3.5.2 Wake-up sources for power-down mode

Wake-up from power-down can be accomplished via the reset pin (to cause a reset). Additionally, a wake-up can be triggered by the low-power sleep timers, IO trigger, NTAG FD interrupt, RTC, comparator, BOD VBAT. Also USART0, SPI0 and I^2C0 operating in limited modes may generate a wake-up trigger.

5.3.5.3 Programming power-down mode

For wake-up from the low-power wake timers, it is necessary to configure and enable the timers to cause an interrupt at the correct time in the future.

The low-power API function, see Chapter 11 “Power Control API”, will perform all other necessary configuration for power-down mode. If some PIO outputs need to be maintained during power-down, then some retention must be programmed in this mode.

5.3.5.4 Wake-up from power-down mode

The part goes through almost the whole start-up process when the wake-up event occurs.

- The PMU will turn on the internal on-chip voltage regulators and the DCDC converter
- Optionally the 32 MHz XTAL will be enabled
- Except for some registers in the PMC, all registers will be in their reset state.

5.3.6 Deep power-down mode

In deep power-down mode, power and clocks are shut off to the entire chip with the exception of the always on controllers and, if enabled, the IO cells.

During deep power-down mode, the contents of the SRAM and registers are not retained. All functional pins are tri-stated in deep power-down mode as long as chip power supplied externally. The functional pins can also be inputs and configured to cause a wake-up from deep power down.

5.3.6.1 Power configuration in deep power-down mode

Deep power-down mode has no configuration options. All clocks, the core, and all peripherals are powered down. If required, the BODVBAT can be enabled in deep power-down state, as long as power is supplied to the device.

5.3.6.2 Programming deep power-down mode

See Chapter 11 “Power Control API” for information on entering deep power-mode using the Power Control API.

5.3.6.3 Wake-up from deep power-down mode

Wake-up from deep power-down can be accomplished via the reset pin, NTAG FD and pin interrupt. When the wake-up event occurs, the part goes through the entire reset process when the wake-up event occurs:

- The PMU will turn on the on-chip voltage regulator, DCDC converter, necessary clocks and proceed trough the start-up sequence.
- All registers will be in their reset state.
- CPU will start executing boot code when it is released from reset.

6.1 Introduction

This chapter provides an overview of the system level clock architecture, including the clock generation, division multiplexing, gating and distribution for this device.

6.2 Clock architecture

The main block of the clock architecture is the CLK_GEN. It receives all the clock sources from the external pads, the PMU and the RADIO. It also generates the clock inputs for all the digital blocks.

The clock sources are described in Section 6.3 “Clock generation (CLK_GEN) module”.

The SYSCON block provides a location for programming all the system clock controls (enables, mux selectors, gating controls).

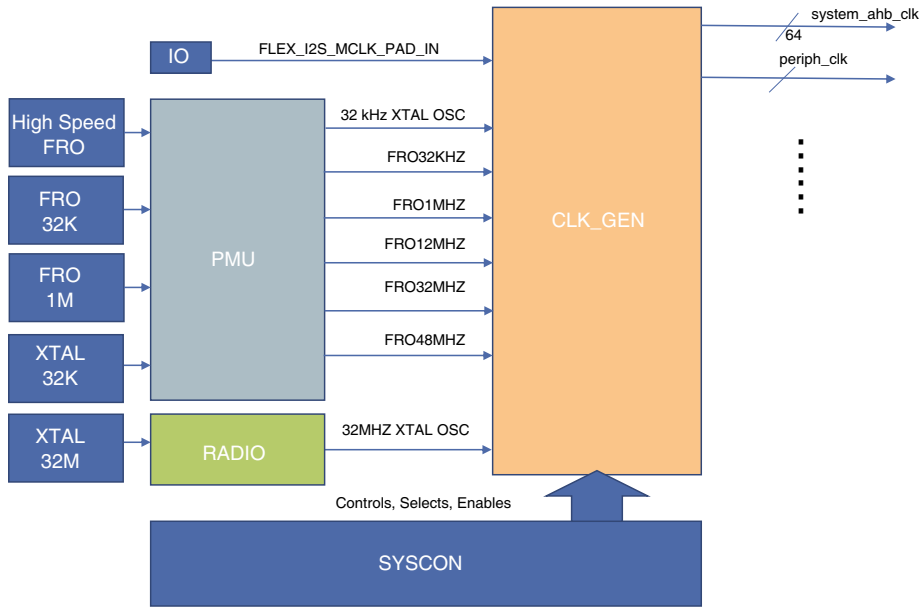


Fig 6. Clock generation high level diagram

6.3 Clock generation (CLK_GEN) module

The CLK_GEN block has multiple input clocks that can be selected as the root clock of the system and peripheral clock outputs and respective clock trees. The following table lists the input clock sources:

Table 9. Clock description

Clock	Frequency	Source	Description
FRO_12MHz	12 MHz	PMU	Derived from the 192 MHz FRO.
FRO_32MHz	32 MHz	PMU	Derived from the 192 MHz FRO.
FRO_48MHz	48 MHz	PMU	Derived from the 192 MHz FRO.
FRO_1MHz	1 MHz	PMU	Used by the PMC (power management block)
FRO_32KHz	32 kHz	PMU	Used in power down modes and for RTC
XTAL_32MHz	32 MHz	Radio	Mainly used by the Radio, and also a source for main_clk
XTAL_32KHz	32 kHz	PMU	To timers and 32 kHz source of several blocks (USART, LSPI, PMC, etc)
MCLK_IN	2 MHz (Max)	PIO	The MCLK input function, when it is connected to a pin by selecting it in the IOCON block. This clock is for DMIC only.

The clock source for the registers and memories is derived from main clock. The main clock can be selected from the sources shown in Figure 7. The main clock, after being optionally divided by the MAINCLK Divider clocks the core, the memories, and the peripherals (register interfaces and peripheral clocks).

All the control registers (indicated in blue) are in the system configuration (SYSCON) block.

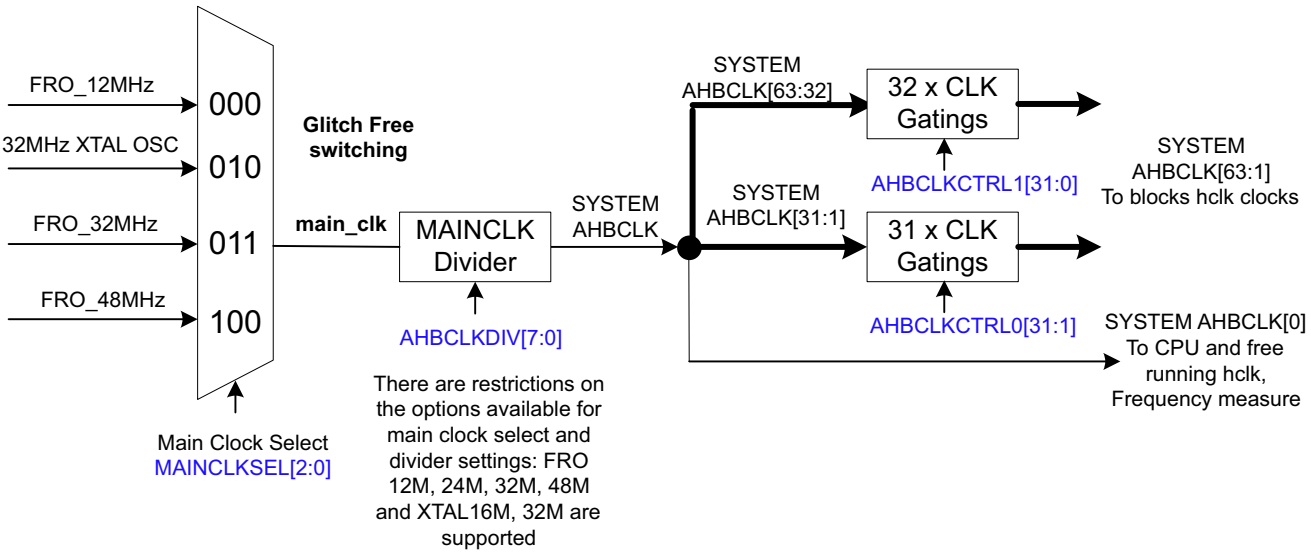


Fig 7. main_clk generation and selection

The Main Clock Select and asynchronous peripheral bridge (APB) clock select muxes are implemented with glitch-free logic. All the other clock muxes described in this chapter cannot be considered as glitch-free, thus it is necessary to pay attention during clock switching. All the dividers can be halted and restarted during clock switching, to provide a glitch free output. During the boot sequence, main_clk and SYSTEM AHBCLK are configured to operate at 12 MHz with the clock sourced from the FRO_12MHz signal.

The 32 MHz and 32 kHz clock sources are also internally muxed, and then used as clock sources for several peripherals. The default source is the FRO option for the two clock muxes. The muxes are shown below:

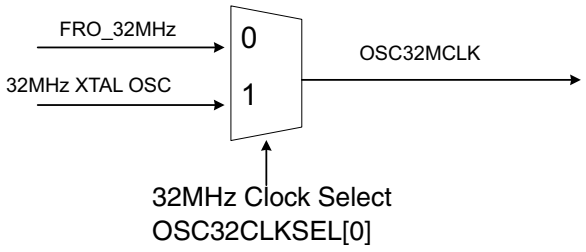


Fig 8. 32 MHz clock sources internal muxing

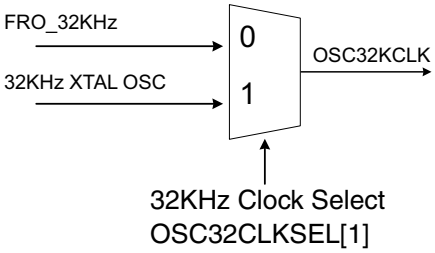


Fig 9. 32 kHz clock sources internal muxing

All the division and/or gating features described in this chapter are provided at CLK_GEN level. Each IP block may provide additional clock control related logic, which is discussed in the related block-specific chapters.

Next pictures show the details of the clock muxing and gating for each clock generated in CLK_GEN block. From reset the clocks are disabled and the clock gate or clock divider blocks.

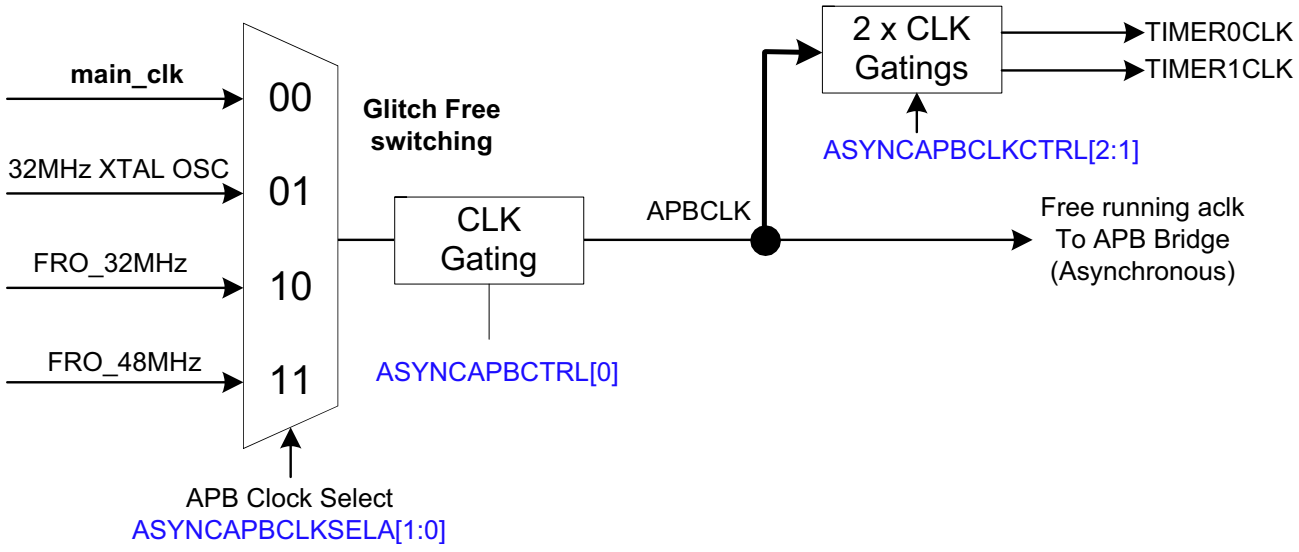


Fig 10. APB clock generation

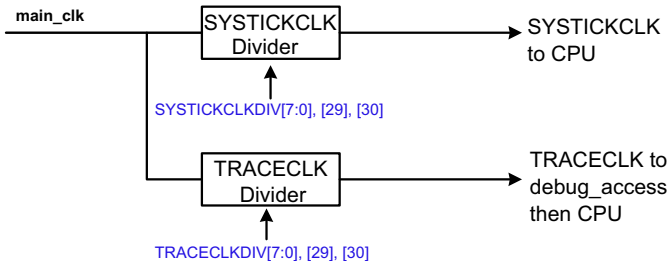


Fig 11. CPU SYSTICK and TRACECLK clock generation

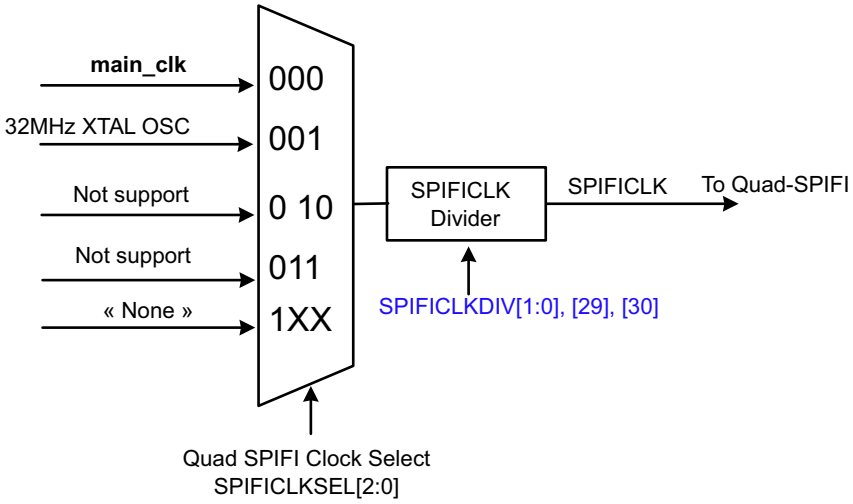


Fig 12. SPIFI clock generation

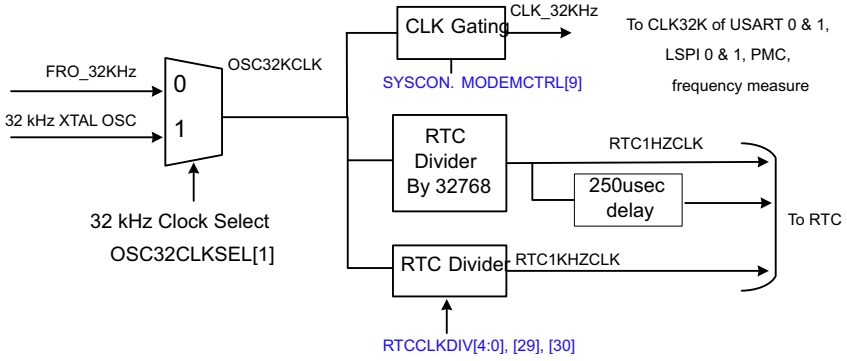


Fig 13. RTC and 32 kHz clocks selection

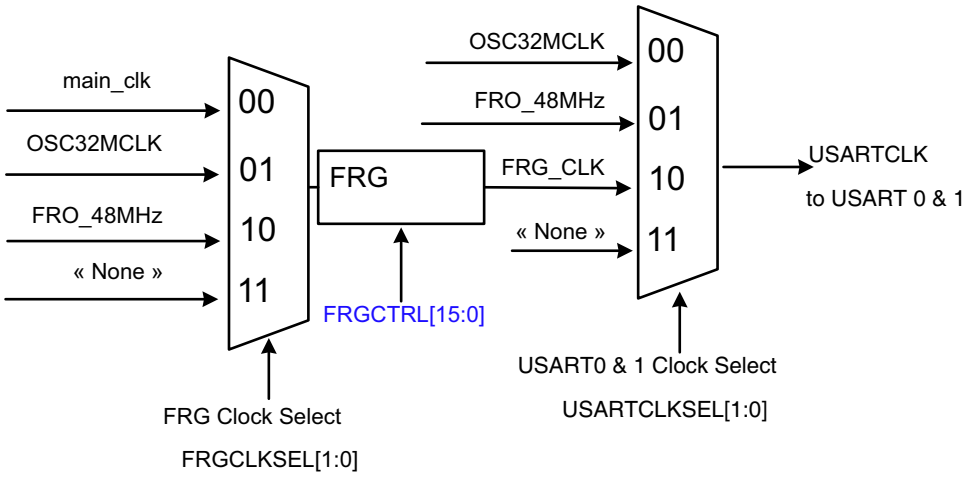


Fig 14. USART clocks selection

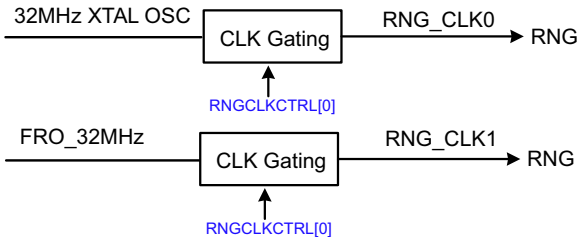


Fig 15. RNG clocks selection and gating

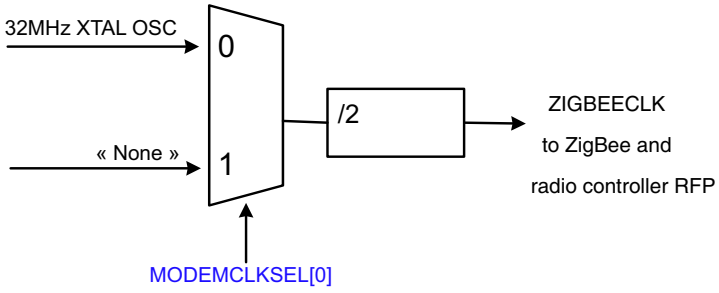


Fig 16. Zigbee/Thread clock selection

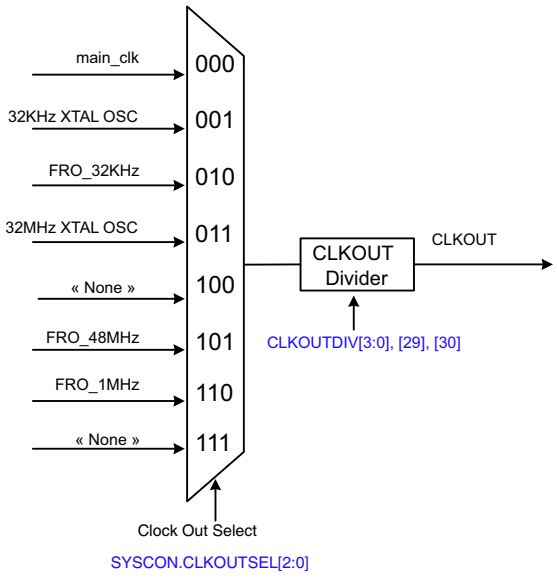


Fig 17. CLKOUT selection

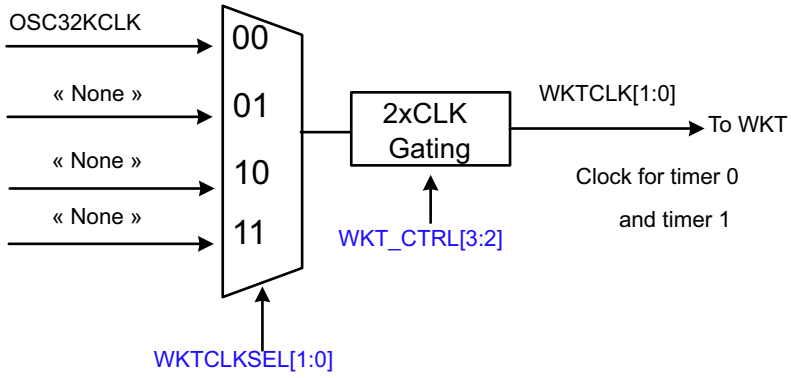


Fig 18. WKT clock selection and gating

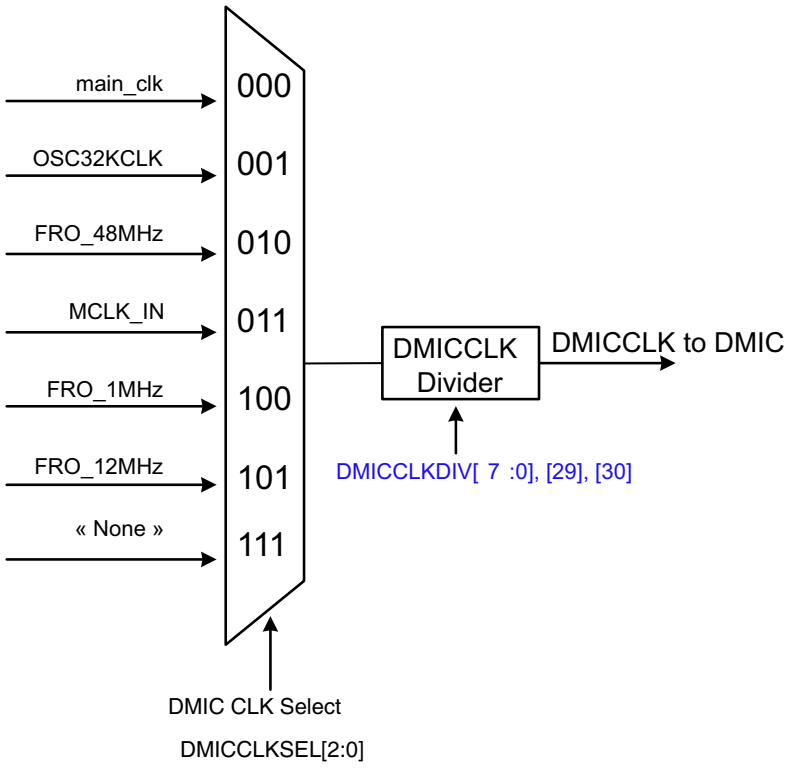


Fig 19. DMIC clock selection

The DMIC IP can receive an external clock (MCLK_IN), as shown in Figure 19 “DMIC clock selection”. This option is provided to have the ability to synchronize the DMIC operation to a clock linked to the audio sub-system.

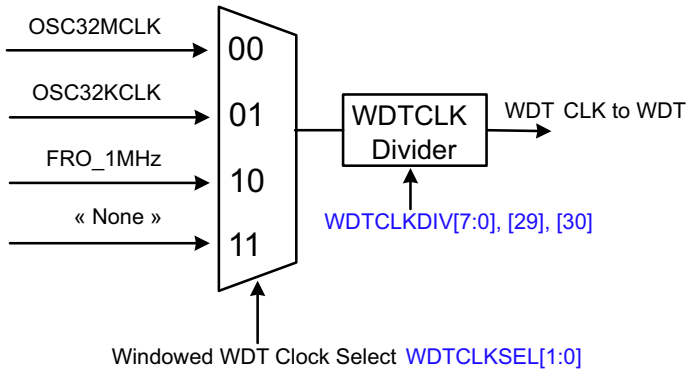


Fig 20. WDT clock selection

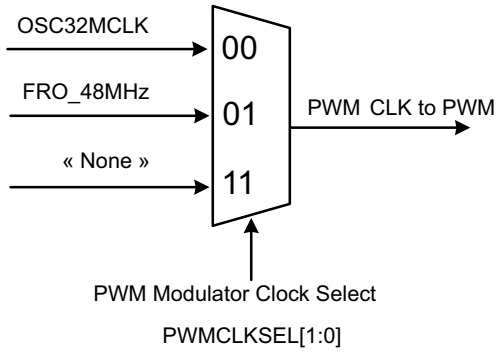


Fig 21. PWM clock

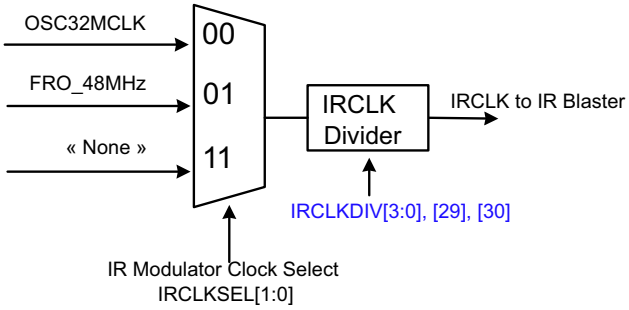


Fig 22. IR Blaster clock

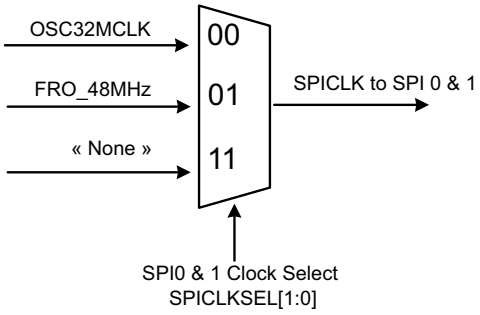


Fig 23. SPICLK selection

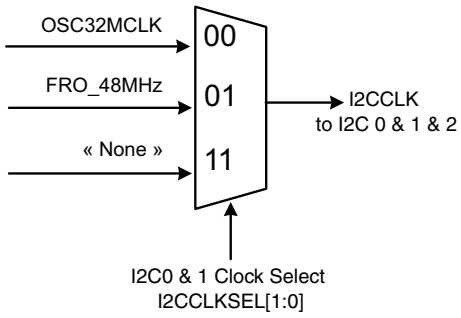


Fig 24. I2C clocks

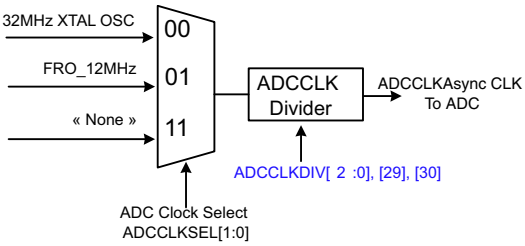


Fig 25. ADC clock generation

6.3.1 Clock outputs

Next table lists all the output clocks generated by the CLK_GEN module, relative sources, division factor(s) and clock gating support availability:

Table 10. Clock outputs

Clock	Source	Division	Gating	Descriptions
system_ahb_clk[63:1]	main_clk[1]	[1]	Yes	To all blocks hclk clocks
system_ahb_clk[0]	main_clk[1]	[1]	No	To CPU, free-running hclk and frequency measurement
system_async_vpb_clk	All inputs for main_clk + XTAL 32M + FRO32M + FRO48M	None	Yes	Free running aclk (to asynchronous APB bridge)
SYSTICKCLK	main_clk	1:256	No	To SYSTICK clock
TRACECLK	main_clk	1:256	No	To TRACECLK
SPIFICLK	main_clk XTAL32M	1:4	No	To Quad SPI
CLK_32KHz	FRO32K XTAL32K	None	Yes	To CLK32K of USART, LSPI, PMC, frequency measurement
RTC1HzCLK	FRO32K XTAL32K	By 32768	No[2]	To RTC
RTC1KHzCLK	FRO32K XTAL32K	1:32	No[2]	To RTC

Table 10. Clock outputs

Clock	Source	Division	Gating	Descriptions
USART_CLK	main_clk OSC32M FRO48M FRGCLK	None	No	To USART 0 and 1
RNG_CLK0	XTAL32M	None	Yes	
RNG_CLK1	FRO32M	None	Yes	
ZIGBEE_CLK	XTAL32M	/2	No	
CLKOUT	main_clk XTAL32K FRO32K XTAL32M FRO48M FRO1M	1:16	No	
WKTCLK[1:0]	XTAL32M	None	Yes	To wakeup timers
DMICCLK	main_clk XTAL32K FRO32K FRO12M FRO48M FRO1M MCLK_IN	1:256	No	
WDTCLK	FRO32K FRO32M XTAL32K XTAL32M FRO1M	1:256	No	
PWMCLK	XTAL32M FRO32M FRO48M	None	No	
IRCLK	XTAL32M FRO32M FRO48M	1:16	No	
ADCCLK	XTAL32M FRO12M	1:8	No	
I2CCLK[1:0]	XTAL32M FRO32M FRO48M	None	No	
SPICLK[1:0]	XTAL32M FRO32M FRO48M	None	No	
CONTROLLED_FRO1MHz	FRO1M	None	Yes	To Frequency Measure
CONTROLLED_32MHz XTAL_OSC	XTAL32M	None	Yes	To Frequency Measure

[1] Frequency and division factors restricted to the following final system clock frequencies: FRO12MHz (Default CPU boot), FRO 16MHz, FRO 24MHz, FRO 32MHz, FRO48 MHz, XTAL 32MHz, XTAL16MHz.

[2] The RTC clock can be gated within the RTC module.

6.3.2 Clock enable and switching

6.3.2.1 Clock switching during active mode

All the clock gating logic and clock dividers are glitch-free (safe). MAINCLKSEL and ASYNCAPBCLKSEL muxes are safe too. This means that the system clock can be safely switched in active mode.

All the remaining clock muxes described in this chapter cannot be considered glitch-free, meaning that every time a clock switching is needed, a safe procedure must be used.

Two different situations can be identified:

Case 1) A clock divider is available in CLK_GEN (i.e. for SPIFI):

1. Disable SPIFI clock divider, by writing “1” to the SYSCON_SPIFICLKDIV[HALT]
2. Set SPIFI source clock by writing the selected value in SYSCON_SPIFICLKSEL register
3. Sets SPIFI clock divider value by writing the selected value to the SYSCON_SPIFICLKDIV[DIV]
4. Reset SPIFI clock divider by writing “1” to the SYSCON_SPIFICLKDIV[RESET]
5. Release the Reset on the clock divider by clearing the SYSCON_SPIFICLKDIV[RESET]
6. Enables SPIFI clock divider (clear the SYSCON_SPIFICLKDIV[HALT])

Case 2) No clock divider available in CLK_GEN (i.e. I^2C)

As the clock mux is not glitch-free, and no divider with HALT and RESET features is available, if the application requires clock switching during activity, the safe procedure is:

1. Put the peripheral under reset, by using the corresponding bitfield of SYSCON_PRESETCTRLSET0 and SYSCON_PRESTCTRLSET1 registers, which are described in Chapter 10 “System Configuration (SYSCON)”.
2. Set the peripheral source clock
3. Release the reset by clearing the peripheral reset bit by using the corresponding bitfield of SYSCON_PRESETCTRLCLR0 and SYSCON_PRESETCTRLCLR1.

This procedure is also recommended for peripheral initialization.

6.3.2.2 Clock selection at power-up and initialization

At power-up, the hardware boot is managed by the PMC state machine. It enables the High Speed FRO, and after the power-up of analog blocks and applying trim values from efuse, it releases the CPU reset.

At the software boot entry point:

- All the clock muxes take the reset value, meaning that the main_clk is connected to high speed FRO(see SYSCON_MAINCLKSEL register).

- All the AHB clocks are gated, except those to the CPU, ROM and FLASH.

Then the boot code:

- Enable clock to SRAM0, SRAM1 and the Flash (by writing a 1 to the corresponding SYSCON_AHBCLKCTRL0 bitfields) (gating is removed)

At CPU boot, the system clock is FRO12MHz.

When leaving the boot (jump to the first instruction of the application), all the clock settings have their reset values, excepts CPU, ROM Flash, SRAM0 and SRAM1. It means that the application needs to select and enable the needed clocks for peripherals, by following the safe procedure(s) described in Section 6.3.2.1 “Clock switching during active mode”.

The system clock frequency needs to be adjusted by the application to meet best compromise between power consumption and product performance requirements.

6.3.2.3 Wake-up from low-power mode

When going to power down mode, the system clock frequency is switched back to 12 MHz (just before going to power down) thus the initial phase of the wake up sequence is performed at 12 MHz.

The system clock frequency is switched back to application frequency at the end of the wake-up phase (by the ROM code)

When waking up from a low-power mode, the Sleep controller and the PMC handle the wake-up sequence. All the blocks which are powered-off are automatically reset by the Sleep Configuration. No action is required by the software.

In addition, in Deep Sleep and Power Down modes, the clock gating and muxing configuration are kept as SYSCON is still powered.

The COMM0 power domain can be optionally switched on or off, according to the application needs. If it is switched OFF, then the PMC/Sleep controller will not handle the recovery / re-initialization procedure for it => the software must ensure that the COMM0 switched-off blocks are reset at wake-up.

6.4 Clock control software functions

Table 11 shows the clocks that can be controlled using the software APIs. Clocks enabled by default should not be altered. Clocks managed by the radio stack software are not shown in this table.

Table 11. Clock sources controllable by software APIs

Clock Name	Description	Enabled by Default?
kCLOCK_Sram0	The clock for SRAM controller0, for SRAM blocks SRAM0 to SRAM7	Yes
kCLOCK_Sram1	The clock for SRAM controller1, for SRAM blocks SRAM8 to SRAM11	Yes
kCLOCK_Spifi	The clock for the Quad SPI Flash controller	No
kCLOCK_InputMux	The clock for the Input Mux	No
kCLOCK_IOCON	The clock for the IOCON module	No
kCLOCK_Pint	The clock for the Pin Interrupt (PINT)	No

Table 11. Clock sources controllable by software APIs

Clock Name	Description	Enabled by Default?
kCLOCK_Gint	The clock for the Group Interrupt (GINT)	No
kCLOCK_Dma	The clock for the DMA	No
kCLOCK_Iso7816	The clock for the ISO7816	No
kCLOCK_WdtOsc	The clock for the Watchdog timer	No
kCLOCK_Rtc	The clock for the RTC	No
kCLOCK_AnaInt	The clock for the Analog Interrupt Controller	No
kCLOCK_WakeTmr	The clock for the Wake up timers	No
kCLOCK_Adc0	The clock for the ADC Controller	No
kCLOCK_Usart0	The clock for the USART0	No
kCLOCK_Usart1	The clock for the USART1	No
kCLOCK_I2c0	The clock for the I2C0	No
kCLOCK_I2c1	The clock for the I2C1	No
kCLOCK_Spi0	The clock for the SPI0	No
kCLOCK_Spi1	The clock for the SPI1	No
kCLOCK_Ir	The clock for the Infra Red	No
kCLOCK_Pwm	The clock for the PWM	Yes
kCLOCK_Rng	The clock for the Random Number Generator	No
kCLOCK_I2c2	The clock for the I2C2	No
kCLOCK_Aes	The clock for the AES	No
kCLOCK_DMic	The clock for the DMIC	No
kCLOCK_Timer0	The clock for the Timer0	No
kCLOCK_Timer1	The clock for the Timer1	No
kCLOCK_Gpio0	The clock for GPIO	Yes
kCLOCK_Sha	The clock for Hash-Crypt peripheral	No
kCLOCK_MainClk	The clock used as MAIN_CLK	Yes
kCLOCK_CoreSysClk	The clock attached to MAIN_CLK	Yes
kCLOCK_BusClk	The clock used on the internal AHB bus	Yes
kCLOCK_Xtal32k	The external 32kHz crystal	No
kCLOCK_Xtal32M	The external 32MHz crystal	No
kCLOCK_Fro32k	The 32kHz clock from the Free Running Oscillator (FRO)	No
kCLOCK_Fro1M	The 1MHz clock from the Free Running Oscillator (FRO)	Yes
kCLOCK_Fro12M	The 12MHz clock from the Free Running Oscillator (FRO)	Yes
kCLOCK_Fro32M	The 32MHz clock from the Free Running Oscillator (FRO)	Yes
kCLOCK_Fro48M	The 48MHz clock from the Free Running Oscillator (FRO)	Yes
kCLOCK_ExtClk	The clock that can be sourced from PIO19	No
kCLOCK_WdtClk	The Watchdog Timer Clock	No
kCLOCK_Frg	The Fractional Rate generator (FRG) that can be used with the USARTS	No
kCLOCK_ClkOut	The clock that can be used to drive PIO17	No
kCLOCK_Fmeas	The clock for frequency measurement	Yes

The previous section showed how most blocks have options on the clock that can be used for the block; i.e. there are multiple clock sources. These are managed within the software using the definition provided in fsl_clock.h

For example, the SPI blocks can take their clock source from one of 2 locations. These are the 32 MHz clock (that can by derived from the 32 MHz crystal or the 32 MHz free running oscillator) and the 48 MHz free running oscillator. On this block there is also the option to switch the clock off. So, within fsl_clock there defines 3 clock sources:

- kOSC32M_to_SPI_CLK
- kFRO48M_to_SPI_CLK
- kNONE_to_SPI_CLK

To select one of these clock sources use:

- CLOCK_AttachClk (kFRO48M_to_SPI_CLK);

In a similar manner, the MAIN_CLK has several sources and so the following definitions have been created:

- kFRO12M_to_MAIN_CLK
- kXTAL32M_to_MAIN_CLK
- kFRO32M_to_MAIN_CLK
- kFRO48M_to_MAIN_CLK

One of these can be selected using: CLOCK_AttachClk (kFRO12M_to_MAIN_CLK);

7.1 Introduction

The JN5189(T)/JN5188(T) device is provided with the following reset sources:

Table 12. Reset

Reset Type	Description
Voltage Resets	System Power-On-Reset (POR)
External Resets	External Pin reset (RESETN)
	Wake-up IO Reset
Internal Resets	Watchdog Timer Reset
	Arm System Reset
	SW Reset
Debug Reset	TRSTN

Each reset source has a corresponding bit in the PMC_RESETCAUSE register, located in the PMC. This information can be used to take appropriate action when coming out of a reset.

7.2 Reset

Asserting a reset to the device, a core, or a peripheral provides a way to start processing from a known set of initial conditions. On de-assertion of a system level reset source, the on-chip regulator is in full regulation and system clock generation is from an internal source. When the device exits reset, the boot core performs the following actions:

- Read the initial Stack Pointer (SP) from vector-table at offset 0x0000_0000
- Read the initial Program Counter (PC) from vector-table at offset 0x0000_0004
- Set the Link Register (LR) to 0xFFFF_FFFF

After a system level reset, the on-chip peripherals are disabled, the non-analog I/O pins return to their default disabled configuration, and the analog I/O pins return to their default analog function configuration.

7.2.1 System Power-On Reset (POR)

The POR voltage monitor is used to ensure that the voltage applied to the system is high enough for analog modules such as the bias circuits and low voltage monitors to operate correctly. When voltage is initially applied to the device or when the supply voltage drops below the Power-On Reset re-arm voltage, the POR monitor asserts POR to the device.

7.2.2 External reset sources

External system resets are provided so that the device can be reset or awakened from very low-power states. This allows the user to start the device at the correct time from a known state. The external system resets available in this device are described in the following sections.

7.2.2.1 External Pin Reset

The RESETN is a dedicated pin on this device. This pin is open drain and has an internal pull-up. Asserting RESETN wakes the device from any mode.

7.2.2.2 Wake-up IO Reset

Some GPIOs can cause a wake-up from low-power mode, if configured the PMC_CTRL[WAKEUPRESETENABLE]. When waking up from deep power-down mode, a wake-up event from IOs causes a general reset. In power-down state, the IO can cause a wake-up event, and also a reset but this is not recommended, as at wake-up from power-down mode, the PMC handles a clean reconfiguration of the blocks that are switched-off.

7.2.3 Internal reset sources

Internal system resets are provided so that the device can be reset when certain erroneous conditions are detected. This allows the device to reset a portion of the device or the entire device to recover from the erroneous conditions. The internal system resets available in this device are described in the following sections.

7.2.3.1 Watchdog Timer Reset

The WDT monitors the operation of the system by receiving periodic communication from the software. This communication is generally known as servicing or refreshing the Watchdog. If this periodic servicing does not occur, then the Watchdog issues a system reset.

7.2.3.2 Software Reset

The software can reset a specific peripheral by writing into its corresponding bit field of the SYSCON_PRESETCTRL0 and SYSCON_PRESETCTRL1 registers for AHB peripherals and into the ASYSCON_ASYNCPRESETCTRL register to reset the peripherals attached to the async APB bridge. Writing a “1“ asserts the reset.

It is also possible to have a chip SW reset, by using the ASYNC_SYSCON[SWRESETCTRL] and PMC_CTRL[SWRRESETENABLE] fields.

7.2.3.3 Arm System Reset

An Arm System reset can be generated by the CPU.

7.2.4 Reset sources summary

Next table summarizes the different reset sources and their effect on all the blocks.

Table 13. Reset outputs from RST_GEN

	Analog														Digital		
	DCDC	Bias (Bandgap)	BOD VBAT	BOD Core	BOD Mem	LDO	32 kHz XTAL	32 MHz XTAL	32kHz FRO	192 MHz FRO	1 MHz FRO	Temperature sensor	General Purpose ADC	Analog comparator	PMC general purpose data storage	All digital units (except the debug access port)	Debug access port (DAP)
Power on reset (POR)	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
External pin reset	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Software reset	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Watchdog timer reset	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Arm system reset	N	N	N	N	N	N	N	Y	N	N	N	Y	Y	N	N	Y	N
Wakeup IO reset	Y	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	Y
Brown out detector (BOD) reset	Y	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	Y

[1] Y: Yes; N: No

7.2.5 Reset management architecture

The previously described reset sources are combined and synchronized by the PMC, in the PMC reset block, as shown in Figure 26.

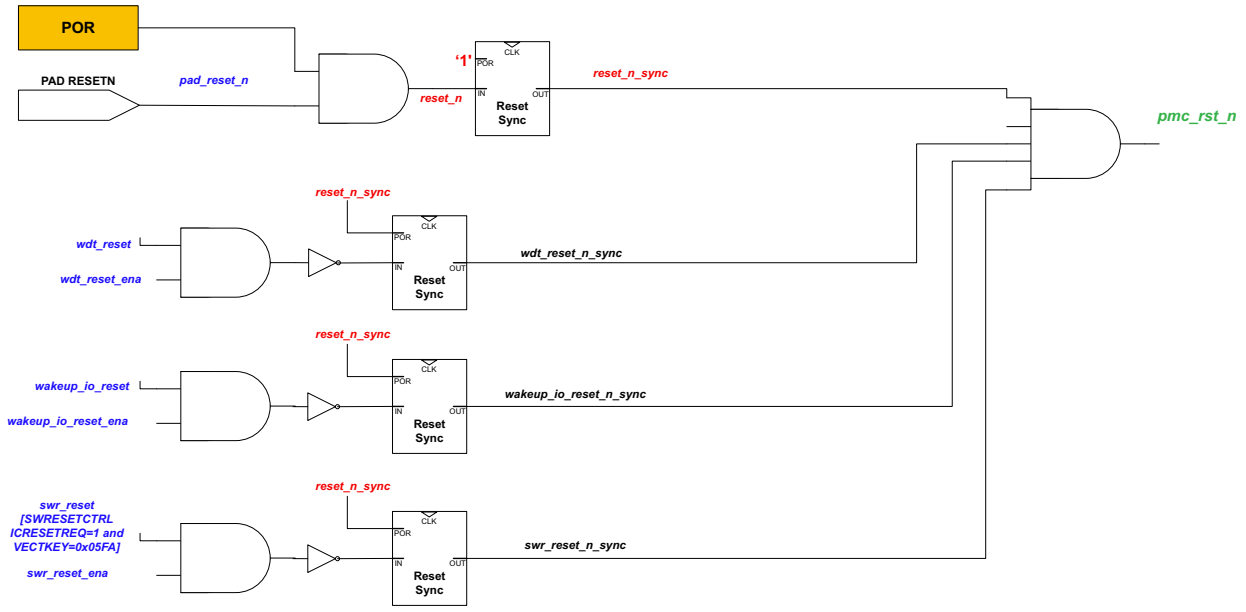


Fig 26. PMC reset block

The PMC logic directly generates the reset signals for the Modem and the Efuse blocks, while the reset inputs to all the remaining peripherals are managed by the Reset Generator (RST_GEN) block, as shown in the next high-level blocking diagram.

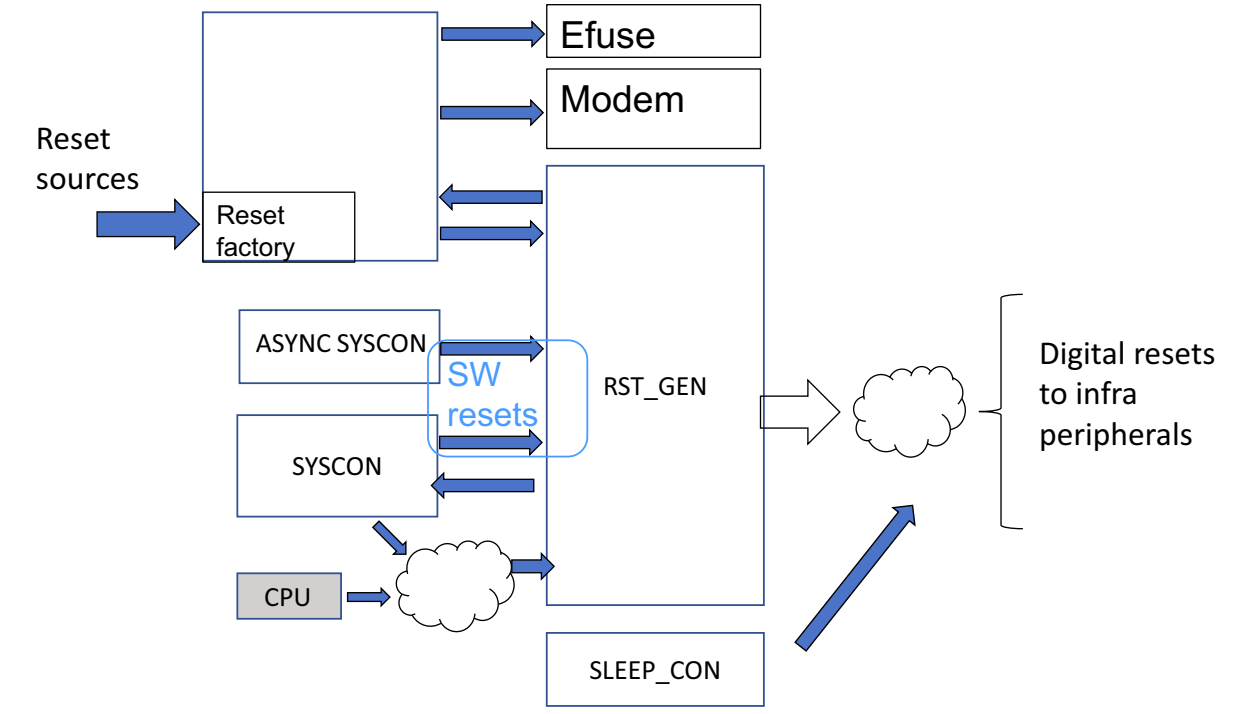


Fig 27. Reset management structure

The RST_GEN receives reset requests from different sources, it combines and synchronizes them and generates the resets for most digital blocks and peripherals.

The SLEEP Controller block must take over the control of CPU and peripherals resets at wake-up from power down and deep power down modes. This is achieved by combining at system level the reset sources from RST_GEN to the reset sources from SLEEP_CON. Each peripheral block can be reset independently by the SYSCON registers.

7.3 Boot

The device is provided with on-chip boot ROM, containing the Boot Loader. The boot code runs when the processor comes out of reset and is responsible for:

- Configuring processor and chip settings (operating mode, processor stack pointer)
- Initializing the application's variable space
- Jumping into the application

It supports also ISP (In-System Programming) access via USART.

7.3.1 Boot modes

The boot mode can be selected and is influenced with different settings and inputs, from external pins and values stored in the flash:

- Flash settings
  - N-2 page: SWD_DIS: 0 for SW to enable SWD, 1 for SW to disable SWD; JTAG_DIS: 0 for SW to enable JTAG, 1 for SW to disable JTAG
  - pFlash_HardwareTestModeEnable enable; ISP access level
- DIO4 and DIO5, allow start-up mode to be specified
  - DIO4=0, DIO5=0: Hardware test mode
  - DIO4=1, DIO5=0: ISP
  - DIO4=x, DIO5=1: Normal boot

There is further information on the SYSCON_CODESECURITYPROT write-once register, used to enable or disable JTAG and SWD, in Chapter 10 “System Configuration (SYSCON)”.

During startup, the boot code reads the state of DIO4 and DIO5. From reset, these IOs have internal pull-up resistors and hence they would both be read as high state. To ensure that, a start-up mode other than normal boot mode is entered, DIO4 and DIO5 must be held low for at least 3 ms after reset is de-asserted. For normal boot mode, they can be left un-asserted during this time. It is not recommended to use DIO4 and DIO5 for any other signal unless the signal will revert to being high during reset. Otherwise it is possible that a resetting or waking device may enter the hardware test mode or ISP mode in error.

The CPU clock defaults to 12 MHz after reset, and the boot code runs at this speed.

Table 14. Boot modes

Mode	PIO0_4	PIO0_5	Flash			Boot SW Actions
			SWD_DIS	JTAG_DIS	Hardware test mode disable	
Hardware test mode requested, denied	0	0	—	—	1	Initialize flash Loop forever Do not enable SWD or JTAG
Hardware test mode requested, allowed	0	0	A	B	0	Initialize flash If A=0, enable SWD If B=0, enable JTAG, else disable it Sit in loop forever
Normal ISP	1	0	A	B	—	Initialize flash If A=0 enable SWD If B=0, enable JTAG, else disable it Enter ISP mode
Normal run	—	1	A	B	—	Initialize flash If A=1, disable SWD If B=0, enable JTAG, else disable it Find and run application

Table 14 shows the possible boot modes. hardware test mode gives a method for allowing control of the device with a debugger, typically this mode would be available during application or product development. After product development, this mode is disabled by setting Flash fields SWD_DIS and JTAG_DIS.

The JTAG option is shown here for completeness and only gives a route for internal test features; it is independent of the SWD debug function. For completeness, it should be disabled before a product is released.

7.3.2 Code protection

The following features are provided to protect the access to the memory:

- SWD and JTAG control
- Configuration of memory protection mechanism (MPU), by the boot code, to minimize threats such as code injection

At the end of the hardware boot sequence, that is, after the release of the CPU reset, and just at the start of the ROM boot code execution, the SW checks the value of the JTAG_DIS field in the flash. The JTAG port, which would only be used for internal test, would then be disabled if the field is set. Hence, all devices must have this JTAG_DIS field set to prevent unwanted use of the test port.

The boot code checks the F_SWD_DIS flag value (stored in the flash); if this flag is set then the SWD port must be disabled. The boot code writes a value to the SYSCON_CODESECURITYPROT register accordingly:

- SWD is enabled if the value written to the SYSCON_CODESECURITYPROT is correct (= 0x87654320). In that case, PIO12 and PIO13 are configured as SWCLK and SWDIO (by default during boot).
- SWD is disabled if the value written to the SYSCON_CODESECURITYPROT is wrong. This will disable the SWD forever, because the register is write once only.

ISP mode is mainly used to allow programming of the device. This mode can be used during product development; then the device should be locked before a product is released. Hardware test mode allows access to the device during code development as a safety route in case of a locked device, this mode should also be locked down after product development.

7.3.3 Boot process

The following figures show the ROM's boot process of the JN5189(T)/JN5188(T). Figure 28 shows the boot sequence from a cold start.

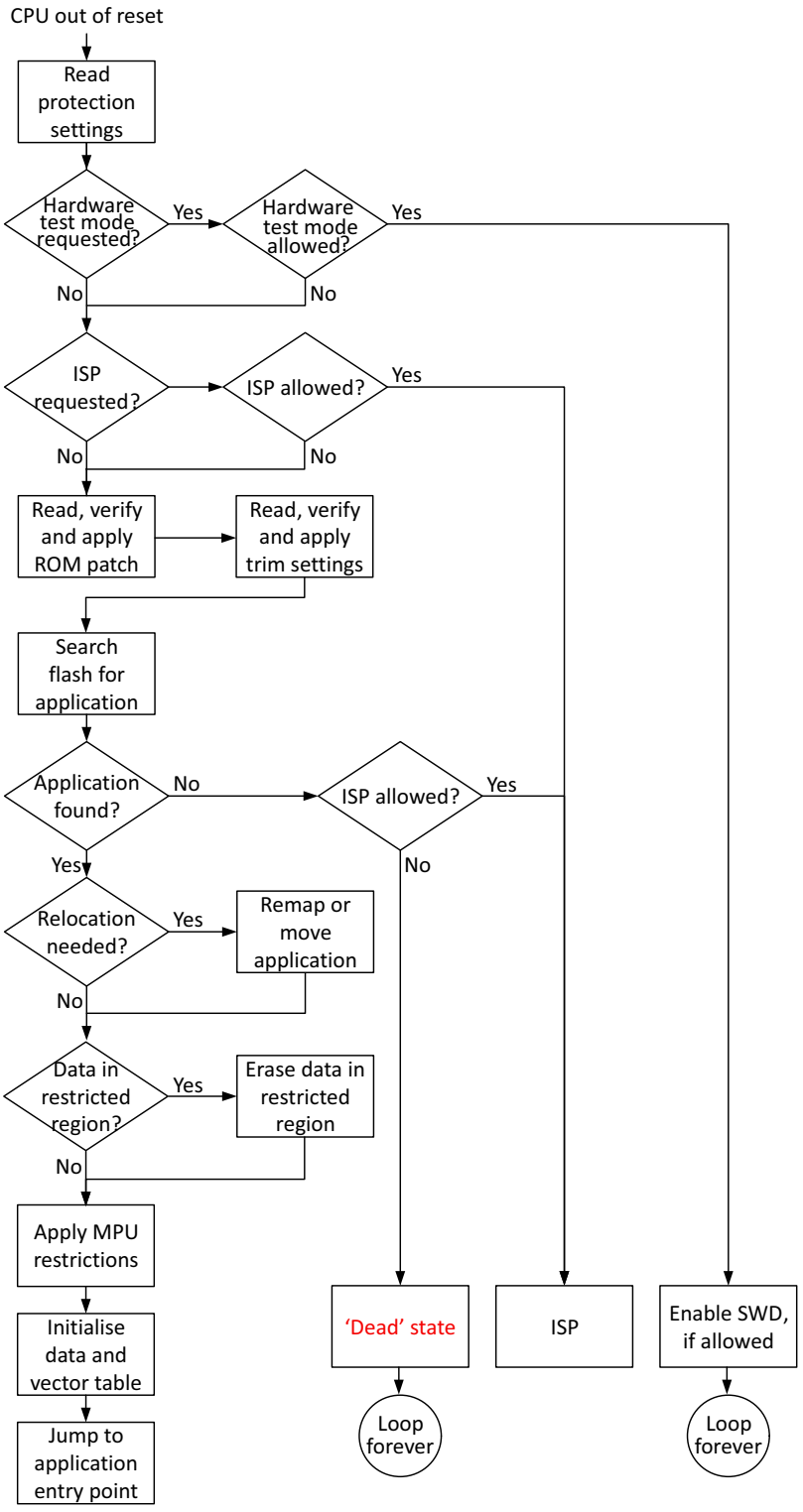


Fig 28. Boot from cold start

Figure 29 shows the boot sequence from a warm start.

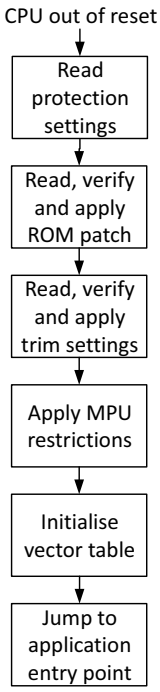


Fig 29. Boot from warm start

Some details about the blocks in the previous diagrams:

The protection settings are stored in the “protected data” region of the flash memory (pFlash), mapped at the address range of 9EC00 - 9EFFF. They contain the ROM patch and several device configuration data.

The hardware test mode can be requested by setting PIO4 and PIO5 low (as shown in Table 14 “Boot modes”); there is a field in the pFlash to allow or prohibit this hardware test mode.

The ROM patch is stored within the pFlash. It is optionally verified by a checksum that the boot loader calculates and compares to the stored value.

- If the ROM patch is not present, or does not match the checksum, no further action is taken.
- If the ROM patch is present and valid, the boot loader performs a jump-and-link to the entry point of the ROM patch.
- If the ROM patch returns, execution of the boot loader will then continue.

The ROM patch is provided by NXP and configured into the flash before out of factory; this description is only for information purposes.

Trim settings are programmed during ATE testing and are stored in the N-2 sector of the flash. These are used to allow software to configure certain functional blocks with optimum settings to overcome process variations.

Application search: if no valid application is found, the boot loader will jump to the ISP. If ISP is disabled, the boot loader will go to a "dead" state.

The ISP state allows a PC-based programmer application to interact through the USART port, and request various actions.

ISP can be used at several levels of access:

- Unrestricted (secure or unsecure): normal developer mode
- Write-only (secure or unsecure): allow updates but existing contents cannot be read
- Locked: only allow limited access, with a secure handshake

Note: Secure means that communication with the PC based programmer application are authenticated; unsecure means no authentication is used.

The default ISP level is unrestricted unsecure access.

See Chapter 38 “In-System Programming (ISP)” for more information.

SWD interface can be enabled if the device is in hardware test mode. This can be used to allow reprogramming of a device. Generally the application must enable the SWD access, if debug capability is required.

7.3.4 Protected regions

Near the top of the flash there is a region assigned as 'protected flash', or pFlash. This holds system settings and configuration data. Some of these fields may need to be modified to suit that requirements of the application being developed. Software utilities are provided to allow modification of these fields.

8.1 Introduction

Power modes are handled by the PMC module (power management control) and the SLEEPCON module (sleep controller).

The PMC organizes and schedules various necessary steps during changes between power modes: wake-up, active, deep sleep, power-down, deep power-down. It contains the state machines.

Waking up from low-power modes is controlled by the SLEEPCON module. Wake-up from IOs are first processed by a dedicated power sub-system module next to the PMC.

Therefore PMC and SLEEPCON work together, but are not active at the same phases.

PMC is configured by its own register interface, while SLEEPCON is configured through SYSCON.

SYSCON (system controller) is used for configuration of most clocks and resets. It also acts as a bridge to SLEEPCON. That is, it provides register interface to control SLEEPCON. See Chapter 10 “System Configuration (SYSCON)” for further details of the SYSCON functions.

There are many complex interactions between these modules. Software APIs in ROM code and SW reference APIs in flash simplify the usage. See Chapter 11 “Power Control API” for further details of the power control API. By using the power APIs, most of the configurations for the low-power modes are managed; this removes the need for understanding many of the registers involved in this implementation.

It is not recommended to access the registers handling power modes directly.

A complete description of registers of PMC and SLEEPCON (subpart of SYSCON registers) is centralized in another part of this user manual and thus will not be duplicated here.

Further details of the power modes are provided in the next section.

8.2 Power modes

Power modes are highly configurable.

Some examples of the possible modes are shown in Table 15.:

Table 15. Possible power modes and state of power domains

Power modes		Power domains											
		PD_MCU	PD_SYSTEM	PD_COMM0	PD_AON	PD_IO_RESET	PD_IO	PD_MEM0(16m)	PD_MEM5(8 K)	PD_MEM6 (4K)	PD_MEM7 (4K)	Other MEM banks	PD_FLASH
Active	PM_ACTIVE template	On	On	On	On	On	On	On	On	On	On	On	On
	PM_ACTIVE_MIN_REF	On	On	On	On	On	On	On	On	On	On	On	On
	PM_ACTIVE_RADIO	On	On	On	On	On	On	On	On	On	On	On	On
	PM_ACTIVE_SINGLE	On	On	On	On	On	On	On	On	On	On	On	On
	PM_ACTIVE_DUAL	On	On	On	On	On	On	On	On	On	On	On	On
Sleep	PM_SLEEP Modes	On	On	On	On	On	On	On	On	On	On	On	On
Deep sleep	PM_DEEP_SLEEP template	On	On	On	On	On	On	RET/ On	RET/ On	RET/ On	RET/ On	RET/ Off	On/ Off
	PM_DEEP_SLEEP_REF_MIN	On	On	On	On	On	On	RET	RET	RET	RET	RET	Off
	PM_DEEP_SLEEP_XTAL	On	On	On	On	On	On	RET	RET	RET	RET	RET	Off
	PM_DEEP_SLEEP_FULL	On	On	On	On	On	On	On	On	On	On	On	On
Power down	PM_DOWN template	RET /On	On	On/ Off	On	On	On	RET/ Off	RET/ Off	RET/ Off	RET/ Off	RET/ Off	Off
	PM_DOWN_REF_MIN	Off	On	Off	On	On	On	Off	Off	Off	Off	Off	Off
	PM_DOWN_NTAG	Off	On	Off	On	On	On	Off	Off	Off	Off	Off	Off
	PM_DOWN_FRO	Off	On	Off	On	On	On	Off	Off	Off	Off	Off	Off
	PM_DOWN_XTAL	Off	On	Off	On	On	On	Off	Off	Off	Off	Off	Off
	PM_DOWN_4K	Off	On	Off	On	On	On	Off	Off	Off	RET	Off	Off
	PM_DOWN_16K	Off	On	Off	On	On	On	Off	RET	RET	RET	Off	Off
	PM_DOWN_COM	Off	On	On	On	On	On	Off	RET	RET	RET	Off	Off
	PM_DOWN_ANA_COMP	Off	On	Off	On	On	On	Off	RET	RET	RET	Off	Off
	PM_DOWN_BODVBAT	Off	On	Off	On	On	On	Off	RET	RET	RET	Off	Off
	PM_DOWN_NTAG_16K	Off	On	Off	On	On	On	Off	RET	RET	RET	Off	Off
	PM_DOWN_FULL_MEM	Off	On	Off	On	On	On	RET	RET	RET	RET	RET	Off
	PM_DOWN_CAL_RET	RET	On	Off	On	On	On	Off	RET	RET	RET	Off	Off
	PM_DOWN_FULL	RET	On	ON	On	On	On	Off	RET	RET	RET	Off	Off
Deep power down	PM_DEEP_DOWN Template	Off	Off	Off	On	On	On	Off	Off	Off	Off	Off	Off
	PM_DEEP_DOWN_REF	Off	Off	Off	On	On	Off	Off	Off	Off	Off	Off	Off
	PM_DEEP_DOWN_NTAG	Off	Off	Off	On	On	Off	Off	Off	Off	Off	Off	Off
	PM_DEEP_DOWN_NIO	Off	Off	Off	On	On	On	Off	Off	Off	Off	Off	Off

Table 16. Proposed power modes and state of analog modules

Power modes		Analog modules													
		GP ADC (LDO)	Analog comparator	DCDC converter	Temperature sensor	Power on reset	BIAS	BODVBAT	XTAL 32 MHz	XTAL 32 kHz	FRO 1 MHz	FRO 192 MHz	FRO 32 kHz	LDO ADC	Radio
Active	PM_ACTIVE template	On/ Off	On/ Off	On	On/ Off	On	On	On	On/ Off	On/ Off	On	On	On/ Off	On/ Off	On/ Off
	PM_ACTIVE_MIN_REF	Off	Off	On	Off	On	On	On	Off	Off	On	On	On	On	Off
	PM_ACTIVE_RADIO	Off	Off	On	Off	On	On	On	On	Off	On	On	On	On	On
	PM_ACTIVE_SINGLE	On	On	On	On	On	On	On	On	On	On	On	Off	On	On
	PM_ACTIVE_DUAL	On	On	On	On	On	On	On	On	On	On	On	Off	On	On
Sleep	PM_SLEEP Modes	On/ Off	On/ Off	On	On/ Off	On	On/ Off	On/ Off	On/ Off	On/ Off	On	On	On/ Off	On/ Off	On/ Off
Deep sleep	PM_DEEP_SLEEP template	On/ Off	On/ Off	On	On/ Off	On	On	On	On/ Off	On/ Off	On	On/ Off	On/ Off	On/ Off	On/ Off
	PM_DEEP_SLEEP_REF_MIN	Off	Off	On	Off	On	On	On	Off	Off	On	Off	On	Off	Off
	PM_DEEP_SLEEP_XTAL	Off	Off	On	Off	On	On	On	Off	On	On	Off	Off	Off	Off
	PM_DEEP_SLEEP_FULL	On	On	On	On	On	On	On	On	On	On	On	On	On	Off
Power down	PM_DOWN template	Off	On/ Off	Off	Off	On	On/ Off	On/ Off	Off	On/ Off	Off	Off	On/ Off	Off	Off
	PM_DOWN_REF_MIN	Off	Off	Off	Off	On	Off	Off	Off	Off	Off	Off	Off	Off	Off
	PM_DOWN_NTAG	Off	Off	Off	Off	On	Off	Off	Off	Off	Off	Off	Off	Off	Off
	PM_DOWN_FRO	Off	Off	Off	Off	On	Off	Off	Off	Off	Off	Off	On	Off	Off
	PM_DOWN_XTAL	Off	Off	Off	Off	On	Off	Off	Off	On	Off	Off	Off	Off	Off
	PM_DOWN_4K	Off	Off	Off	Off	On	Off	Off	Off	Off	Off	Off	Off	Off	Off
	PM_DOWN_16K	Off	Off	Off	Off	On	Off	Off	Off	Off	Off	Off	On	Off	Off
	PM_DOWN_COM	Off	Off	Off	Off	On	Off	Off	Off	Off	Off	Off	On	Off	Off
	PM_DOWN_ANA_COMP	Off	On	Off	Off	On	Off	Off	Off	Off	Off	Off	Off	Off	Off
	PM_DOWN_BODVBAT	Off	Off	Off	Off	On	On	On	Off	Off	Off	Off	Off	Off	Off
	PM_DOWN_NTAG_16K	Off	Off	Off	Off	On	Off	Off	Off	Off	Off	Off	Off	Off	Off
	PM_DOWN_FULL_MEM	Off	Off	Off	Off	On	Off	Off	Off	Off	Off	Off	Off	Off	Off
	PM_DOWN_CAL_RET	Off	Off	Off	Off	On	Off	Off	Off	Off	Off	Off	On	Off	Off
	PM_DOWN_FULL	Off	On	Off	Off	On	On	On	Off	On	Off	Off	On	Off	Off
Deep power down	PM_DEEP_DOWN Template	Off	Off	Off	Off	On	Off	Off	Off	Off	Off	Off	Off	Off	Off
	PM_DEEP_DOWN_REF	Off	Off	Off	Off	On	Off	Off	Off	Off	Off	Off	Off	Off	Off
	PM_DEEP_DOWN_NTAG	Off	Off	Off	Off	On	Off	Off	Off	Off	Off	Off	Off	Off	Off
	PM_DEEP_DOWN_NIO	Off	Off	Off	Off	On	Off	Off	Off	Off	Off	Off	Off	Off	Off

8.3 Low level drivers APIs

The low level driver APIs provide the way for the application to request a specific power mode. The API is then responsible for performing any required configuration and sequencing to put the device into the required mode. As a developer, it is not necessary to know the detail of this. However, to give some insight into the operations, the following list shows the functionality that may be controlled within the API functions.

1. Set clocks. For example, switch off or lower clock speed, activate FRO to replace XTAL,
2. Enable data retention mode in the radio controller
3. Reset unused modules
4. Deactivate power domains, to reduce power consumption
5. Configure LDOs to operate at lower voltage levels when in the low-power mode
6. Set wake-up conditions: on timer, event on IOs, interrupt from internal modules

The PMC itself will schedule most of these operations, based on settings put in registers. See Chapter 11 “Power Control API” for further information.

9.1 How to read this chapter

Available interrupt sources vary slightly with specific JN5189(T)/JN5188(T) device type. A device with internal NTAG will have an additional interrupt source.

9.2 Features

- Nested Vectored Interrupt Controller that is an integral part of the CPU.
- Tightly coupled interrupt controller provides low interrupt latency.
- Controls system exceptions and peripheral interrupts.
- The NVIC of the Cortex-M4 supports:
  - a large array of vectored interrupt slots.
  - 8 programmable interrupt priority levels with hardware priority level masking.
  - Vector table offset register VTOR.
- Support for NMI from any interrupt (see Section 9.3.2 “Non-maskable interrupt”).

9.3 General description

The tight coupling of the NVIC to the CPU allows for low interrupt latency and efficient processing of late arriving interrupts.

9.3.1 Interrupt sources

Table 17 lists the interrupt sources for each peripheral function. Each peripheral device may have one or more interrupt lines to the Vectored Interrupt Controller. Each line may represent more than one interrupt source. The interrupt number does not imply any interrupt priority; interrupt priorities are configured within the NVIC.

The interrupt source table shows the name of the defined interrupt handler. Each IRQ is defined as 'weak'. Naming a function with these names will cause this function to become the default handler for the interrupt.

See Ref. 1 “Cortex-M4 TRM” for detailed descriptions of the NVIC and the NVIC registers.

Table 17. Connection of interrupt sources to the NVIC

Interrupt source number	Interrupt source name	Interrupt Handler	Descriptions
-14	NonMaskableInt_IRQn	NMI_Handler	Cortex-M4 Non Maskable Interrupt
-13	HardFault_IRQn	HardFault_Handler	Cortex-M4 SV Hard Fault Interrupt
-12	MemoryManagement_IRQn	MemManage_Handler	Cortex-M4 Memory Management Interrupt
-11	BusFault_IRQn	BusFault_Handler	Cortex-M4 Bus Fault Interrupt
-10	UsageFault_IRQn	UsageFault_Handler	Cortex-M4 Usage Fault Interrupt
-5	SVCall_IRQn	SVC_Handler	Cortex-M4 SV Call Interrupt

Table 17. Connection of interrupt sources to the NVIC

Interrupt source number	Interrupt source name	Interrupt Handler	Descriptions
-4	DebugMonitor_IRQn	DebugMon_Handler	Cortex-M4 Debug Monitor Interrupt
-2	PendSV_IRQn	PendSV_Handler	Cortex-M4 Pend SV Interrupt
-1	SysTick_IRQn	SysTick_Handler	Cortex-M4 System Tick Interrupt
0	System_IRQn	System_IRQHandler	System (BOD, Watchdog Timer, Flash controller) interrupt
1	DMA_IRQn	DMA0_IRQHandler	DMA interrupt
2	GINT_IRQn	GINT0_IRQHandler	GPIO global interrupt
3	IRBlaster_IRQn	CIC_IRB_DriverIRQHandler	Infra Red Blaster interrupt
4	PINT0_IRQn	PIN_INT0_DriverIRQHandler	Pin Interrupt and Pattern matching 0
5	PINT1_IRQn	PIN_INT1_DriverIRQHandler	Pin Interrupt and Pattern matching 1
6	PINT2_IRQn	PIN_INT2_DriverIRQHandler	Pin Interrupt and Pattern matching 2
7	PINT3_IRQn	PIN_INT3_DriverIRQHandler	Pin Interrupt and Pattern matching 3
8	SPIFI_IRQn	SPIFI_IRQHandler	Quad-SPI flash interface interrupt
9	Timer0_IRQn	CTIMER0_DriverIRQHandler	Counter/Timer 0 interrupt
10	Timer1_IRQn	CTIMER1_DriverIRQHandler	Counter/Timer 1 interrupt
11	USART0_IRQn	FLEXCOMM0_DriverIRQHandler	USART 0 interrupt
12	USART1_IRQn	FLEXCOMM1_DriverIRQHandler	USART 1 interrupt
13	I2C0_IRQn	FLEXCOMM2_DriverIRQHandler	I2C 0 interrupt
14	I2C1_IRQn	FLEXCOMM3_DriverIRQHandler	I2C 1 interrupt
15	SPI0_IRQn	FLEXCOMM4_DriverIRQHandler	SPI 0 interrupt
16	SPI1_IRQn	FLEXCOMM5_DriverIRQHandler	SPI 1 interrupt
17	PWM0_IRQn	PWM0_IRQHandler	PWM channel 0 interrupt
18	PWM1_IRQn	PWM1_IRQHandler	PWM channel 1 interrupt
19	PWM2_IRQn	PWM2_IRQHandler	PWM channel 2 interrupt
20	PWM3_IRQn	PWM3_IRQHandler	PWM channel 3 interrupt
21	PWM4_IRQn	PWM4_IRQHandler	PWM channel 4 interrupt
22	PWM5_IRQn	PWM5_IRQHandler	PWM channel 5 interrupt
23	PWM6_IRQn	PWM6_IRQHandler	PWM channel 6 interrupt
24	PWM7_IRQn	PWM7_IRQHandler	PWM channel 7 interrupt
25	PWM8_IRQn	PWM8_IRQHandler	PWM channel 8 interrupt
26	PWM9_IRQn	PWM90_IRQHandler	PWM channel 9 interrupt
27	PWM10_IRQn	PWM10_IRQHandler	PWM channel 10 interrupt
28	I2C2_IRQn	FLEXCOMM6_DriverIRQHandler	I2C 2 interrupt
29	RTC_IRQn	RTC_IRQHandler	Real Time Clock interrupt
30	NFCTag_IRQn	NFCTag_IRQHandler	NFC Tag interrupt (Device with internal NTAG only)
32	ADC_SEQ_IRQn	ADC_SEQ_IRQHandler	ADC Sequence A interrupt
34	ADC_THCMP_OVR_IRQn	ADC_THCMP_OVR_IRQHandler	ADC Threshold compare and overrun interrupt
35	DMIC_IRQn	DMIC0_IRQHandler	DMIC interrupt

Table 17. Connection of interrupt sources to the NVIC

Interrupt source number	Interrupt source name	Interrupt Handler	Descriptions
36	HWVAD_IRQn	HWVAD0_IRQHandler	Hardware Voice activity detection interrupt
42	ZIGBEE_MAC_IRQn	ZIGBEE_MAC_IRQHandler	Zigbee/Thread MAC interrupt
43	ZIGBEE_MODEM_IRQn	ZIGBEE_MODEM_IRQHandler	Zigbee/Thread Modem interrupt
44	RFP_TMU_IRQn	RFP_TMU_IRQHandler	RFP Timing Management Unit (TMU) interrupt
45	RFP_AGC_IRQn	RFP_AGC_IRQHandler	RFP AGC interrupt
46	ISO7816_IRQn	ISO7816_IRQHandler	ISO7816 controller interrupt
47	ANA_COMP_IRQn	ANA_COMP_IRQHandler	Analog Comparator interrupt
48	WAKE_UP_TIMER0_IRQn	WAKE_UP_TIMER0_IRQHandler	Wake up Timer 0 interrupt
49	WAKE_UP_TIMER1_IRQn	WAKE_UP_TIMER1_IRQHandler	Wake up Timer 1 interrupt
55	SHA_IRQn	SHA_IRQHandler	SHA interrupt

9.3.2 Non-maskable interrupt

The NVIC supports a non-maskable interrupt (NMI) which is an interrupt source that can not be disabled and is the second highest priority interrupt after the reset interrupt. This feature is configured by the SYSCON_NMISRC register.

To use the NMI, it is necessary to enable it; it is disabled by default. The source for the NMI can be any of the interrupt sources already identified in Section 9.3.1 “Interrupt sources”.

The SYSCON_NMISRC[IRQM40] selects the interrupt source to drive the NMI of the processor. For instance, setting this to 1 would select the DMA interrupt. The NMI is enabled for the processor by setting the NMISRC[NMIENM40]. If the NMI interrupt can cause the CPU to wake from sleep.

9.3.3 Vector table offset

Within the NVIC, it is possible to configure the location of the vector table using the standard Cortex NVIC registers (SCB->VTOR). This is managed by the boot code.

9.3.4 Interrupt priorities

Each interrupt has a configurable priority; 8 different priority levels are supported. Priority 0 is the highest; this is the default setting. The lowest priority is 7. The priorities are configured by the NVIC registers of each processor. Therefore, each processor can have its own settings and priority settings can be set as required by the application.

10.1 Introduction

The system configuration module is used for many different purposes. Located in power domain "system", it is effective in most power modes: active, sleep, deep-sleep, power-down. Thus the only power mode which is not supported is deep power-down.

It contains many registers either used for its own purpose or for independent slave modules: sleep controller (SLEEPCON), reset controller, clock controller, analog interrupt controller, timers. These latter are described in other chapters of the user manual and thus will not be detailed further here.

The SYSCON registers are reset by pin reset, watchdog reset, brownout reset, power-on reset, Arm System reset and software reset.

10.2 Flash remapping

The device provides hardware support for having two different applications, APP_0 and APP_1 in the flash. In addition, each application area can be optionally divided into two so that there can be an active and inactive images. This allows for OTA support where a new image may arrive via the radio and then need to be switched to be the active image at a certain point. It is also possible to just define one application space and have this divided into an active and inactive region.

This feature is used by the boot loader and the selective OTA code; it should not be used by application code.

10.3 System tick clock

System tick clock is a clock sent to the CPU to serve as a reference for counting delays. Some calibration information can be provided for even better accuracy.

Two registers are used for this clock, which is derived from main clock: SYSTCKCAL and SYSTICKCLKDIV.

The System tick timer calibration value can be configured by SYSTCKCAL[CAL] and also read back from the register within the Cortex STCL, SysTick Calibration value register. This calibration value is used to indicate the number of system tick clock periods to give a 10 ms period. For example, if the system tick clock period was 5 µs then it would require a calibration value of 2000 (10 ms/5 µs). See also SYSTICKCLKDIV register.

SYSTCKCAL goes directly to an internal register of Cortex-M CPU, with a minor re-mapping (e.g. CAL becomes TENMS):

```
Bits  Name  Function

[31]  NOREF

Indicates whether the device provides a reference clock to the processor:

0 = reference clock provided

1 = no reference clock provided

If your device does not provide a reference clock, the SYST_CSR.CLKSOURCE bit
     reads-as-one and ignores writes.

[30]  SKEW

Indicates whether the TENMS value is exact:

0 = TENMS value is exact

1 = TENMS value is inexact, or not given.

An inexact TENMS value can affect the suitability of SysTick as a software real time
     clock.

[29:24]-   Reserved.

[23:0]TENMSReload value for 10ms (100Hz) timing, subject to system clock skew errors.
     If the value reads as zero, the calibration value is not known.
```

System tick clock comes from main clock, which is set by the MAINCLKSEL register, divided by a rate which is set by SYSTICKCLKDIV register. Therefore, take care of actual frequency of main clock to compute SYSTCK[CAL] value.

10.4 Non maskable interrupts

Some regular interrupts can be selected to trigger a non-maskable interrupts. See register NMISRC for details.

10.5 Accessing peripherals through internal buses

There are several internal buses within the IC and consequently, some bridges between these buses.

It is necessary to enable async APB bridge to access async system controller (ASYNC_SYSCON) and timer modules (CTIMER0 and CTIMER1), with the ASYNCAPBCTRL register. The bridge can be enabled with the software function CLOCK_EnableAPBBridge().

It is recommended to keep this bridge enabled, but a very small amount of power could be saved by disabling it, using CLOCK_DisableAPBBridge(), when not required.

For each individual module connected on these buses, accessing its registers requires to enable its own register clock. This is done through registers AHBCLKCTRL0 and AHBCLKCTRL1 of which all bits are configurable. Alternatively, use AHBCLKCTRLSET0 and AHBCLKCTRLSET1 to enable clocks and use AHBCLKCTRLCLR0 and AHBCLKCTRLCLR1 to disable clocks.

Note, accessing registers which do not have an active AHB clock may stall the bus and consequently lead to a system failure. Some registers within the device are 'write only' registers. If software attempts to read a 'write only' register, it may cause an exception or potentially lock-up the device. Hence, software must not read a 'write only' register.

If a module's clock has been disabled to save power then care is required to ensure that the clock is re-enabled before any further register accesses are made to the module.

Peripheral modules need to be released from their reset state before being used. This is done through registers PRESETCTRL0 and PRESETCTRL1, where block resets can be set and cleared. For setting or clearing resets, the PRESETCTRLCLR0, PRESETCTRLCLR1, PRESETCTRLSET0 and PRESETCTRLSET1 can be used. The function RESET_PeripheralReset() should be used to manage the resets of the peripheral modules.

10.6 Clock selection

Chapter 6 “Clock Distribution” describes the system clocks; these are controlled and configured by registers in the SYSCON module.

In general, clocks can be enabled, configured with a division ratio, driven with various sources: FROs, oscillators, etc. See registers *CLKSEL for source selection (If it is required to disable a clock, then the selection of TESTCLK will result in no clock since the device is in functional mode.). Each clock divider has its own control register, these registers are named *CLKDIV. Within these registers it is possible to set the divide ratio. Also, using the RESET and HALT fields the divider can be stopped and started; by default it is stopped due to the HALT field being set.

CLOCKGENUPDATELOCKOUT[LOCK] is a protection bit for the 2 sets of registers above. Writing 1 to this bit prevents writing to registers *CLKSEL and *CLKDIV.

It is recommended to use CLOCK_AttacheCLK. For the SPI, the following clocks are defined in fsl_clock:

- kOSC32M_to_SPI_CLK
- kFRO48M_to_SPI_CLK
- kNONE_to_SPI_CLK

Select one of these clock sources using CLOCK_AttachClk (kFRO48M_to_SPI_CLK);

10.7 TRNG control

TRNG stands for true random number generation. Within the SYSCON module the register RNGCLKCTRL[ENABLE] enables input clocks that are used within the TRNG to generate randomness. In addition, the AHB clock for the random number generator must be enabled within the AHBCLKCTRL1 register to allow monitoring of entropy creation (CHI computing) and read back of the random numbers. Since the input clocks to the TRNG are FRO32M and XTAL32M, those clocks need to be enabled as well.

If any of these clocks are lacking, access to the RNG IP will be blocked, which will lead to a system failure.

The SDK supports use of the random number generator with the following functions:

```
status_t TRNG_GetDefaultConfig ( trng_config_t *userConfig );

status_t TRNG_Init ( RNG_Type *base, const trng_config_t *userConfig );

void TRNG_Deinit(RNG_Type *base);

status_t TRNG_GetRandomData(RNG_Type *base, void *data, size_t data_size);
```

10.8 Wake-up interrupts

In order to wake-up from low-power modes, some event sources must be enabled by registers STARTER*, which are used by module SLEEPCON. These registers are configured within POWER_SetLowPower() and thus are not expected to be used directly by end user.

All sources of wake-up listed in STARTER0 and STARTER1 registers can be used to exit deep sleep-mode. But only a subset of them can be used to exit power down mode; they are: WDT_BOD, USART0, I2C0, SPI0, RTC, ANA_COMP, WAKE_UP_TIMER0, WAKE_UP_TIMER1, GPIO (including NFC tag when available in the part).

10.9 PIO retention control

In power down mode (only), some PIO pins may need to be maintained as outputs. Since most peripherals are powered-off at that time, they cannot drive these outputs directly. A mechanism is available to latch and maintain current value which is put out on these PIO pins.

If it is necessary to clamp at least one PIO pin during power down, then it necessary to perform the following steps:

1. Activate IOCLAMP feature in appropriate IOCON_PIOn registers
2. Set RETENTIONCTRL[IOCLAMP] bit to enable the feature

3. Disable peripherals
4. Go to power down mode

After wake-up from power-down, it is necessary to release the clamping to allow for normal operation. However, to prevent the IOs from being disturbed, it is necessary to follow this procedure:

1. Wake from power down mode
2. Activate peripherals
3. Disable RETENTIONCTRL[IOCLAMP] bit. This disables the whole clamp function and so it is not necessary to change the IOCLAMP values within the IOCON_PIOn[SSEL] or IOCON_PIOn[IO_CLAMP].

10.10 Interrupts from analog modules

SYSCON provides registers for analog interrupt controller module. Register ANACTRL_CTRL controls on how the interrupts from analog interrupt controller module are used (level, edge, etc).

Same mechanism for interrupts coming from BODs. Registers are ANACTRL_*.

10.11 Additional clock controls

On top of registers used to enable bus clocks of peripherals, SYSCON provides controls for specific clocks. See register CLOCK_CTRL.

10.12 Low power wake-up timers

The Wakeup Timers provide a low-power timer function that can run in Sleep, Deep Sleep or Power down modes. Software support of these functions is provided by fsl_wtimer functions.

There are two low-power timers which can operate independently. The counter can be programmed with a count value and enabled. The counter decrements until it reaches 0. At this point, it can create an interrupt or, if the device is in power-down, a wakeup event. The counter will wrap and then continue decrementing so that software can determine how long ago the interrupt occurred.

Features:

- Wake timer 0 has 41 bits
- Wake timer 1 has 28 bits
- Setting start value for timers
  - Wake timer 0 can be loaded by WKT_LOAD_WKT0_LSB register and then writing the 9 most significant bits to WKT_LOAD_WKT0_MSB.
  - Wake timer 1 can be loaded using WKT_LOAD_WKT1 register
  - Set start value when the timer is not enabled and then enable the timer
- Wake timer current counter values can be read

  - Wake timer0 value can be read through WKT_VAL_WKT0_LSB and WKT_VAL_WKT0_MSB registers
  - Wake timer1 value can be read through WKT_VAL_WKT1 register
  - When reading the wake timers, it is necessary to re-read it until a consistent value is read twice. This is because a minimal design is used to save power in the low-power mode and a single read may give a value at the point the count value is transitioning.
- Enable each timer with WKT_CTRL[WKT0_ENA] and WKT_CTRL[WKT1_ENA]
- Enable clocks for each timer by WKT_CTRL[WKT0_CLK_ENA] and WKT_CTRL[WKT1_CLK_ENA]
- The wakeup timers run on the OSC32K_CLK. The OSC32K_CLK is derived by either kFRO32K or kXTAL32K. It defaults to kFRO32K because not all hardware designs have the 32K crystal fitted. To move the OSC32K_CLK onto the kXTAL32K, add the following to your clock initialization:

  CLOCK_EnableClock(kCLOCK_Xtal32k); // Switch on the crystal

  CLOCK_AttachClk(kXTAL32K_to_OSC32K_CLK); // Drive the OSC32K_CLK from kXTAL32K_CLOCK

  This should give you accurate sleep timings. Note, an external 32.768 kHz crystal must be fitted for operation of the XTAL32K.
- Each Wake Timer can cause an interrupt to the processor which can be enabled through the WKT_INTENSET, WKT_INTENCLR and WKT_INTSTAT registers
  - Wake timer 0 uses interrupt bit 48
  - Wake timer 1 uses interrupt bit 49

  Specific interrupt handlers can be declared by overloading the weak function definitions for the interrupts. In this case WAKE_UP_TIMER0_IRQHandler and WAKE_UP_TIMER1_IRQHandler.
- Status flag in WKT_STAT to show if the timer has expired. Bit 0 for WKT0 timeout and Bit 1 for WKT1 timeout
  - These status bits can be cleared by writing a 1 to the required bit.
- Counter running/enabled status bit is available for each timer: WKT_STAT[WKT0_RUNNING], WKT_STAT[WKT1_RUNNING].
- The timers can cause wakeup from deep sleep and power down by configuring the STARTER0[TIMER0] and STARTER0[TIMER1] fields.
- The operation of one timer will not alter the behavior of the other timer.

11.1 Introduction

The device has several power domains and low-power modes, as described in previous chapters. To support software applications running on the device to use the power modes, a power control API is provided. This supports functionality to configure and enter low-power modes, identify the cause of a reset or a wake-up, to control the system voltages and to generate a software reset or an Arm system reset.

In power-down mode, it is possible to keep radio calibration values held in retention registers. This is a very low-power method for keeping the state of just the results of the radio calibration operations such that full radio recalibration is not required on power-up. The activation of this feature is controlled through the power control API.

See Chapter 8 “Power Management Controller and SLEEPCON” for full details of the power modes. When using the power management software, the user identifies the required functionality. The software then manages which power mode to use and the functional blocks that need to be active.

UM11138

Chapter 12: I/O Pin Configuration (IOCON)

Rev. 1.4 — June 2020 User manual

12.1 How to read this chapter

The IOCON block is included on all JN5189(T)/JN5188(T) parts. Registers for pins that are not available on a specific package are reserved.

JN5189(T)/JN5188(T) package is a HVQFN40 (6x6 mm).

Table 18. Available pins and configuration registers

Package	Total GPIOs	GPIO Port
40-pin	22	PIO0_0 to PIO0_21

Remark: Some functions, such as SCTimer/PWM inputs, frequency measure, and ADC triggers are not configured through IOCON. The connections for these functions are described in either the Chapter 13 “Input Multiplexing (INPUTMUX)” or the chapter for the specific function.

12.2 Features

The following electrical properties are configurable for standard port pins:

- Pull-up/pull-down resistor
- Open-drain mode
- Inverted function

Pins PIO0_10 through PIO0_11 (See Chapter 3 “Pin and Pad Descriptions” for details) are true open-drain pins that can be configured for different I^2C-bus speeds. Configuration options are different for these pins, as described in this chapter. The data sheet for this device gives full electrical details of all the PIO pins.

12.3 Basic configuration

Enable the clock to the IOCON by the SYSCON_AHBCLKCTRL0[IOCON] bit. Once the pins are configured, the IOCON clock can be disabled to conserve power.

12.4 General description

12.4.1 Pin configuration

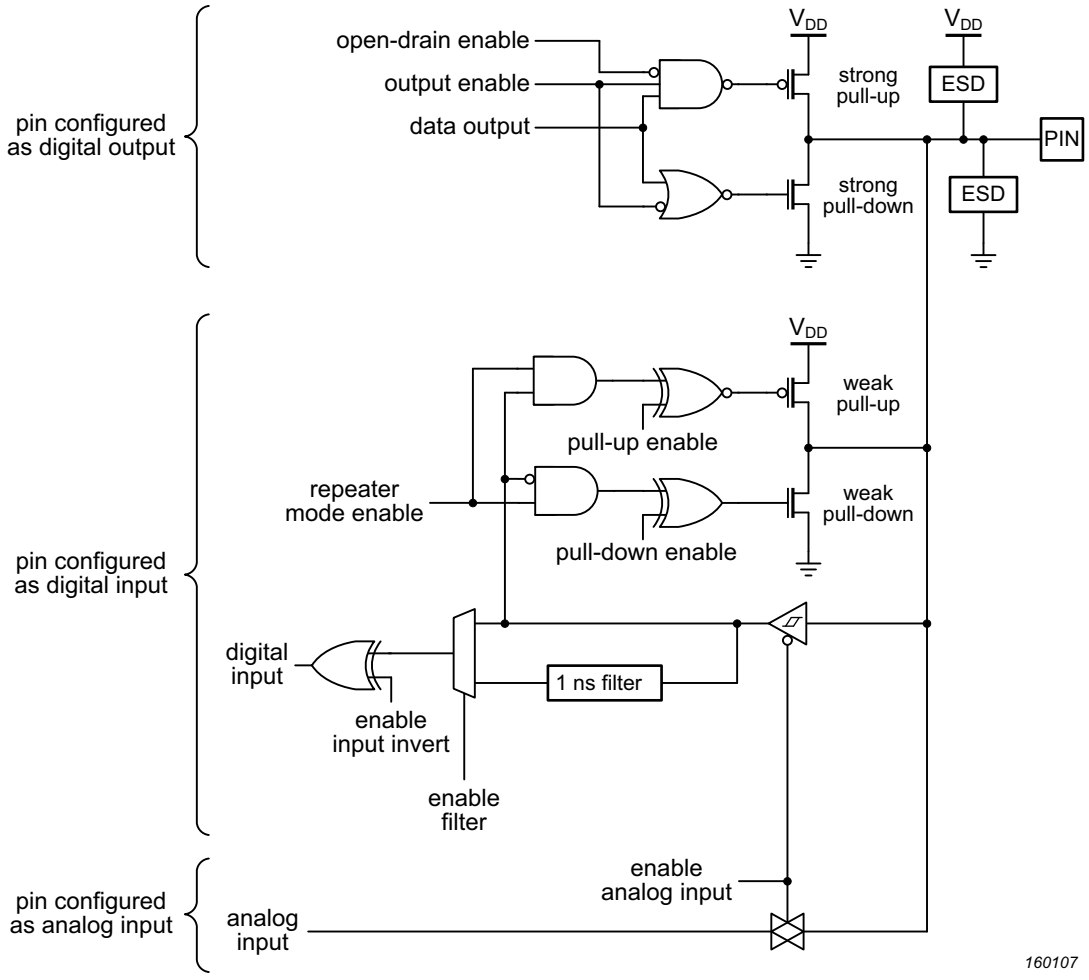


Fig 30. Pin configuration for standard digital IO cell

12.4.2 IOCON registers

The IOCON registers control the functions of device pins. Each GPIO pin has a dedicated control register to select its function and characteristics. Each pin has a unique set of functional capabilities. Not all pin characteristics are selectable on all pins. For instance, pins that have an I^2C function can be configured for different I^2C bus modes, while pins that have an analog alternate function can be selected to analog mode. The following sections describe specific characteristics of pins.

Multiple connections

Since a particular peripheral function may be allowed on more than one pin, it is possible to configure more than one pin to perform the same function. If a peripheral output function is configured on more than one pin, it will be routed to those pins. For correction operation, a peripheral input function must be configured to come from only one source.

12.4.2.1 Pin function

To optimize functionality in small packages, pins have several functions available via signal multiplexing.

The PIOn[FUNC] field can be set to GPIO (typically value 000b) or to a special function. For pins set to GPIO, the GPIO_DIR register determines whether the pin is configured as an input or output (see Section 14.5.4 “GPIO direction”). For any special function, the pin direction is controlled automatically depending on the function. The GPIO_DIR register has no effect for special functions.

Table 19. IOMUX functions

IO names	Value of FUNC field in PIOn registers								Register	Default at reset / during boot	Default internal pull-up / pull-down	ISP mode	Max. Freq
	000	001	010	011	100	101	110	111					
PIO0_0	GPIO0	USART0_SCK	USART1_TXD[2]	—	PWM0-PU[2]	SPI1_SCK	—	PDM0_DATA[1]	PIO0_0	GPIO0 (I)	pull-up	—	10 MHz
PIO0_1	GPIO1		USART1_RXD[1]	—	PWM1-PD[2]	SPI1_MISO	—	PDM0_CLK[2]	PIO0_1	GPIO1(I)	pull-down	—	10 MHz
PIO0_2	GPIO2	SPI0_SCK	USART0_RXD[1]	—	PWM2-PD[2]	SPI1_MOSI	ISO7816_RST	MCLK[1]	PIO0_2	GPIO2(I)	pull-down	—	10 MHz
PIO0_3	GPIO3	SPI0_MISO	USART0_TXD[2]	—	PWM3-PU[2]	SPI1_SSELN0	ISO7816_CLK	—	PIO0_3	GPIO3(I)	pull-up	—	10 MHz
PIO0_4	GPIO4/ISP_SEL	SPI0_MOSI	USART0_CTS[1]	—	PWM4-PU[2]	SPI1_SSELN1	ISO7816_IO	RFTX[2]	PIO0_4	GPIO4(I)	pull-up	ISP_SEL	10 MHz
PIO0_5	GPIO5/ISP_ENTRY	SPI0_SSELN	USART0_RTS[2]	—	SPI1_MISO	SPI1_SSELN2	—	RFRX[2]	PIO0_5	GPIO5(I)	pull-up	ISP_ENTRY	10 MHz
PIO0_6	GPIO6	SPI0_SCK	USART0_RTS[2]	CT32B1_MAT0[2]	PWM6-PD[2]	I2C1_SCL	USART1_TXD[2]	ADE[2]	PIO0_6	GPIO6(I)	pull-down	—	10 MHz
PIO0_7	GPIO7	SPI0_MISO	USART0_CTS[1]	CT32B1_MAT1[2]	PWM7-PD[2]	I2C1_SDA	USART1_RXD[1]	ADO[2]	PIO0_7	GPIO7(I)	pull-down	—	10 MHz
PIO0_8	GPIO8	SPI0_MOSI	USART0_TXD[2]	CT32B0_MAT0[2]	PWM8-PU[2]	ANA_COMP_OUT[2]	RFTX[2]	PDM1_DATA[1]	PIO0_8	GPIO8(I)[3]	pull-up	USART_ISP	10 MHz
PIO0_9	GPIO9	SPI0_SSELN	USART0_RXD[1]	CT32B1_CAP1[1]	PWM9-PU[2]	USART1_SCK	ADO[2]	PDM1_CLK[2]	PIO0_9	GPIO9(I)[4]	pull-up	USART_ISP	10 MHz
PIO0_10	GPIO10	CT32B0_CAP0[1]	USART1_TXD[2]	—	RFTX[2]	I2C0_SCL	SPI0_SCK	PDM0_DATA[1]	PIO0_10	GPIO10 (I)	external pull-up	—	10 MHz
PIO0_11	GPIO11	CT32B1_CAP0[1]	USART1_RXD[1]	—	RFRX[2]	I2C0_SDA	SPI0_MISO	PDM0_CLK[2]	PIO0_11	GPIO11 (I)	external pull-up	—	10 MHz
PIO0_12	GPIO12	IR_BLASTER[2]	SWCLK[1]	—	PWM0-PU[2]	I2C1_SCL	SPI0_MOSI	ANA_COMP_OUT[2]	PIO0_12	SWCLK	pull-up	—	10 MHz
PIO0_13	GPIO13	SPI1_SSELN2	SWDIO	—	PWM2-PU[2]	I2C1_SDA	SPI0_SSELN	—	PIO0_13	SWDIO	pull-up	—	10 MHz

Table 19. IOMUX functions *...continued*

IO names	Value of FUNC field in PIOn registers								Register	Default at reset / during boot	Default internal pull-up / pull-down	ISP mode	Max. Freq
	000	001	010	011	100	101	110	111					
PIO0_14/ADC	GPIO14	SPI1_SSELN1	USART0_SCK	CT32B0_CAP1[1]	PWM1-PU[2]	SWO[2]	MCLK[1]	RFTX[2]	PIO0_14	GPIO14(I)	pull-up	—	10 MHz
PIO0_15/ADC	GPIO15	SPI1_SCK	ANA_COMP_OUT[2]	—	PWM3-PU[2]	I2C0_SCL	PDM1_DATA[1]	RFRX[2]	PIO0_15	GPIO15(I)	pull-up	—	10 MHz
PIO0_16/ADC	GPIO16	SPI1_SSELN0	ISO7816_RST	—	PWM5-PU[2]	I2C0_SDA	PDM1_CLK[2]	SPIFI_CSN [set IO to high drive]	PIO0_16	GPIO16(I)	pull-up	—	33 MHz
PIO0_17/ADC	GPIO17	SPI1_MOSI	ISO7816_CLK	SWO[2]	PWM6-PD[2]	CLK_OUT[2]	—	SPIFI_IO3 [set IO to high drive]	PIO0_17	GPIO17(I)	pull-down	—	33 MHz
PIO0_18/ADC	GPIO18	SPI1_MISO	ISO7816_IO	CT32B0_MAT1[2]	PWM7-PD[2]	USART0_TXD[2]	—	SPIFI_CLK [set IO to high drive]	PIO0_18	GPIO18(I)	pull-down	—	33 MHz
PIO0_19/ADC	GPIO19	ADO[2]	USART1_RXD[1]	CLK_IN[1]	PWM4-PD[2]	USART0_RXD[1]	—	SPIFI_IO0 [set IO to high drive]	PIO0_19	GPIO19(I)	pull-down	—	33 MHz
PIO0_20/ACP	GPIO20	IR_BLASTER[2]	USART1_TXD[2]	—	PWM8-PD[2]	RFTX[2]	—	SPIFI_IO2 [set IO to high drive]	PIO0_20	GPIO20(I)	pull-down	—	33 MHz
PIO0_21/ACM	GPIO21	IR_BLASTER[2]	USART1_SCK	—	PWM9-PU[2]	RFRX[2]	SWO[2]	SPIFI_IO1 [set IO to high drive]	PIO0_21	GPIO21(I)	pull-up	—	33 MHz

[1] Input mode only.

[2] Output mode only.

[3] In ISP mode, it is configured to USART0_TXD.

[4] In ISP mode, it is configured to USART0_RXD.

12.4.2.2 Pin configuration

The IO cells for PIO0_0 to PIO0_21 can all be configured for digital operation. To support analog functionality, such as ADC inputs, they can also be configured in analog mode.

For analog IOs, you can use the following settings:

- Receiver can be disabled → DIGIMODE = 0
- Enable weak pull-up can be disabled → MODE[1] = 1 (Only for MFIO pads)
- Enable weak pull-down can be disabled → MODE[0] = 0 (Only for MFIO pads)

12.4.2.3 Pin mode

The PIOn[MODE] field selects the on-chip pull-up or pull-down resistors for each pin or select the repeater mode.

For standard IO cells (all PIOs except PIO10 or 11):

- The possible on-chip resistor configurations are pull-up enabled (0x0), pull-down enabled (0x3), or no pull-up/pull-down. The default value is pull-up or pull-down enabled; Table 19 “IOMUX functions” shows which IOs have pull-up and pull-down resistors by default.
- The repeater mode (0x1) enables the pull-up resistor if the pin is high and enables the pull-down resistor if the pin is low. This causes the pin to retain its last known state if it is configured as an input and is not driven externally. Such state retention is not applicable in the deep power-down mode. Repeater mode may typically be used to prevent a pin from floating (and potentially using significant power if it floats to an indeterminate state) if it is temporarily not driven.
- No pull-up or pull-down enabled is referred to as plain mode and is available by setting the PIOn[MODE] field to (0x2).

For IO cells supporting true I^2C (PIO10 & 11 only):

- 0x0: I^2C standard/fast and FP transmit mode (SDA and SCL) and I^2C high speed transmit mode (only SDAH)
- 0x1/0x3: GPIO mode (high speed if EHS is high, low speed if EHS is low: see SLEW0)
- 0x2: I^2C high speed transmit mode (SCLH)

12.4.2.4 Hysteresis

The input buffer for digital functions has built-in hysteresis. See the appropriate specific device data sheet for details.

12.4.2.5 Invert pin

This option is used to avoid including an external inverter on an input that is meant to be the opposite polarity of the external signal.

12.4.2.6 Analog/digital mode

When not in digital mode (PIOn[DIGIMODE] = 0b), a pin is in analog mode, the digital output buffer is disabled and analog pin functions are enabled. In digital mode (PIOn[DIGIMODE] = 1b), analog pin functions are disabled and digital pin functions are

enabled. This protects the analog input from voltages outside the range of the analog power supply and reference that may sometimes be present on digital pins, since they are typically 3.6 V tolerant. All pin types include this control, even if they do not support any analog functions.

In order to use a pin that has an ADC input option for that purpose, select GPIO (PIOn[FUNC] = 000b) and disable the digital pin function (PIOn[DIGIMODE] = 0b). The PIOn[MODE] field should also be set to 10b.

In analog mode, the PIOn[MODE] field should be “Plain Input' (10)”; the INVERT, FILTEROFF, and OD fields settings have no effect. For an unconnected pin that has an analog function, keep the DIGIMODE bit set to 1 (digital mode), and pull-up or pull-down mode selected in the MODE field.

12.4.2.7 Input filter

Some pins include a filter that can be selectively disabled by setting the FILTEROFF bit. The filter suppresses input pulses smaller than about 1 ns.

12.4.2.8 Output slew rate

The SLEW bits of digital outputs that do not need to switch state very quickly must be set as “slow stew” (see Table 5). This setting allows multiple outputs to switch simultaneously without noticeably degrading the power/ground distribution of the device, and has only a small effect on signal transition time. This is particularly important if analog accuracy is significant to the application. See the data sheet for more details.

12.4.2.9 I^2C modes

Pins that support I^2C with specialized IO cells (PIO[10] and PIO[11]) have different configuration bits and pin electronics (P0[23] through P0[28]) have additional configuration bits. These have multiple configurations to support I^2C variants. These are not hard-wired so that the pins can be more easily used for non-I^2C functions. See Chapter 25 “Inter-Integrated Circuit (I^2C)” for IO cell settings to support I^2C operation.

For non-I^2C operation, these special IO cells can be configured to operate as a standard GPIO cell by setting PIOn[EGP].

12.4.2.10 Open-Drain mode

When output is selected, either by selecting a special function in the FUNC field, or by selecting the GPIO function for a pin having a 1 in the related bit of that port’s GPIO_DIR register, a 1 in the OD bit selects open-drain operation, that is, a 1 disables the high-drive transistor. This option has no effect on the primary I^2C pins. Note that the properties of a pin in this simulated open-drain mode are different from those of a true open drain output.

12.5 Software control

IOCON functions are defined in the fsl_iocon.h as below:

IOCON_PinMuxSet

This function is used to associate a single pin to a peripheral.

IOCON_SetPinMuxing

This function is used to associate a multiple pins to a peripheral.

The following values are available:

- IOCON_FUNC0 to IOCON_FUNC7: Selects the pin function to a value between 0 and 7
- IOCON_ANALOG_EN: Enables analog function on the pin
- IOCON_DIGITAL_EN: Enables digital function on the pin
- IOCON_I2C_SLEW I2C: Slew Rate Control
- IOCON_INV_EN: Enables invert function on input
- IOCON_INPFILT_OFF: Disable input filter
- IOCON_INPFILT_ON: Enable input filter Pulses of less than 10ns are ignored
- IOCON_SLEW1_OFF: Disable Slew Rate Control
- IOCON_SLEW1_ON: Enable Slew Rate Control

  IOCON_OPENDRAIN_EN: Enables open-drain function

The following additional options are for pullup control:

- IOCON_MODE_PULLUP: Enable the Pullup
- IOCON_MODE_PULLDOWN: Enable the Pulldown

  IOCON_MODE_INACT: Disable both the pullup and pulldown

  IOCON_MODE_REPEATER: This mode allows a signal level to be ‘remembered’ after the drive is taken away

Setting UART0 RX/TX pins using IOCON_PinMuxSet

IOCON_PinMuxSet(IOCON, 0, 8, IOCON_MODE_INACT | IOCON_FUNC2 | IOCON_DIGITAL_EN);

IOCON_PinMuxSet(IOCON, 0, 9, IOCON_MODE_INACT | IOCON_FUNC2 | IOCON_DIGITAL_EN);

Setting UART0 RX/TX pins using IOCON_SetPinMuxing

iocon_group_t iopins[] ={

{0, 8, IOCON_MODE_INACT | IOCON_FUNC2 | IOCON_DIGITAL_EN},

{0, 9, IOCON_MODE_INACT | IOCON_FUNC2 | IOCON_DIGITAL_EN}};

IOCON_SetPinMuxing(IOCON,iopins,2);

13.1 How to read this chapter

Input multiplexing is present on all JN5189(T)/JN5188(T) devices.

13.2 Features

- Configure the inputs to the pin interrupt block and pattern match engine.
- Configure the inputs to the DMA triggers.
- Configure the inputs to the frequency measure function. This function is controlled by the ASYNC_SYSCON_FREQMECTRL register.

13.3 Basic configuration

Once set up, no clocks are needed for the input multiplexer to function. The system clock is needed only to write to or read from the INPUTMUX registers. Once the input multiplexer is configured, disable the clock to the INPUT MUX block by the SYSCON_AHBCLKCTRL0[MUX].

13.4 Pin description

The input multiplexer has no dedicated pins. However, all digital pins can be selected as inputs to the pin interrupts. Multiplexer inputs from external pins work independently of any other function assigned to the pin as long as no analog function is enabled.

Table 20. INPUT MUX pin description

Pins	Peripheral
Any existing pin on port 0	Pin interrupts 0 to 7
PIO0_4, PIO0_20, PIO0_16, PIO0_15	Frequency measure block

13.5 General description

The inputs to the DMA triggers, to the eight pin interrupts, and to the frequency measure block are multiplexed to multiple input sources. The sources can be external pins, interrupts, or output signals of other peripherals.

The input multiplexing makes it possible to design event-driven processes without CPU intervention by connecting peripherals like the ADC.

The DMA can use trigger input multiplexing to sequence DMA transactions without the use of interrupt service routines.

13.5.1 Pin interrupt input multiplexing

13.5.1.1 Pin interrupt select register

Each of these 5 bits selects one pin from PIO inputs as the source of a pin interrupt or as the input to the pattern match engine. To select a pin for any of the 8 pin interrupts or pattern match engine inputs, write the GPIO port pin number as 0 to 21 for pins PIO0_0 to PIO0_21 to the INTPIN field. For example, setting PINTSEL0[INTPIN] to 0x5 selects pin PIO0_5 as the source for the pin interrupt '0' signal, or the '0' input to the pattern match engine. To determine the GPIO port pin number for a given device package, see Section 3.3 “Pinout signaling descriptions”.

Each of the pin interrupts must be enabled in the NVIC before it becomes active. In JN5189(T)/JN5188(T), only pin interrupt 0 to 3 is connected to the NVIC.

To use the selected pins for pin interrupts or the pattern match engine, see Chapter 15 “Pin Interrupt and Pattern Match (PINT)”

Table 21. Pin interrupt select registers (PINTSELn (n = 0-7), offsets [0x0C0:0x0DC]) bit description

Bit	Symbol	Descriptions	Reset Value
4:0	INTPIN	Pin number select for pin interrupt or pattern match engine input.	0x1F
31:5	—	Reserved	—

13.5.1.2 Example

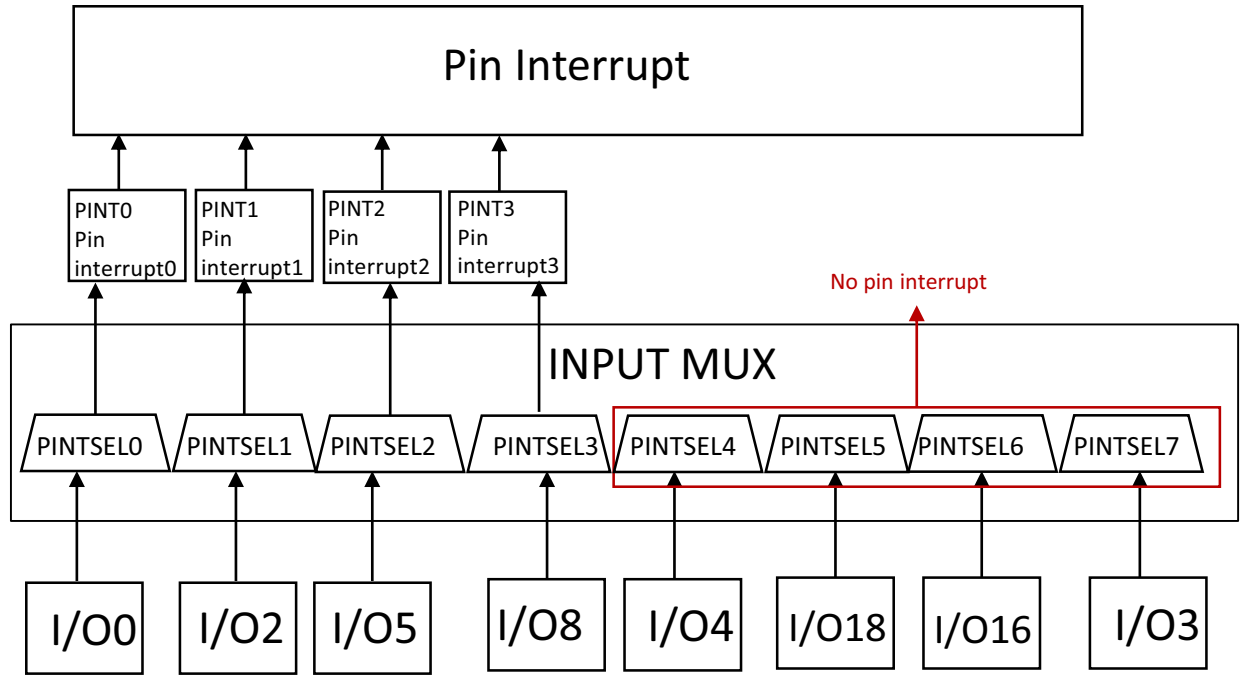


Fig 31. Input Mux and PINTSEL values

In the Figure 31, 8 PIOs are connected to the Input Mux block. Here are the corresponding values of each PINTSEL for this configuration:

- PIO0_0 is connected to PINTSEL0 → PINTSEL0->INTPIN[4:0] = 0x0
- PIO0_2 is connected to PINTSEL1 → PINTSEL1->INTPIN[4:0] = 0x2
- PIO0_5 is connected to PINTSEL2 → PINTSEL2->INTPIN[4:0] = 0x5
- PIO0_8 is connected to PINTSEL3 → PINTSEL3->INTPIN[4:0] = 0x8
- PIO0_4 is connected to PINTSEL4 → PINTSEL4->INTPIN[4:0] = 0x4
- PIO0_18 is connected to PINTSEL5 → PINTSEL5->INTPIN[4:0] = 0x12
- PIO0_16 is connected to PINTSEL6 → PINTSEL6->INTPIN[4:0] = 0x10
- PIO0_3 is connected to PINTSEL7 → PINTSEL7->INTPIN[4:0] = 0x3

13.5.2 DMA trigger input multiplexing

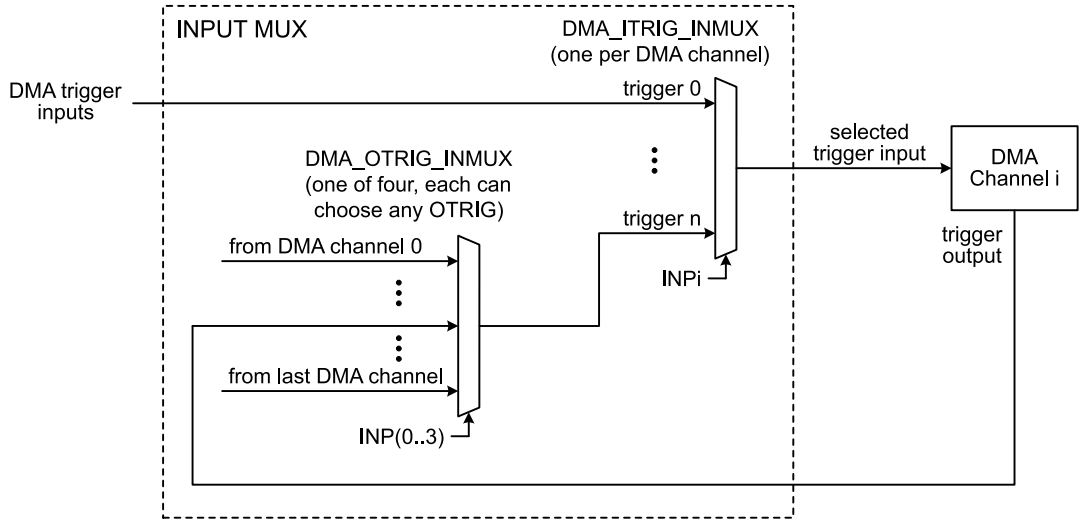


Fig 32. DMA trigger multiplexing

DMA operations can be triggered by on- or off-chip events. Each DMA channel can select one trigger input from 18 sources. Trigger sources include ADC interrupts, timer interrupts, pin interrupts, AES, HASH and DMA output triggers feedback.

13.5.2.1 DMA trigger input mux registers 0 to 18

With the DMA trigger input mux registers, one trigger input can be selected for each of the 19 DMA channels from the potential internal sources. By default, none of the triggers are selected.

Table 22. DMA trigger Input mux registers (DMA_ITRIG_INMUXn (n = 0-18), offsets [0xE0:0x128]) bit description

Bit	Symbol	Descriptions	Reset Value
4:0	INP	Trigger input number (decimal value) for DMA channel n (n = 0 to 17). • 0 = ADC0 Sequence A interrupt • 1 = Reserved • 2 = Timer CT32B0 Match 0 • 3 = Timer CT32B0 Match 1; • 4 = Timer CT32B1 Match 0; • 5 = Timer CT32B1 Match 1; • 6 = Pin interrupt 0; • 7 = Pin interrupt 1; • 8 = Pin interrupt 2; • 9 = Pin interrupt 3; • 10 = AES RX; • 11 = AES TX; • 12 = Hash RX; • 13 = Hash TX; • 14 = DMA output trigger mux 0; • 15 = DMA output trigger mux 1; • 16 = DMA output trigger mux 2; • 17 = DMA output trigger mux 3;	0x1F
31:5	—	Reserved	—

13.5.2.2 DMA output trigger selection

In Section 13.5.2.1, it was shown that the DMA trigger for each DMA channel can be selected from 18 different triggers. Four of these trigger sources are DMA output triggers so that it is possible to chain DMA activities. For each of these four DMA output trigger sources, there is a mux to select which of the 19 DMA channels will be selected. The four muxes are controlled by the DMA_OTRIG_INMUX registers.

Table 23. DMA output trigger selection to become DMA trigger (DMA_OTRIG_INMUXn (n = 0-3), offsets [0x160:0x16C]) bit description

Bit	Symbol	Descriptions	Reset Value
4:0	INP	DMA trigger output number (decimal value) for DMA channel n (n = 0 to 18).	0x1F
31:5	—	Reserved	0x0

14.1 How to read this chapter

GPIO registers support up to 22 pins (see Table 18).

14.2 Basic configuration

For the GPIO registers, enable the clock to the GPIO block in the SYSCON_AHBCLKCTRL0 register.

14.3 Features

- GPIO pins can be configured as input or output by software.
- All GPIO pins default to inputs with interrupt disabled at reset.
- Pin registers allow pins to be sensed and set individually.
- Direction (input/output) can be set and cleared individually.

14.4 General description

The GPIO pins can be used in several ways to set pins as inputs or outputs and use the inputs as combinations of level and edge sensitive interrupts.

The GPIOs can be used as external interrupts together with the pin interrupt and group interrupt blocks, see Chapter 15 and Chapter 16.

The GPIO registers configure each GPIO pin as input or output, and read the state of each pin if the pin is configured as input, or set the state of each pin if the pin is configured as output.

14.5 Functional description

14.5.1 Reading pin state

Software can read the state of all GPIO pins except those selected for analog input or output in the “I/O Configuration” logic. A pin does not have to be selected for GPIO in “I/O Configuration” in order to read its state; it can be assigned to another digital function. There are four ways to read pin state:

- The state of a single pin can be read with 7 high-order zeros from a Byte Pin register.
- The state of a single pin can be read in all bits of a byte, halfword, or word from a Word Pin register.
- The state of multiple pins can be read from the PIN register.
- The state of a selected subset of the pins can be read from a Masked Pin (MPIN) register. Pins having a 1 in the Mask register will read as 0 from its MPIN register.

14.5.2 GPIO output

Each GPIO pin has an output bit in the GPIO block. These output bits are the targets of write operations to the pins. Two conditions must be met in order for a pin's output bit to be driven onto the pin:

1. The pin must be selected for GPIO operation via IOCON (this is the default configuration except for PIOs 8, 9, 12 and 13), and
2. The pin must be selected for output by a 1 in its port's DIR register.

If either or both of these conditions is (are) not met, writing to the pin has no effect.

There are seven ways to change GPIO output bits:

- Writing to the Byte Pin register loads the output bit from the least significant bit.
- Writing to the Word Pin register loads the output bit with the OR of all of the bits written. (This feature follows the definition of truth of a multi-bit value in programming languages.)
- Writing to the PIN register loads the output bits of all the pins written to.
- Writing to the MPIN register loads the output bits of pins identified by zeros in corresponding positions of the MASK register.
- Writing ones to the SET register sets output bits.
- Writing ones to the CLR register clears output bits.
- Writing ones to the NOT register toggles/complements/inverts output bits.

The state of the output bits can be read from its SET register. Reading any of the registers described in 14.5.1 returns the state of pins, regardless of their direction or alternate functions.

14.5.3 Masked I/O

The MASK register defines which of its pins should be accessible in its MPIN register. Zeroes in MASK enable the corresponding pins to be read from and written to MPIN. Ones in MASK force a pin to read as 0 and its output bit to be unaffected by writes to MPIN. When a port's MASK register contains all zeros, its PIN and MPIN registers operate identically for reading and writing.

Applications in which interrupts can result in Masked GPIO operation, or in task switching among tasks that do Masked GPIO operation, must treat code that uses the Mask register as a protected/restricted region. This can be done by interrupt disabling or by using a semaphore.

The simpler way to protect a block of code that uses a MASK register is to disable interrupts before setting the MASK register, and re-enable them after the last operation that uses the MPIN or MASK register.

More efficiently, software can dedicate a semaphore to the MASK register, and set/capture the semaphore controlling exclusive use of the MASK register before setting the MASK register, and release the semaphore after the last operation that uses the MPIN or MASK registers.

14.5.4 GPIO direction

Each GPIO pin can be configured as input or output using the DIR register. The direction of individual pins can be set, cleared, or toggled using the DIRSET, DIRCLR, and DIRNOT registers. When configuring a single GPIO direction, it is faster to use these registers.

14.5.5 Recommended practices

The following lists some recommended uses for using the GPIO registers:

- For initial setup after Reset or re-initialization, write the DIR and PIN registers.
- To change the state of one pin, write a Byte Pin or Word Pin register.
- To change the state of multiple pins at a time, write the SET and/or CLR registers.
- To change the state of multiple pins in a tightly controlled environment like a software state machine, consider using the NOT register. This can require less write operations than SET and CLR.
- To read the state of one pin, read a Byte PIN or Word PIN register.
- To make a decision based on multiple pins, read and mask the PIN register.

14.6 Software control

To support the use of the GPIOs the following functions are provided within the SDK:

- GPIO_WritePinOutput
- GPIO_ReadPinInput
- GPIO_SetPinsOutput
- GPIO_ClearPinsOutput
- GPIO_TogglePinsOutput
- GPIO_ReadPinsInput
- GPIO_SetPortMask
- GPIO_WriteMPort
- GPIO_ReadMPort

15.1 How to read this chapter

The pin interrupt generator and the pattern match engine are available on all JN5189(T)/JN5188(T) parts.

15.2 Features

This block has two mutually exclusive features:

- Pin interrupts
  - Up to four pins can be selected from all GPIO pins as edge- or level-sensitive interrupt requests. Each request creates a separate interrupt in the NVIC.
  - Edge-sensitive interrupt pins can interrupt on rising or falling edges or both.
  - Level-sensitive interrupt pins can be HIGH- or LOW-active.
- Pattern match engine
  - Up to 8 pins can be selected from all digital pins to contribute to a boolean expression. The boolean expression consists of specified levels and/or transitions on various combinations of these pins.
  - Each bit slice minterm (product term) comprising the specified boolean expression can generate its own, dedicated interrupt request.
  - Any occurrence of a pattern match can be programmed to also generate a receive event, RXEV, notification to the CPU.
  - Pattern match can be used, in conjunction with software, to create complex state machines based on pin inputs.

15.3 Basic configuration

- Pin interrupts:
  - Select up to four external interrupt pins from all digital port pins in the Input Mux block (using INPUTMUX_PINTSEL register). The pin selection process is the same for pin interrupts and the pattern match engine. The two features are mutually exclusive.
  - Enable the clock to the pin interrupt register block in the SYSCON_AHBCLKCTRL0 register
  - In order to use the pin interrupts to wake up the part from deep-sleep mode, enable the pin interrupt wake-up feature in the SYSCON_STARTER0 register for pin interrupt 0 through 3.

    Remark: The PINT block can not cause a wake-up from Power-down and deep power-down states. However, individual IOs can be configured to cause wake-ups from these states. See Section 5.3.

– Pin interrupts 0 to 3 are assigned to an interrupt in the NVIC (Table 17 “Connection of interrupt sources to the NVIC”). Interrupt 4 to 7 are not supported in this processor. (see Chapter 9 “Nested Vectored Interrupt Controller (NVIC)”).Section 5.3

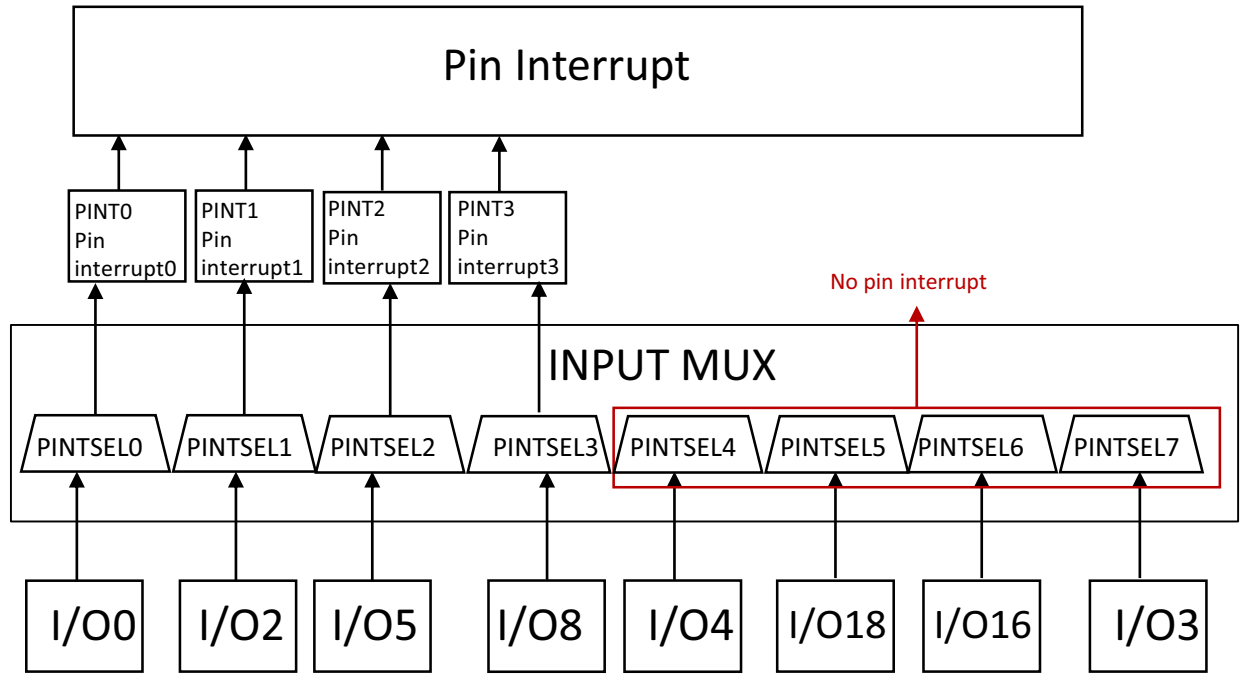


Fig 33. Pin Interrupts

- Pattern match engine:
  - Select up to eight external pins from all digital port pins in the Input mux block (See Pin interrupt select registers for details). The pin selection process is the same for pin interrupts and the pattern match engine. The two features are mutually exclusive.
  - Enable the clock to the pin interrupt register block in the SYSCON_AHBCLKCTRL0 register.
  - Only the first 4 bit slices of the pattern match engine are assigned to an interrupt in the NVIC (Table 17 “Connection of interrupt sources to the NVIC”)

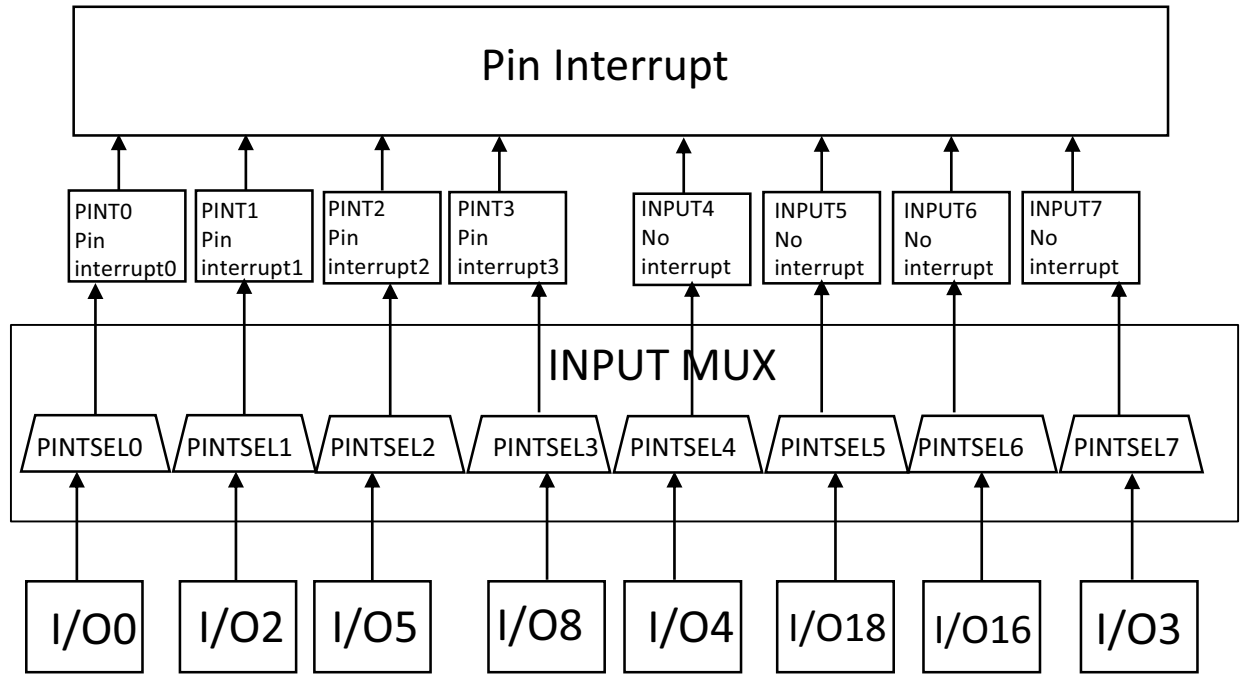


Fig 34. Pattern Match Engine

15.3.1 Configure pins as pin interrupts or as inputs to the pattern match engine

Follow these steps to configure pins as pin interrupts:

1. Determine which digital pins are required for the pin interrupt or pattern match function.
2. For each pin interrupt, program the GPIO port pin number into one of the eight INPUTMUX_PINTSEL register in the Input mux block.

   Remark: The port pin number serves to identify the pin to the INPUTMUX_PINTSEL register. Any function, including GPIO, can be assigned to this pin via IOCON.
3. Enable each pin interrupt in the NVIC.

Once the pin interrupts or pattern match inputs are configured, the pin interrupt detection levels or the pattern match boolean expression can set up.

See Section 13.5.1.1 “Pin interrupt select register” in the Input mux block for the INPUTMUX_PINTSEL registers.

Remark: The inputs to the Pin interrupt select registers bypass the IOCON function selection. They do not have to be selected as GPIO in IOCON. Make sure that no analog function is selected on pins that are input to the pin interrupts.

15.4 Pin description

The inputs to the pin interrupt and pattern match engine are determined by the INPUTMUX_PINTSEL. See Section 13.5.1.1 “Pin interrupt select register”.

15.5 General description

Pins with configurable functions can serve as external interrupts or inputs to the pattern match engine. Up to eight pins can be configured using the INPUTMUX_PINTSEL registers for these features.

15.5.1 Pin interrupts

From all available GPIO pins, up to four pins can be selected in the system control block to serve as external interrupt pins (see Chapter 13 “Input Multiplexing (INPUTMUX)” and Section 13.5.1.2 “Example”).

15.5.2 Pattern match engine

The pattern match feature allows complex boolean expressions to be constructed from the same set of eight GPIO pins that were selected for the GPIO pin interrupts. Each term in the boolean expression is implemented as one slice of the pattern match engine. A slice consists of an input selector and a detect logic that monitors the selected input continuously and creates a HIGH output if the input qualifies as detected, that is as true. Several terms can be combined to a minterm and a pin interrupt is asserted when the minterm evaluates as true.

The detect logic of each slice can detect the following events on the selected input:

- Edge with memory (sticky): A rising edge, a falling edge, or a rising or falling edge that is detected at any time after the edge-detection mechanism has been cleared. The input qualifies as detected (the detect logic output remains HIGH) until the pattern match engine detect logic is cleared again.
- Event (non-sticky): Every time an edge (rising or falling) is detected, the detect logic output for this pin goes HIGH. This bit is cleared after one clock cycle, and the detect logic can detect another edge,
- Level: A HIGH or LOW level on the selected input.

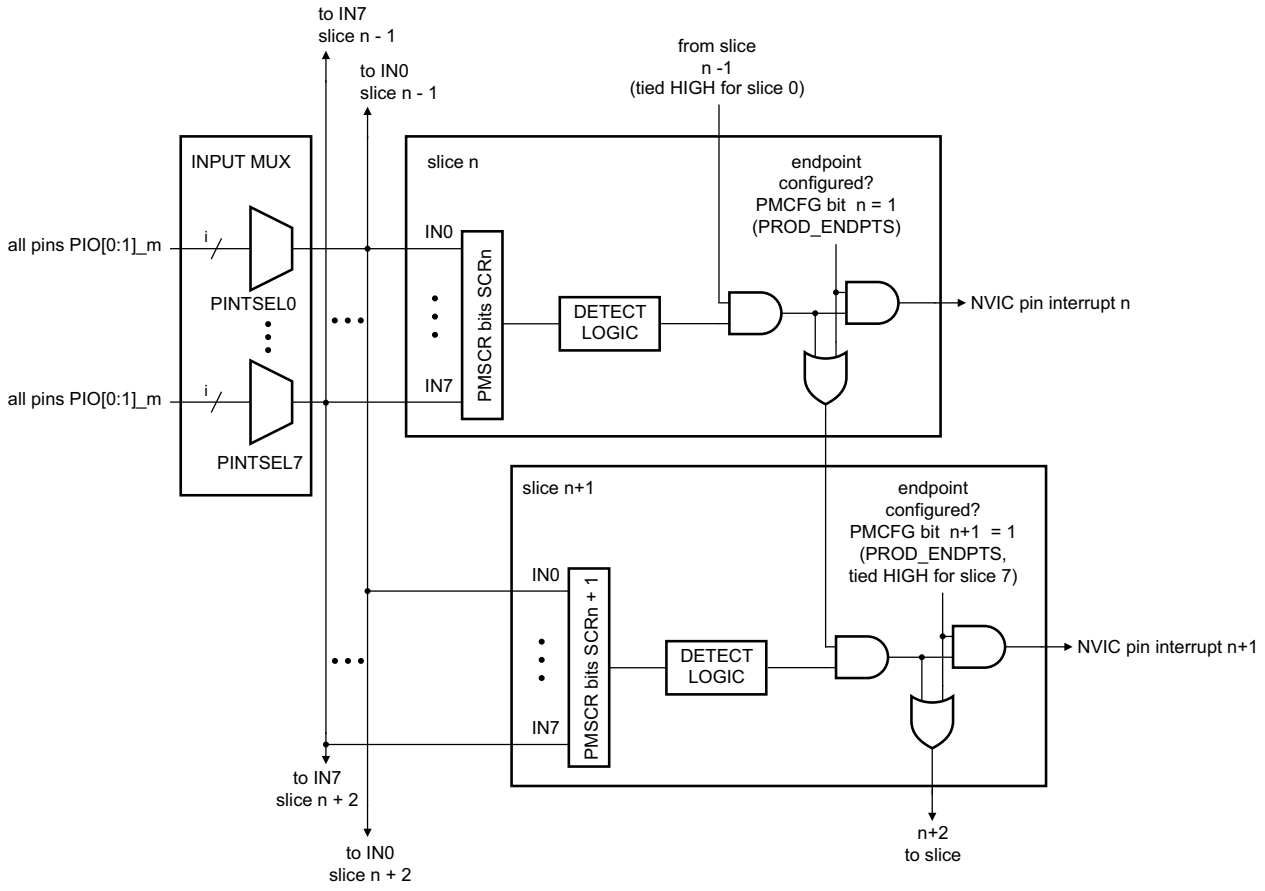
Figure 36 shows the details of the edge detection logic for each slice.

Sticky events can be combined with non-sticky events to create a pin interrupt whenever a rising or falling edge occurs after a qualifying edge event.

A time window can be created during which rising or falling edges can create a pin interrupt by combining a level detect with an event detect. See Section 15.6.3 for details.

The connections between the pins and the pattern match engine are shown in Figure 35. All pins that are inputs to the pattern match engine are selected in the Syscon block and can be GPIO port pins or other pin function depending on the IOCON configuration.

Remark: The pattern match feature requires clocks in order to operate, and can thus not generate an interrupt or wake up the device during reduced power modes below sleep mode.



See Figure 36 for the detect logic block.

Fig 35. Pattern match engine connections

The pattern match logic continuously monitors the eight inputs and generates interrupts when any one or more minterms (product terms) of the specified boolean expression is matched. A separate interrupt request is generated for the first 4 minterms (no separate interrupt for the last four minterms).

In addition, the pattern match module can be enabled to generate a Receive Event (RXEV) output to the Arm core when the entire boolean expression is true (i.e. when any minterm is matched).

The pattern match function utilizes the same eight interrupt request lines as the pin interrupts so these two features are mutually exclusive as far as interrupt generation is concerned. A control bit is provided to select whether interrupt requests are generated in response to the standard pin interrupts or to pattern matches. Note that, if the pin interrupts are selected, the RXEV request to the CPU can still be enabled for pattern matches.

Remark: Pattern matching cannot be used to wake the part up from deep-sleep mode. Pin interrupts must be selected in order to use the GPIO for wake-up.

The pattern match module is constructed of eight bit-slice elements. Each bit slice is programmed to represent one component of one minterm (product term) within the boolean expression. The interrupt request associated with the last bit slice for a particular minterm will be asserted whenever that minterm is matched (only for the 4 first minterms). (See bit slice drawing Figure 36).

The pattern match capability can be used to create complex software state machines. Each minterm (and its corresponding individual interrupt, for minterms 0 to 3) represents a different transition event to a new state. Software can then establish the new set of conditions (that is a new boolean expression) that will cause a transition out of the current state.

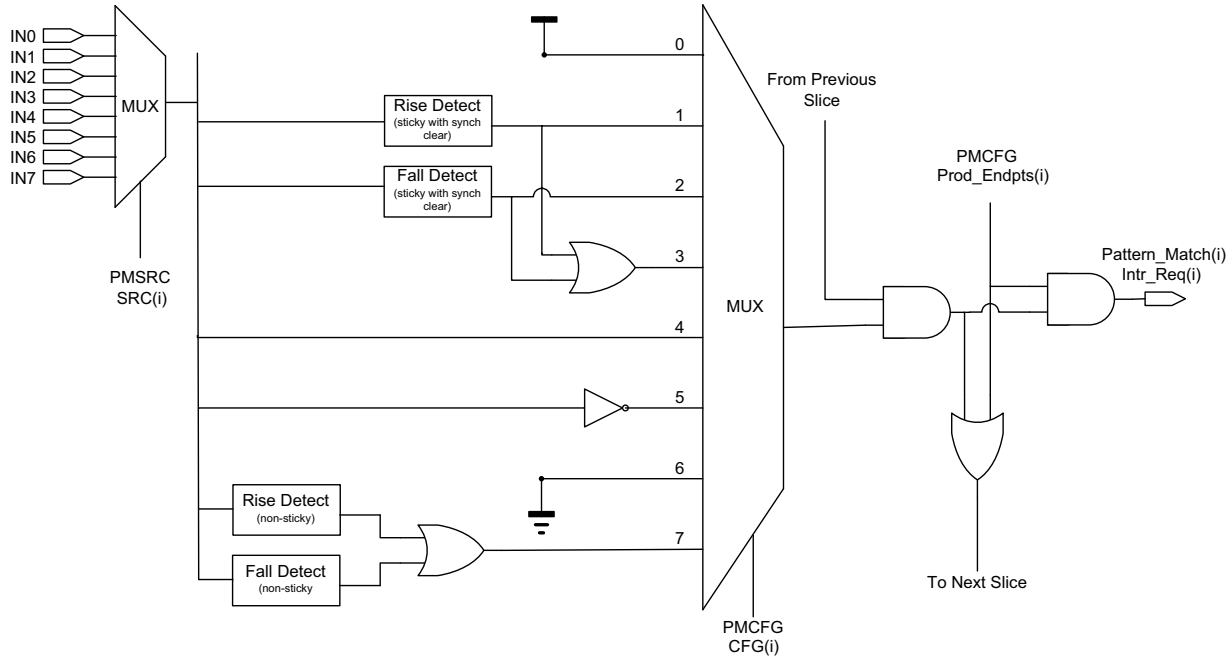


Fig 36. Pattern match bit slice with detect logic

15.5.2.1 Example

Assume the expression: (IN0)~(IN1)(IN3)^ + (IN1)(IN2) + (IN0)~(IN3)~(IN4) is specified through the registers PMSRC and PMCFG. Each term in the boolean expression, (IN0), ~(IN1), (IN3)^, etc., represents one bit slice of the pattern match engine.

- In the first minterm (IN0)~(IN1)(IN3)^, bit slice 0 monitors for a high-level on input (IN0), bit slice 1 monitors for a low level on input (IN1) and bit slice 2 monitors for a rising-edge on input (IN3). If this combination is detected, that is if all three terms are true, the interrupt associated with bit slice 2 will be asserted.
- In the second minterm (IN1)(IN2), bit slice 3 monitors input (IN1) for a high level, bit slice 4 monitors input (IN2) for a high level. If this combination is detected, the interrupt associated with bit slice 4 will be asserted.
- In the third minterm (IN0)~(IN3)~(IN4), bit slice 5 monitors input (IN0) for a high level, bit slice 6 monitors input (IN3) for a low level, and bit slice 7 monitors input (IN4) for a low level. If this combination is detected, the interrupt associated with bit slice 7 will be asserted.

- The ORed result of all three minterms asserts the RXEV request to the CPU. That is, if any of the three terms are true, the output is asserted.

Related links: Section 15.6.2

15.6 Functional description

15.6.1 Pin interrupts

In this interrupt facility, up to 4 pins are identified as interrupt sources by the Pin Interrupt Select registers (INPUTMUX_PINTSEL[2:0]). All registers in the pin interrupt block contain 8 bits, corresponding to the pins called out by the PINTSEL0-7 registers. The ISEL register defines whether each interrupt pin is edge- or level-sensitive. The RISE and FALL registers detect edges on each interrupt pin, and can be written to clear (and set) edge detection. The IST register indicates whether each interrupt pin is currently requesting an interrupt, and this register can also be written to clear interrupts.

The other pin interrupt registers play different roles for edge-sensitive and level-sensitive pins, as described in Table 24.

Table 24. Pin interrupt registers for edge- and level-sensitive pins

Name	Edge-sensitive function	Level-sensitive function
IENR	Enables rising-edge interrupts.	Enables level interrupts.
SIENR	Write to enable rising-edge interrupts.	Write to enable level interrupts.
CIENR	Write to disable rising-edge interrupts.	Write to disable level interrupts.
IENF	Enables falling-edge interrupts.	Selects active level.
SIENF	Write to enable falling-edge interrupts.	Write to select high-active.
CIENF	Write to disable falling-edge interrupts.	Write to select low-active.

15.6.2 Pattern Match engine example

Suppose the desired boolean pattern to be matched is:
(IN1) + (IN1 * IN2) + (~IN2 * ~IN3 * IN6fe) + (IN5 * IN7ev)

with:

IN6fe = (sticky) falling-edge on input 6

IN7ev = (non-sticky) event (rising or falling edge) on input 7

Each individual term in the expression shown above is controlled by one bit-slice. To specify this expression, program the pattern match bit slice source and configuration register fields as follows:

- PMSRC register:
  - Since bit slice 5 will be used to detect a sticky event on input 6, a 1 can be written to the SRC5 bits to clear any pre-existing edge detects on bit slice 5.
  - SRC0: 001 - select input 1 for bit slice 0
  - SRC1: 001 - select input 1 for bit slice 1
  - SRC2: 010 - select input 2 for bit slice 2
  - SRC3: 010 - select input 2 for bit slice 3

- SRC4: 011 - select input 3 for bit slice 4
- SRC5: 110 - select input 6 for bit slice 5
- SRC6: 101 - select input 5 for bit slice 6
- SRC7: 111 - select input 7 for bit slice 7

- PMCFG register:
  - PROD_ENDPTS0 = 1
  - PROD_ENDPTS2 = 1
  - PROD_ENDPTS5 = 1
  - All other slices are not product term endpoints and their PROD_ENDPTS bits are 0. Slice 7 is always a product term endpoint and does not have a register bit associated with it.
  - = 0100101 - bit slices 0, 2, 5, and 7 are product-term endpoints. (Bit slice 7 is an endpoint by default - no associated register bit).
  - CFG0: 000 - high level on the selected input (input 1) for bit slice 0
  - CFG1: 000 - high level on the selected input (input 1) for bit slice 1
  - CFG2: 000 - high level on the selected input (input 2) for bit slice 2
  - CFG3: 101 - low level on the selected input (input 2) for bit slice 3
  - CFG4: 101 - low level on the selected input (input 3) for bit slice 4
  - CFG5: 010 - (sticky) falling edge on the selected input (input 6) for bit slice 5
  - CFG6: 000 - high level on the selected input (input 5) for bit slice 6
  - CFG7: 111 - event (any edge, non-sticky) on the selected input (input 7) for bit slice 7
- PMCTRL register:
  - Bit0: Setting this bit will select pattern matches to generate the pin interrupts in place of the normal pin interrupt mechanism.

    For this example, pin interrupt 0 will be asserted when a match is detected on the first product term (which, in this case, is just a high level on input 1).

    Pin interrupt 2 will be asserted in response to a match on the second product term.

    Pin interrupt 5 will be asserted when there is a match on the third product term.

    Pin interrupt 7 will be asserted on a match on the last term.
  - Bit1: Setting this bit will cause the RxEv signal to the CPU to be asserted whenever a match occurs on ANY of the product terms in the expression. Otherwise, the RXEV line will not be used.
  - Bit31:24: At any given time, bits 0, 2, 5 and/or 7 may be high if the corresponding product terms are currently matching.
  - The remaining bits will always be low.

15.6.3 Pattern match engine edge detect examples

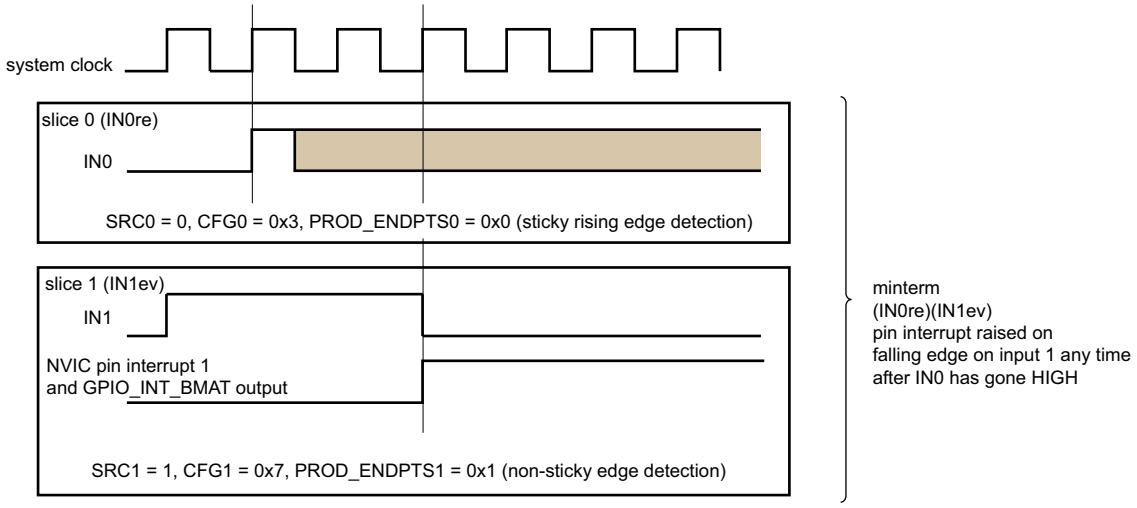


Figure shows pattern match functionality only and accurate timing is not implied. Inputs (INn) are shown synchronized to the system clock for simplicity.

Fig 37. Pattern match engine examples: sticky edge detect

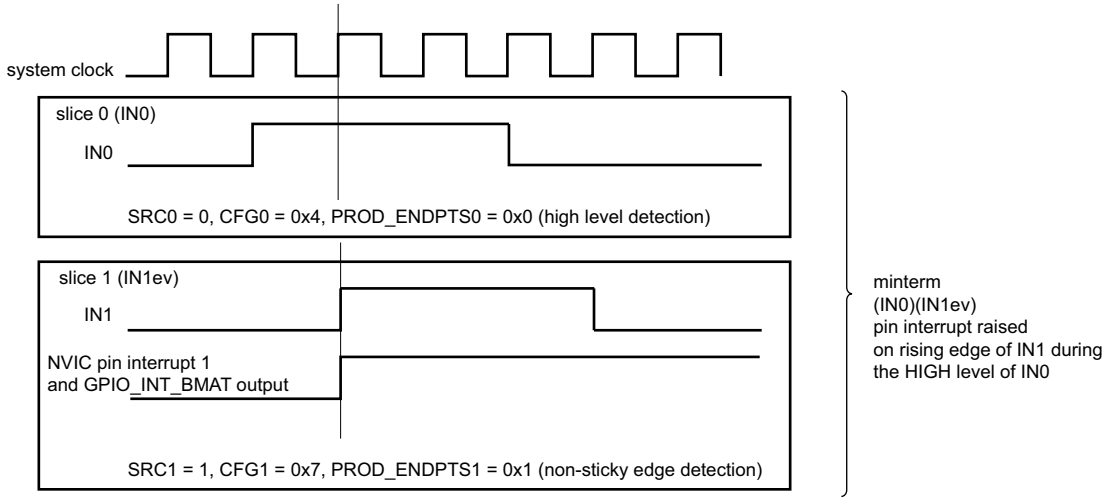


Figure shows pattern match functionality only and accurate timing is not implied. Inputs (INn) are shown synchronized to the system clock for simplicity.

Fig 38. Pattern match engine examples: Windowed non-sticky edge detect evaluates as true

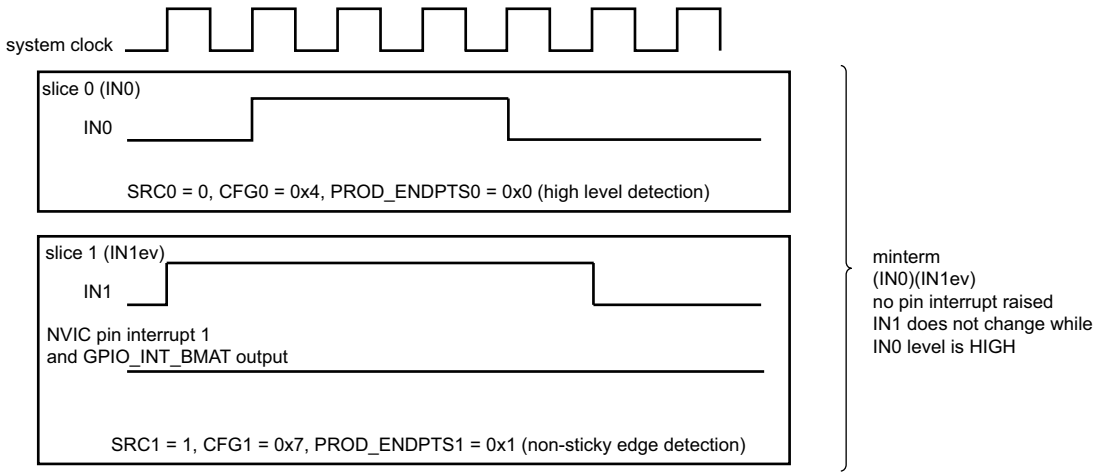


Figure shows pattern match functionality only and accurate timing is not implied. Inputs (INn) are shown synchronized to the system clock for simplicity.

Fig 39. Pattern match engine examples: Windowed non-sticky edge detect evaluates as false

15.7 Software control

To use the functionality of the PINT module, it is recommended to use software functions from within fsl_pint.c.

16.1 Features

- The inputs from any number of digital pins can be enabled to contribute to a combined group interrupt.
- The polarity of each input enabled for the group interrupt can be configured HIGH or LOW.
- Enabled inputs can be logically combined through an OR or AND operation.
- The group interrupt can wake up the part from sleep or deep-sleep modes.

16.2 Basic configuration

For the group interrupt feature, enable the clock to the GINT register interfaces in the SYSCON_AHBCLKCTRL0 register. The group interrupt wake-up feature is enabled in the SYSCON_STARTER0 register for GINT (see GINT register descriptions). The interrupt must also be enabled in the NVIC (see Table 17 “Connection of interrupt sources to the NVIC”).

The pins can be configured as GPIO pins through IOCON, but they don't have to be; if they are under control of a functional block then they must be configured as an input due to the blocks functionality. The GINT block reads the input from the pin bypassing IOCON multiplexing. Make sure that no analog function is selected on pins that are input to the group interrupt. Selecting an analog function in IOCON disables the digital portion of the pin and the digital signal is tied to 0.

16.3 General description

For each port/pin connected to the GPIO Grouped Interrupt block, the GPIO grouped interrupt registers determine which pins are enabled to generate interrupts and what the active polarities of each of those inputs are.

The GPIO grouped interrupt registers also select whether the interrupt output will be level or edge triggered and whether it will be based on the OR or the AND of all of the enabled inputs.

When the designated pattern is detected on the selected input pins, the GPIO grouped interrupt block generates an interrupt. If the part is in sleep or deep-sleep state, it first asynchronously wakes the part up prior to asserting the interrupt request. When using edge triggered interrupts, the interrupt request line can be cleared by writing a one to the interrupt status bit in the control register.

16.3.1 Contrast between the GPIO Pattern Match Interrupt (PINT) and the GPIO Group Interrupt (GINT) features

Although these two features both enable interrupt generation based on patterns of GPIO inputs, they are different. The group interrupt allows a specified subset of the available inputs to generate one single interrupt request, either when any one of the enabled inputs

is active or when all of the enabled inputs are active. It cannot, however, respond to AND/OR Boolean expressions. The pattern matching function (PINT) can be used to specify complex AND/OR Boolean expressions and can generate multiple, separate interrupt requests for each AND term within the expression.

16.4 Functional description

Any subset of the GPIO pins can be selected to contribute to a common group interrupt (GINT) and can be enabled to wake the part up from sleep or deep-sleep mode.

An interrupt can be requested based on any selected subset of pins. The pins that contribute to the interrupt are selected by 1s in the port Enable register (PORT_ENA0), and an pin polarity can be selected for each pin in the port Polarity register (PORT_POL0). The level on each pin is exclusive-ORed with its polarity bit, and the result is ANDed with its enable bit. The results for all enabled bits are used to create an array that is fed into the AND/OR function. If CTRL[COMB] control bit is set then the block is operating in AND mode; if all bits in the array are set then the group interrupt condition is met. If CTRL[COMB] control bit is clear then the block is operating in OR mode; if any bit in the array is set then the group interrupt condition is met.

If the block is operating in edge triggered mode (CTRL[TRIG]=0) then the interrupt flag (CTRL[INT]) will be set at the instant that the group interrupt condition is set. It can be cleared by writing a 1 to this bit. Alternatively, the block can operate in level triggered mode (CTRL[TRIG]=1), then the interrupt flag reflects whether the group interrupt condition is currently met or not. It can be observed in the status bit (CTRL[INT]) but will only be cleared when the group interrupt condition is no longer met.

The raw interrupt request from the group interrupt is sent to the NVIC, which can be programmed to treat it as level- or edge-sensitive, or it can be edge-detected by the wake-up interrupt logic.

16.5 Example

16.5.1 Example 1

The following example shows some example waveforms input into the GINT block. For the GINT configuration specified here it is shown when interrupts would be generated.

The GINT block is configured to generate an interrupt for "(IO0 asserted) OR (IO2 asserted) OR (IO4 asserted)". This is achieved with the configuration:

- GINT->CTRL = 0x0 - OrComb/EdgeTrigInt
- GINT->POL_PIO = 0x15 - High pol on IO0, IO2 and IO4
- GINT->ENA_PIO = 0x15 - Enable int on IO0, IO2 and IO4

The following figure shows example inputs and when interrupts are generated.

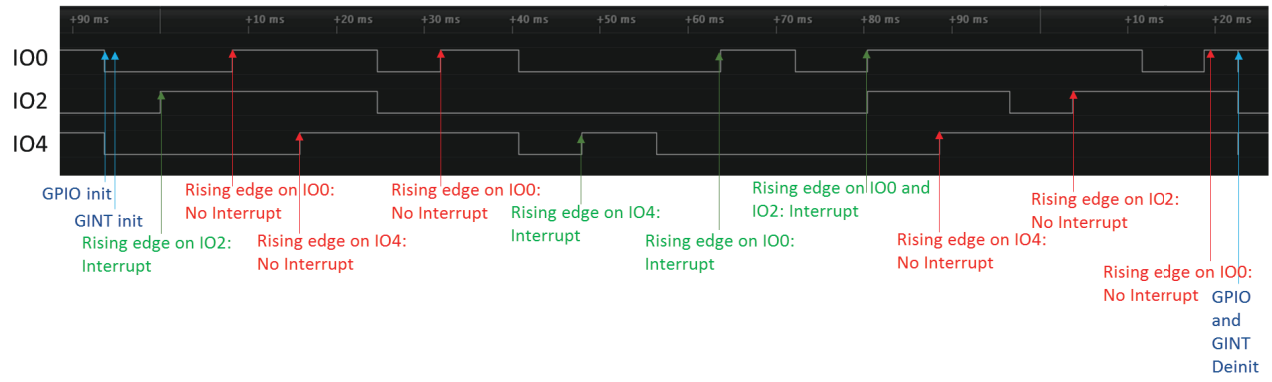


Fig 40. GINT Example for "(IO0 asserted) OR (IO2 asserted) OR (IO4 asserted) "

16.5.2 Example 2

The following example shows some example waveforms input into the GINT block. For the GINT configuration specified here it is shown when interrupts would be generated.

The GINT block is configured to generate an interrupt for "(IO0 asserted) AND (IO2 asserted)". This is achieved with the configuration:

- GINT->CTRL = 0x2 - AndComb/EdgeTrigInt
- GINT->POL_PIO = 0x5 - High pol on IO0 and IO2
- GINT->ENA_PIO = 0x5 - Enable int on IO0 and IO2

The following figure shows example inputs and when interrupts are generated by the GINT block in this configuration.

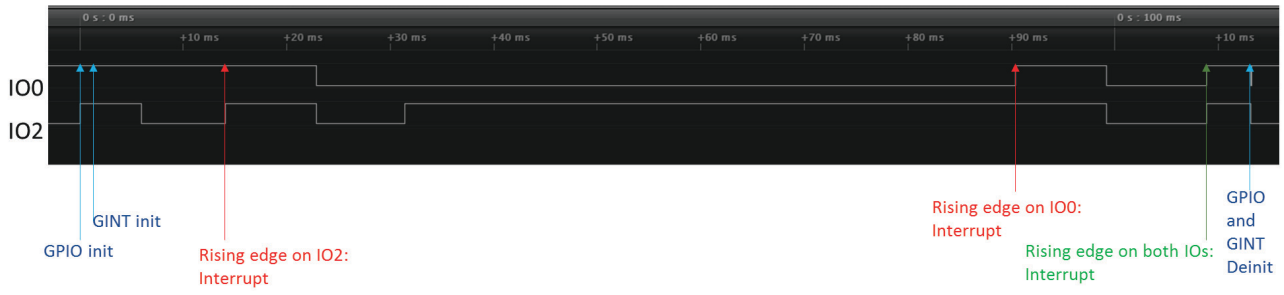


Fig 41. GINT example for "(IO0 asserted) AND (IO2 asserted)"

16.6 Software control

To use the functionality of the GINT module, it is recommended to use software functions from within fsl_gint.c.

17.1 How to read this chapter

The DMA controller is available on all JN5189(T)/JN5188(T) devices.

17.2 Features

- 19 channels which are connected to peripheral DMA requests. These come from the USART, I^2C, SPI and SPIFI Interfaces, the hash peripheral and digital microphone peripheral. Any unused channel can also be used for memory-to-memory transfers.
- DMA operations can be triggered by on- or off-chip events. Each DMA channel can select one trigger input from 18 sources. Trigger sources include ADC interrupts, Timer interrupts, pin interrupts, and the SCT DMA request lines.
- Priority is user selectable for each channel (up to eight priority levels).
- Continuous priority arbitration.
- Address cache with four entries (each entry is a pair of transfer addresses).
- Efficient use of data bus.
- Supports single transfers up to 1,024 words.
- Address increment options allow packing and/or unpacking data.

17.3 Basic configuration

Configure the DMA as follows:
Use the SYSCON_AHBCLKCTRL0 register to enable the clock to the DMA registers interface.

- Clear the DMA peripheral reset using the SYSCON_PRESETCTRL0 register.
- The DMA controller provides an interrupt to the NVIC.
- Most peripherals that support DMA, the ADC being an exception, have at least one DMA request line associated with them. The related channel(s) must be set up according to the desired operation. the ADC uses a trigger instead of a DMA request. DMA requests and triggers are described in detail in Section 17.5.1
- For peripherals using DMA requests, DMA operation must be triggered before any transfer will occur. This can be done by software, or can optionally be signaled by one of 18 hardware triggers, through the input mux registers INPUTMUX_DMA_ITRIG_INMUX[4:0]. DMA requests and triggers are described in detail in Section 17.5.1
- Trigger outputs may optionally cause other DMA channels to be triggered for more complex DMA functions. Trigger outputs are connected to INPUTMUX_DMA_INMUX_INMUX as inputs to DMA triggers.

For details on the trigger input and output multiplexing, see Section 13.5.2 “DMA trigger input multiplexing”.

17.4 Pin description

The DMA controller has no direct pin connections. However, some DMA triggers can be associated with pin functions (see Section 17.5.1.2).

17.5 General description

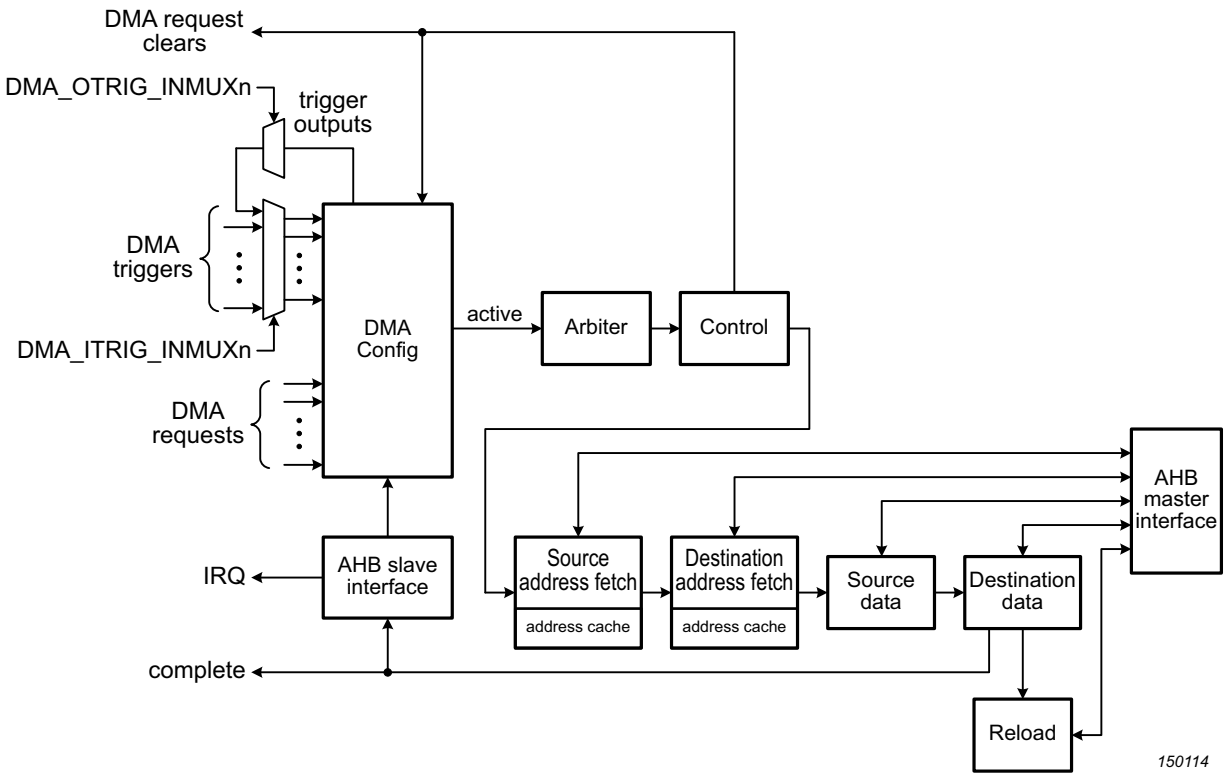


Fig 42. DMA block diagram

17.5.1 DMA requests and triggers

In general, DMA requests are intended to pace transfers to match what the peripheral (including its FIFO if it has one) can do. For example, the USART will issue a transmit DMA request when its transmit FIFO is not full, and a receive DMA request when its receive FIFO is not empty. DMA requests are summarized in Table 25.

Triggers start the transfer. In typical cases, only a software trigger will probably be used. Other possibilities are provided for, such as starting a DMA transfer when certain timer or pin related events occur. Those transfers would usually still be paced by a peripheral DMA request if a peripheral is involved in the transfer. Note that no DMA activity will take place for any particular DMA channel unless that channel has been triggered, either by software or hardware. DMA triggers are summarized in Table 25.

There is one specific exception to the above description, which is the ADC. The ADC doesn't fit the simple pacing signal model very well because of the possibilities represented by the programmable conversion sequences. A sequence complete DMA

request is likely to require transferring several non-contiguous result registers at once (see Chapter 27 “12-bit ADC Controller (ADC)”). It might also require other things to be done that can be done by the DMA without software intervention. This model fits better with the trigger facility, so that is how the ADC is connected to the DMA controller.

Once triggered by software or hardware, a DMA operation on a specific channel is initiated by a DMA request if it is enabled for that channel.

A DMA channel using a trigger can respond by moving data from any memory address to any other memory address. This can include fixed peripheral data registers, or incrementing through RAM buffers. The amount of data moved by a single trigger event can range from a single transfer to many transfers. A transfer that is started by a trigger can still be paced using the channel’s DMA request. This allows sending a string to a serial peripheral, for instance, without overrunning the peripheral’s transmit buffer.

Each DMA channel also has an output that can be used as a trigger input to another channel. The trigger outputs appear in the trigger source list for each channel and can be selected through the INPUTMUX_DMA_OTRIG_INMUX register as inputs to other channels.

17.5.1.1 DMA requests

DMA requests are directly connected to the peripherals. Each channel supports one DMA request line and one trigger input which is multiplexed to many possible input sources, as shown in Table 25.

Table 25. DMA requests & trigger muxes

DMA channel #	Request input	DMA trigger mux
0	USART 0 RX	DMA_ITRIG_INMUX0
1	USART 0 TX	DMA_ITRIG_INMUX1
2	USART 1 RX	DMA_ITRIG_INMUX2
3	USART 1 TX	DMA_ITRIG_INMUX3
4	I2C 0 Slave	DMA_ITRIG_INMUX4
5	I2C 0 Master	DMA_ITRIG_INMUX5
6	I2C 1 Slave	DMA_ITRIG_INMUX6
7	I2C 1 Master	DMA_ITRIG_INMUX7
8	SPI 0 RX	DMA_ITRIG_INMUX8
9	SPI 0 TX	DMA_ITRIG_INMUX9
10	SPI 1 RX	DMA_ITRIG_INMUX10
11	SPI 1 TX	DMA_ITRIG_INMUX11
12	SPIFI	DMA_ITRIG_INMUX12
13	I2C 2 Slave	DMA_ITRIG_INMUX13
14	I2C 2 Master	DMA_ITRIG_INMUX14
15	DMIC Channel 0	DMA_ITRIG_INMUX15
16	DMIC Channel 1	DMA_ITRIG_INMUX16
17	Hash RX	DMA_ITRIG_INMUX17
18	Hash TX	DMA_ITRIG_INMUX18

[1] See Section 17.5.1.1.1 below for information about DMA for the I2C Monitor function.

17.5.1.1.1 DMA with I²C monitor mode

The I²C monitor function may be used with DMA if one of the channels related to the same I²C Interface is available.

Table 26. DMA with the I²C Monitor function

I²C Master DMA	I²C Slave DMA	I²C Monitor DMA
Not enabled	-	If I²C Monitor DMA is enabled, it will use the DMA channel for the Master function of the same I²C Interface.
Enabled	Not enabled	If I²C Monitor is DMA enabled, it will use the DMA channel for the Slave function of the same I²C Interface.
Enabled	Enabled	The I²C Monitor function cannot use DMA.

17.5.1.2 Hardware triggers

Each DMA channel can use one trigger that is independent of the request input for this channel. The trigger input is selected in the INPUTMUX_DMA_ITRIG_INMUX register. There are 18 possible internal trigger sources for each DMA channel. In addition, the DMA trigger output can be routed to the trigger input of another channel through the trigger input multiplexing. See Table 25 and Section 13.5.2 “DMA trigger input multiplexing”.

Note that the ADC is unique in that it uses DMA triggers only, and has no DMA requests.

17.5.1.3 Trigger operation detail

A trigger of some kind is always needed to start a transfer on a DMA channel. This can be a hardware or software trigger, and can be used in several ways.

If a channel is configured with the XFERCFGn[SWTRIG] bit equal to 0, the channel can be later triggered either by hardware or software. Software triggering is accomplished by writing a 1 to the appropriate bit in the SETTRIG register. Hardware triggering requires setup of the HWTRIGEN, TRIGPOL, TRIGTYPE, and TRIGBURST fields in the CFG register for the related channel. When a channel is initially set up, the XFERCFGn[SWTRIG] can be set, causing the transfer to begin immediately.

Once triggered, transfer on a channel will be paced by DMA requests if the CFGn[PERIPHREQEN] bit is set. Otherwise, the transfer will proceed at full speed.

The CTLSTATn[TRIG] bit can be cleared at the end of a transfer, determined by the value XFERCFGn[SWTRIG]. When a 1 is found in XFERCFGn[CLRTRIG], the trigger is cleared when the descriptor is exhausted.

17.5.1.4 Trigger output detail

Each channel of the DMA controller provides a trigger output. This allows the possibility of using the trigger outputs as a trigger source to a different channel in order to support complex transfers on selected peripherals. This kind of transfer can, for example, use more than one peripheral DMA request. An example use would be to input data to a holding buffer from one peripheral, and then output the data to another peripheral, with both transfers being paced by the appropriate peripheral DMA request. This kind of operation is called “chained operation” or “channel chaining”.

17.5.2 DMA Modes

The DMA controller doesn't really have separate operating modes, but there are ways of using the DMA controller that have commonly used terminology in the industry.

Once the DMA controller is set up for operation, using any specific DMA channel requires initializing the registers associated with that channel (see Table 25), and supplying at least the channel descriptor, which is located somewhere in memory, typically in on-chip SRAM. The channel descriptor is shown in Table 27.

Table 27: Channel descriptor

Offset	Description
+ 0x0	Reserved
+ 0x4	Source data end address
+ 0x8	Destination end address
+ 0xC	Link to next descriptor

The source and destination end addresses, as well as the link to the next descriptor are just memory addresses that can point to any valid address on the device. The starting address for both source and destination data is the specified end address minus the transfer length (XFERCOUNT * the address increment as defined by SRCINC and DSTINC). The link to the next descriptor is used only if it is a linked transfer.

After the channel has had a sufficient number of DMA requests and/or triggers, depending on its configuration, the initial descriptor will be exhausted. At that point, if the transfer configuration directs it, the channel descriptor will be reloaded with data from memory pointed to by the "Link to next descriptor" entry of the initial channel descriptor. Descriptors loaded in this manner look slightly different the channel descriptor, as shown in Table 28. The difference is that a new transfer configuration is specified in the reload descriptor instead of being written to the XFERCFG register for that channel.

This process repeats as each descriptor is exhausted as long as reload is selected in the transfer configuration for each new descriptor.

Table 28: Reload descriptors

Offset	Description
+ 0x0	Transfer configuration.
+ 0x4	Source end address. This points to the address of the last entry of the source address range if the address is incremented. The address to be used in the transfer is calculated from the end address, data width, and transfer size.
+ 0x8	Destination end address. This points to the address of the last entry of the destination address range if the address is incremented. The address to be used in the transfer is calculated from the end address, data width, and transfer size.
+ 0xC	Link to next descriptor. If used, this address must be aligned to a multiple of 16 bytes (i.e., the size of a descriptor).

17.5.3 Single buffer

This generally applies to memory to memory moves, and peripheral DMA that occurs only occasionally and is set up for each transfer. For this kind of operation, only the initial channel descriptor shown in Table 29 is needed.

Table 29: Channel descriptor for a single transfer

Offset	Description
+ 0x0	Reserved
+ 0x4	Source data end address
+ 0x8	Destination data end address
+ 0xC	(not used)

This case is identified by the XFERCFGn[RELOAD] = 0. When the DMA channel receives a DMA request or trigger (depending on how it is configured), it performs one or more transfers as configured, then stops. Once the channel descriptor is exhausted, additional DMA requests or triggers will have no effect until the channel configuration is updated by software.

17.5.4 Ping-Pong

Ping-pong is a special case of a linked transfer. It is described separately because it is typically used more frequently than more complicated versions of linked transfers.

A ping-pong transfer uses two buffers alternately. At any one time, one buffer is being loaded or unloaded by DMA operations. The other buffer has the opposite operation being handled by software, readying the buffer for use when the buffer currently being used by the DMA controller is full or empty. Table 30 shows an example of descriptors for ping-pong from a peripheral to two buffers in memory.

Table 30: Example descriptors for ping-pong operation: peripheral to buffer

Channel Descriptor	
+ 0x0	(not used)
+ 0x4	Peripheral data end address
+ 0x8	Buffer A memory end address
+ 0xC	Address of descriptor B

Descriptor B	
+ 0x0	Buffer B transfer configuration
+ 0x4	Peripheral data end address
+ 0x8	Buffer B memory end address
+ 0xC	Address of descriptor A

Descriptor A	
+ 0x0	Buffer A transfer configuration
+ 0x4	Peripheral data end address
+ 0x8	Buffer A memory end address
+ 0xC	Address of descriptor B

In this example, the channel descriptor is used first, with a first buffer in memory called buffer A. The configuration of the DMA channel must have been set to indicate a reload. Similarly, both descriptor A and descriptor B must also specify reload. When the channel descriptor is exhausted, descriptor B is loaded using the link to descriptor B, and a transfer interrupt informs the CPU that buffer A is available.

Descriptor B is then used until it is also exhausted, when descriptor A is loaded using the link to descriptor A contained in descriptor B. Then a transfer interrupt informs the CPU that buffer B is available for processing. The process repeats when descriptor A is exhausted, alternately using each of the 2 memory buffers.

17.5.5 Interleaved transfers

One use for the XFERCFGn[SRCINC] and XFERCFGn[DSTINC] configurations is to handle data in a buffer such that it is interleaved with other data.

For example, if 4 data samples from several peripherals need to be interleaved into a single data structure, this may be done while the data is being read in by the DMA. Setting SRCINC to 4x width for each channel involved will allow room for 4 samples in a row in the buffer memory. The DMA will place data for each successive value at the next location for that peripheral.

The reverse of this process could be done using XFERCFGn[DSTINC] to de-interleave combined data from the buffer and send it to several peripherals or locations.

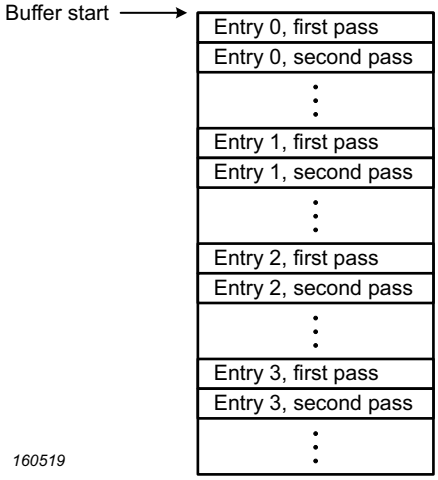


Fig 43. Interleaved transfer in a single buffer

17.5.6 Linked transfers (linked list)

A linked transfer can use any number of descriptors to define a complicated transfer. This can be configured such that a single transfer, a portion of a transfer, one whole descriptor, or an entire structure of links can be initiated by a single DMA request or trigger.

An example of a linked transfer could start out like the example for a ping-pong transfer (Table 30). The difference would be that descriptor B would not link back to descriptor A, but would continue on to another different descriptor. This could continue as long as desired, and can be ended anywhere, or linked back to any point to repeat a sequence of descriptors. Of course, any descriptor not currently in use can be altered by software as well.

17.5.7 Address alignment for data transfers

Transfers of 16-bit width require an address alignment to a multiple of 2 bytes. Transfers of 32-bit width require an address alignment to a multiple of 4 bytes. Transfers of 8-bit width can be at any address.

17.5.8 Channel chaining

Channel chaining is a feature which allows completion of a DMA transfer on channel x to trigger a DMA transfer on channel y. This feature can for example be used to have DMA channel x reading n bytes from USART to memory, and then have DMA channel y transferring the received bytes to the CRC engine, without any action required from the Arm core.

To use channel chaining, first configure DMA channels x and y as if no channel chaining would be used. Then:

- For channel x:

- If channel x is configured to auto reload the descriptor on exhausting of the descriptor (XFERCFGn[RELOAD] is set), then enable 'clear trigger on descriptor exhausted' by setting bit XFERCFGn[CLRTRIG].

- For channel y:
  - Configure the input trigger input mux register (INPUTMUX_DMA_ITRIG_INMUX[4:0]) for channel y to use any of the available DMA trigger muxes (DMA trigger mux 0/1).
  - Configure the chosen DMA trigger mux to select DMA channel x.
  - Enable hardware triggering by setting bit CFGn[HWTRIGEN].
  - Set the trigger type to edge sensitive by clearing bit CFGn[TRIGTYPE].
  - Configure the trigger edge to falling edge by clearing bit CFGn[TRIGPOL].

Note that after completion of channel x the descriptor may be reloaded (if configured so), but remains un-triggered. To configure the chain to auto-trigger itself, setup channels x and y for channel chaining as described above. In addition to that:

- A ping-pong configuration for both channel x and y is recommended, so that data currently moved by channel y is not altered by channel x.
- For channel x:
  - Configure the input trigger input mux register (INPUTMUX_DMA_ITRIG_INMUX[4:0]) for channel y to use the same DMA trigger mux as chosen for channel y.
  - Enable hardware triggering by setting bit CFGn[HWTRIGEN].
  - Set the trigger type to edge sensitive by clearing bit CFGn[TRIGTYPE].
  - Configure the trigger edge to falling edge by clearing bit CFGn[TRIGPOL]..

17.5.9 DMA in reduced power modes

DMA in sleep mode

In sleep mode, the DMA can operate and access all enabled SRAM blocks, without waking the CPU.

DMA in deep-sleep mode

DMA operation is not possible in deep-sleep mode.

17.6 Functional Description

17.6.1 Control Register

The control register contains the control bit to enable the DMA controller.

17.6.2 Interrupt Status Register

The read-only INTSTAT register provides an overview of DMA status. This allows quick determination of whether any enabled interrupts are pending. Details of which channels are involved can be found in the interrupt type specific registers.

17.6.3 SRAM Base address register

The DMA function uses a DMA descriptor table which is located in SRAM. The SRAMBASE register must be configured with an address where DMA descriptors will be stored. Software must set up the descriptors for those DMA channels that will be used in the application.

Each DMA channel has an entry for the channel descriptor in the SRAM table. The values for each channel start at the address offsets found in Table 31. Only the descriptors for channels in use are used. The contents of each channel descriptor are described in Table 27.

Table 31: Channel descriptor map

Descriptor	Offset
Channel descriptor for DMA channel 0	0x000
Channel descriptor for DMA channel 1	0x010
Channel descriptor for DMA channel 2	0x020
Channel descriptor for DMA channel 3	0x030
Channel descriptor for DMA channel 4	0x040
Channel descriptor for DMA channel 5	0x050
Channel descriptor for DMA channel 6	0x060
Channel descriptor for DMA channel 7	0x070
Channel descriptor for DMA channel 8	0x080
Channel descriptor for DMA channel 9	0x090
Channel descriptor for DMA channel 10	0x0A0
Channel descriptor for DMA channel 11	0x0B0
Channel descriptor for DMA channel 12	0x0C0
Channel descriptor for DMA channel 13	0x0D0
Channel descriptor for DMA channel 14	0x0E0
Channel descriptor for DMA channel 15	0x0F0
Channel descriptor for DMA channel 16	0x100
Channel descriptor for DMA channel 17	0x110
Channel descriptor for DMA channel 18	0x120

17.6.4 Enable Set Register

The ENABLESET0 register determines whether each DMA channel is enabled or disabled. Disabling a DMA channel does not reset the channel in any way. A channel can be paused and restarted by clearing, then setting the Enable bit for that channel. Reading ENABLESET0 provides the current state of all of the DMA channels represented by that register. Writing a 1 to a bit position in ENABLESET0 that corresponds to an implemented DMA channel sets the bit, enabling the related DMA channel. Writing a 0 to any bit has no effect. Enables are cleared by writing to ENABLECLR0.

17.6.5 Enable Clear Register

The ENABLECLR0, write-only, register is used to clear the enable of one or more DMA channels.

17.6.6 Active status register

The ACTIVE0 register indicates which DMA channels are active at the point when the read occurs. The register is read-only.

A DMA channel is considered active when a DMA operation has been started but not yet fully completed. The Active status will persist from a DMA operation being started, until the pipeline is empty after end of the last descriptor (when there is no reload). An active channel may be aborted by software by setting the appropriate bit in one of the Abort register (see Section 17.6.15)

17.6.7 Busy Status register

The BUSY0 register indicates which DMA channels are busy at the point when the read occurs. This register is read-only.

A DMA channel is considered busy when there is any operation related to that channel in the DMA controller's internal pipeline. This information can be used after a DMA channel is disabled by software (but still active), allowing confirmation that there are no remaining operations in progress for that channel.

17.6.8 Error Interrupt register

The ERRINT0 register contains flags for each DMA channel's Error Interrupt. Any pending interrupt flag in the register will be reflected on the DMA interrupt output.

Reading the registers provides the current state of all DMA channel error interrupts. Writing a 1 to a bit position in ERRINT0 that corresponds to an implemented DMA channel clears the bit, removing the interrupt for the related DMA channel. Writing a 0 to any bit has no effect.

17.6.9 Interrupt Enable read and Set register

The INTENSET0 register controls whether the individual Interrupts for DMA channels contribute to the DMA interrupt output.

Reading the registers provides the current state of all DMA channel interrupt enables. Writing a 1 to a bit position in INTENSET0 that corresponds to an implemented DMA channel sets the bit, enabling the interrupt for the related DMA channel. Writing a 0 to any bit has no effect. Interrupt enables are cleared by writing to INTENCLR0.

17.6.10 Interrupt Enable Clear register

The INTENCLR0 register is used to clear interrupt enable bits in INTENSET0. The register is write-only.

17.6.11 Interrupt A register

The INTA0 register contains the interrupt A status for each DMA channel. The status will be set when the SETINTA bit is 1 in the transfer configuration for a channel, when the descriptor becomes exhausted. Writing a 1 to a bit in this register clears the related INTA flag. Writing 0 has no effect. Any interrupt pending status in the registers will be reflected on the DMA interrupt output if it is enabled in the related INTENSET0 register.

17.6.12 Interrupt B register

The INTB0 register contains the interrupt B status for each DMA channel. The status will be set when the SETINTB bit is 1 in the transfer configuration for a channel, when the descriptor becomes exhausted. Writing a 1 to a bit in the register clears the related INTB flag. Writing 0 has no effect. Any interrupt pending status in this register will be reflected on the DMA interrupt output if it is enabled in the INTENSET0 register.

17.6.13 Set valid register

The SETVALID0 register allows setting the Valid bit in the CTLSTATn register for one or more DMA channels. See Section 17.6.17 for a description of the VALID bit. This register is write-only.

The XFERCFGn[CFGVALID] and SV (set valid) bits allow more direct DMA block timing control by software. Each Channel Descriptor, in a sequence of descriptors, can be validated by either the setting of the CFGVALID bit or by setting the channel's SETVALID flag. Normally, the CFGVALID bit is set. This tells the DMA that the Channel Descriptor is active and can be executed. The DMA will continue sequencing through descriptor blocks whose CFGVALID bit are set without further software intervention. Leaving a CFGVALID bit set to 0 allows the DMA sequence to pause at the Descriptor until software triggers the continuation. If, during DMA transmission, a Channel Descriptor is found with CFGVALID set to 0, the DMA checks for a previously buffered SETVALID0 setting for the channel. If found, the DMA will set the descriptor valid, clear the SV setting, and resume processing the descriptor. Otherwise, the DMA pauses until the channels SETVALID0 bit is set.

17.6.14 Set Trigger register

The SETTRIG0 register allows setting the CTLSTATn[TRIG] for one or more DMA channel. See Section 17.6.17 for a description of the TRIG bit, and Section 17.5.1 for a general description of triggering. This register is write-only.

17.6.15 Abort Register

The ABORT0 register allows aborting operation of a DMA channel if needed. To abort a selected channel, the channel should first be disabled by clearing the corresponding Enable bit by writing a 1 to the proper bit in ENABLECLR0 register. Then wait until the channel is no longer busy by checking the corresponding bit in BUSY0 register. Finally, write a 1 to the proper bit of ABORT0 register. This prevents the channel from restarting an incomplete operation when it is enabled again. This register is write-only.

17.6.16 Channel Configuration registers

The CFGn register contains various configuration options for DMA channel n.

- PERIPHREQEN: control if channel is to be linked to its peripheral
- HWTRIGEN: control if hardware triggering is to be used
- TRIGPOL: configure trigger to be rising or falling edge
- TRIGTYPE: configure if trigger is edge triggered or level triggered
- TRIGBURST: configure if the trigger causes a single transfer or a burst
- BURSTPOWER: configure the size of a burst, up to 1024 transfers

- SRCBURSTWRAP: configure whether the source address range for each burst is the same
- DSTBURSTWRAP: configure whether the source address range for each burst is the same
- CHPRIORITY: configure channel priority

Table 32: Trigger setting summary

TrigBurst	TrigType	TrigPol	Description
0	0	0	Hardware DMA trigger is falling edge sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap.
0	0	1	Hardware DMA trigger is rising edge sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap.
0	1	0	Hardware DMA trigger is low level sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap.
0	1	1	Hardware DMA trigger is high level sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap.
1	0	0	Hardware DMA trigger is falling edge sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap, and also determines how much data is transferred for each trigger.
1	0	1	Hardware DMA trigger is rising edge sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap, and also determines how much data is transferred for each trigger.
1	1	0	Hardware DMA trigger is low level sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap, and also determines how much data is transferred for each trigger.
1	1	1	Hardware DMA trigger is high level sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap, and also determines how much data is transferred for each trigger.

17.6.17 Channel Control and Status Registers

The CTLSTATn, read only, register provides status flags specific to DMA channel n. Status information indicates the 'Valid pending' status and the trigger status.These registers are read-only.

17.6.18 Channel transfer configuration registers

The XFERCFGn register contains transfer related configuration information for DMA channel n.

- CFGVALID: this indicates if the channel descriptor is valid
- RELOAD: Using the Reload bit, this register can optionally be automatically reloaded when the current settings are exhausted (the full transfer count has been completed), allowing linked transfers with more than one descriptor to be performed.

- SWTRIG: Used to trigger the DMA channel by SW
- CLRTRIG: Controls if trigger is cleared when the descriptor is exhausted
- SETINTA: control if INTA flag is set when the descriptor is exhausted
- SETINTB controls if INTB flag is set when the descriptor is exhausted
- WIDTH configured if 8, 16 or 32 bit transfers are performed
- SRCINC: configure increment mode for source address
- DSTINC: configure increment mode for the destination address
- XFERCOUNT: configures the number of transfers to be performed

See Section 17.5.1.3 “Trigger operation detail” for details on trigger operation.

17.7 Software control

To use the functionality of the DMA module, it is recommended to use software functions from within fsl_dma.c.

18.1 How to read this chapter

The PWM is available on all JN5189(T)/JN5188(T) devices.

18.2 Features

The features of the PWM are:

- Ten 16-bit auto reload down counters
- Operate on APB clock
- Programmable 10-bit prescaler for the ten channels
- Predictable PWM initial output state
- Buffered compare register and polarity register to ensure correct PWM output
- Programmable overflow interrupt generation
- Configurable level (high or low) of PWM output when PWM is disabled
- Option to drive all ten outputs from a single PWM channel

18.3 Basic configuration

Configure the PWM as follows:

- Enable the clock to the PWM by setting SYSCON_AHBCLKCTRL1[PWM] to enable the register interface and the peripheral clock.
- The PWM provides interrupt to the NVIC, PWM0_IRQn to PWM10_IRQn, see Table 17 “Connection of interrupt sources to the NVIC”
- Configure the required connections to the device IO pins

18.4 Pin description

See Chapter 12 “I/O Pin Configuration (IOCON)” to assign PWM functions to external pins.

Two of the PWM channels, PWM8 and PWM9, can be used as a trigger for the ADC SEQ controller. See Chapter 27 “12-bit ADC Controller (ADC)” for this configuration.

Table 33. PWM possible output connections

PWM output	IO with pull-down default	IO with pull-up default
PWM0		PIO0 and PIO12
PWM1	PIO1	PIO14
PWM2	PIO2	PIO13
PWM3		PIO3 or PIO15
PWM4	PIO19	PIO4
PWM5		PIO16
PWM6	PIO6 or PIO17	
PWM7	PIO7 or PIO18	
PWM8	PIO20	PIO8
PWM9		PIO9 or PIO21

18.5 General description

The general architecture of the PWM block is shown in Figure 44. The PWM block can control up to ten PWM channels. These are either ten independent channels or they can all be driven from one PWM channel (PWM10). The block receives only the APB clock and there is a prescaler of each PWM channel to support a wide range of PWM periods. Each of the PWM channels, including pwm10, can generate an interrupt to the processors.

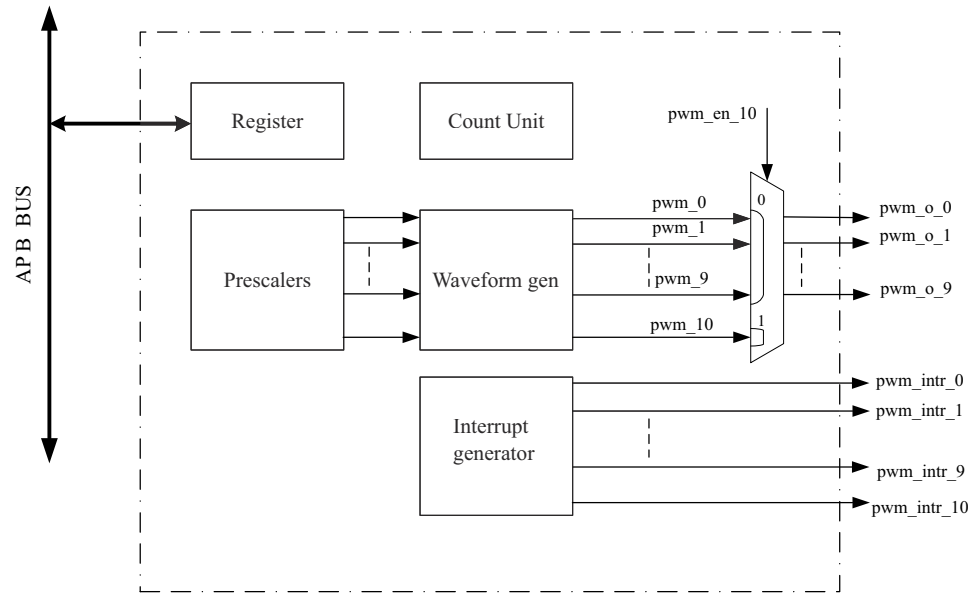


Fig 44. PWM Architecture

18.6 Functional description

18.6.1 Prescaler

There are eleven 10-bit prescalers. The frequency of the scaled clock enable is calculated as follows.

$F_{scaled} = F_{clk} / (pscl +1)$

Where F_{clk} is the frequency of the AHB clock, pscl is the prescaler setting for the PWM channel.

18.6.2 Counter Unit

The main part of the counter unit is eleven 16-bit counters and its control logic. The following figure shows a block diagram of the counter unit (n=0,1,2 . . . 10)

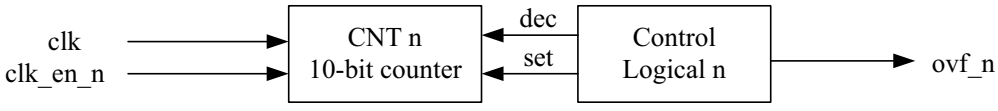


Fig 45. PWM Counter Block

18.6.3 Waveform generator

There is one compare, polarity, disable state level and period register for each channel. The PWM period is updated at the end of each PWM period, and the compare and polarity are loaded from their buffers respectively.

- Compare: when the compare value is reached the PWM output will change on the next counter decrement and be stable from 'compare-1' to 0
- Disable state level: when the PWM channel is disabled, this sets the level of the PWM channel. Note: the common PWM channel (PWM10) does not have this setting; when it is not enabled each of the ten PWM channel will then take settings based on their own control registers.
- Period register: PWM counter will decrement from the period value and give actual period of 'Period +1'
- Polarity: when configured low then the PWM o/p is set high on a compare match; when configured high then the PWM o/p is set low on a compare match

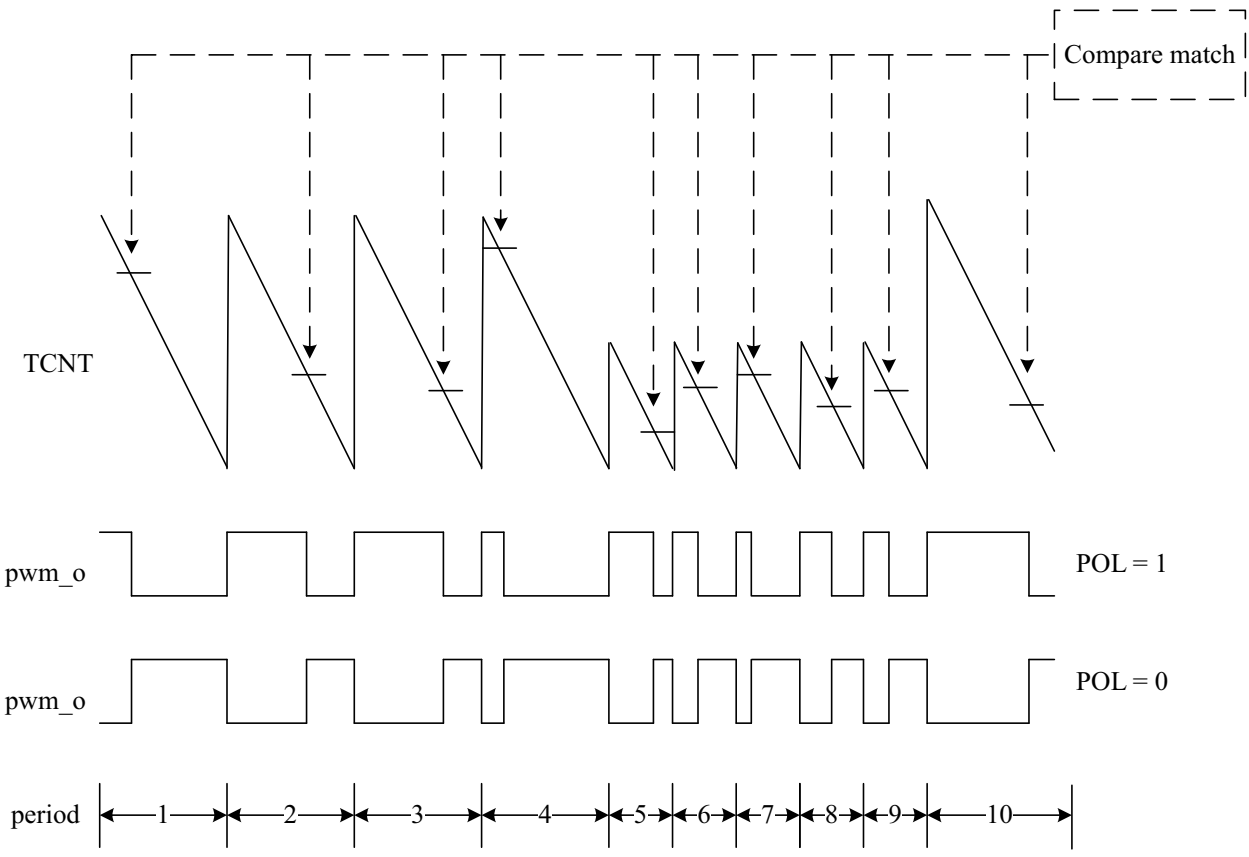


Fig 46. PWM Waveform

18.6.4 Interrupt generator

There are eleven overflow interrupts which connect directly to the NVIC interface of the MCU. There interrupts can be read from the PWM status registers PST0 to PST2. Pending status bits can be cleared by writing 1 to the corresponding bit.

The interrupts can be enabled in the control register CTRL0; there is a separate enable bit for each channel.

18.6.5 Common PWM mode

By enabling PWM10, all 10 PWM outputs are driven by the PWM10 channel and so they will all have the same output.

18.6.6 PWM trigger for ADC

PWM8 and PWM9 can be used as trigger sources for ADC SEQ sequencer. Apart from configuring the PWM8 or 9 channels, no extra configuration is required within the PWM block to enable this feature.

18.7 Software control

To use the functionality of the PWM module it is recommended to use software functions from within fsl_pwm.c.

19.1 How to read this chapter

These two standard timers are available on all JN5189(T)/JN5188(T) devices.

19.2 Features

- Each is a 32-bit counter/timer with a programmable 32-bit prescaler. Both of the timers include two capture and two match pin connections.
- Counter or timer operation.
- Up to four 32-bit captures that can take a snapshot of the timer value when an input signal transitions. A capture event may also optionally generate an interrupt.
- The timer and prescaler may be configured to be cleared on a designated capture event. This feature permits easy pulse-width measurement by clearing the timer on the leading edge of an input pulse and capturing the timer value on the trailing edge.
- Four 32-bit match registers that allow:
  - Continuous operation with optional interrupt generation on match.
  - Stop timer on match with optional interrupt generation.
  - Reset timer on match with optional interrupt generation.
- Up to 2 external outputs corresponding to match registers with the following capabilities (the number of match outputs for each timer that are actually available on device pins may vary by part number):
  - Set LOW on match.
  - Set HIGH on match.
  - Toggle on match.
  - Do nothing on match.
- Up to 4 match registers can be configured for PWM operation, allowing up to 2 single edged controlled PWM outputs.
- Up to 2 match registers can be used to generate DMA requests.

19.3 Basic configuration

- Set the appropriate bits to enable clocks to timers that will be used: CTIMER0 and CTIMER1 in the ASYNCAPBCLKCTRL register.
- Clear the timer reset using the ASYNCPRESETCTRL register. Note that bit positions in the reset control registers match the bit positions in the clock control registers.
- Pins: Select timer pins and pin modes as needed through the relevant IOCON registers (Chapter 12 “I/O Pin Configuration (IOCON)”).
- Interrupts: See register MCR and CCR for match and capture events. Interrupts must be enabled in the NVIC, see Table 17 “Connection of interrupt sources to the NVIC” for information on enabling interrupts.

- DMA: Some timer match conditions can be used to generate timed DMA requests, see Table 25 “DMA requests & trigger muxes”.

19.4 Applications

- Interval Timer for counting internal events.
- PWM outputs
- Pulse Width Demodulator via Capture inputs.
- Free running timer.

19.5 General description

The counter/timer block is used to count cycles of the APB bus clock or an externally supplied clock and can optionally generate interrupts or perform other actions at specified timer values based on four match registers. Each counter/timer also includes capture inputs to trap the timer value when an input signal transitions, optionally generating an interrupt.

In PWM mode, three match registers can be used to provide a single-edge controlled PWM output on the match output pins. One match register is used to control the PWM cycle length.

19.5.1 Capture inputs

The capture signal can be configured to load the CCR register (CR0 or CR1) with the value in the counter/timer and optionally generate an interrupt. The capture signal is generated by one of the pins with a capture function. Each capture signal is connected to one capture channel of the timer.

The Counter/Timer block can select a capture signal as a clock source instead of the APB bus clock.

19.5.2 Match outputs

When a March register (MR0, MR1, MR2 or MR3) equals the timer counter (TC), the corresponding match output can either toggle, go LOW, go HIGH, or do nothing. The External Match Register (EMR) and the PWM Control Register (PWMC) control the functionality of this output.

19.5.3 Architecture

The block diagram for the timers is shown in Figure 47.

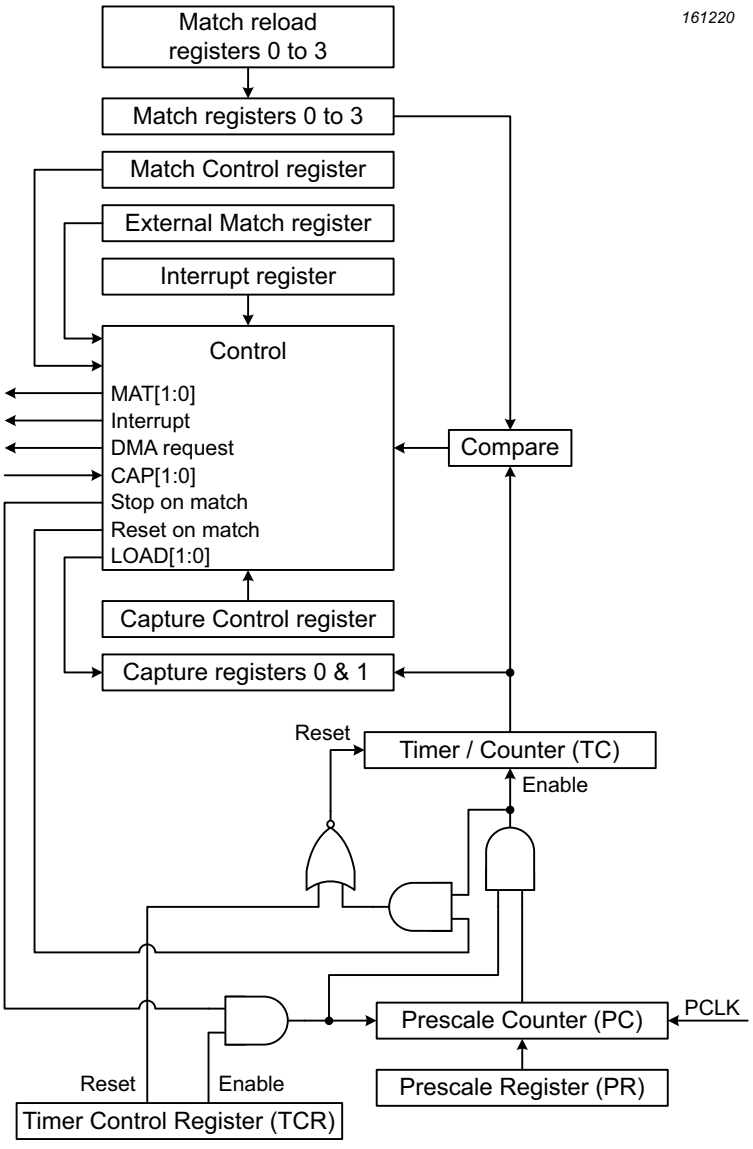


Fig 47. 32-bit counter/timer block diagram

19.6 Pin description

Table 34 gives a summary of each of the Timer/Counter related pins. This is followed by information about IOs that could be used and the configuration for these IOs. Also refer to Table 19 “IOMUX functions”. Recommended IOCON settings are shown in Table 35, Table 36, and Table 37.

Table 34. Timer/Counter pin description

Pin	Type	Description
CTIMER0_CAP1:0 CTIMER1_CAP1:0	Input	Capture Signals- A transition on a capture pin can be configured to load one of the Capture Registers with the value in the Timer Counter and optionally generate an interrupt. Timer/Counter block can select a capture signal as a clock source instead of the APB bus clock.
CTIMER0_MAT1:0 CTIMER1_MAT1:0	Output	External Match Output - When a match register (MR3:0) equals the timer counter (TC) this output can either toggle, go low, go high, or do nothing. The External Match Register (EMR) controls the functionality of this output. Note that match conditions may be used internally without the use of a device pin.

Table 35. CTIMER possible IO connections

CTIMER signal	IO	IO cell type
Timer0 Cap0	PIO0_10	Combo IO cell
Timer0 Cap1	PIO0_14	Standard GPIO IO cell
Timer0 Mat0	PIO0_8	Standard GPIO IO cell
Timer0 Mat1	PIO0_18	Standard GPIO IO cell
Timer1 Cap0	PIO0_11	Combo IO cell
Timer1 Cap1	PIO0_9	Standard GPIO IO cell
Timer1 Mat0	PIO0_6	Standard GPIO IO cell
Timer1 Mat1	PIO0_7	Standard GPIO IO cell

Table 36: Suggested CTIMER pin setting for standard GPIO IO

IOCON bit(s)	Field	Setting	Comment
2:0	Func	Set for CTIMER function	
4:3	Mode	2	No pullup or pull-down
5	Slew0	0	See slew1
6	Invert	0	No need to invert
7	Digimode	1	Digital mode
8	FilterOff	1	Generally disable filter
9	Slew1	0	With slew0: generally set to 0 for low speed signals
10	OD	0	Normal GPIO mode
11	IO_clamp	0	Clamp Off

Table 37: Suggested CTIMER pin setting for combo IO cells (PIO10 and PIO11)

IOCON bit(s)	Field	Setting	Comment
2:0	Func	Set for CTIMER function	
3	EGP	1	GPIO mode
4	ECS	0	Standard GPIO mode
5	EHS	0	Low speed GPIO. Generally set to 0 for low speed signals

Table 37: Suggested CTIMER pin setting for combo IO cells (PIO10 and PIO11)

IOCON bit(s)	Field	Setting	Comment
6	Invert	0	No need to invert
7	Digimode	1	Digital mode
8	FilterOff	1	Generally disable filter
9	Fsel	0	Not valid for GPIO mode
10	OD	0	Normal GPIO mode
12	IO_clamp	0	Clamp Off

19.7 Functional description

Figure 48 shows a timer configured to reset the count and generate an interrupt on match. The prescaler is set to 2 and the match register set to 6. At the end of the timer cycle where the match occurs, the timer count is reset. This gives a full length cycle to the match value. The interrupt indicating that a match occurred is generated in the next clock after the timer reached the match value.

Figure 49 shows a timer configured to stop and generate an interrupt on match. The prescaler is again set to 2 and the match register set to 6. In the next clock after the timer reaches the match value, the timer enable bit in TCR is cleared, and the interrupt indicating that a match occurred is generated.

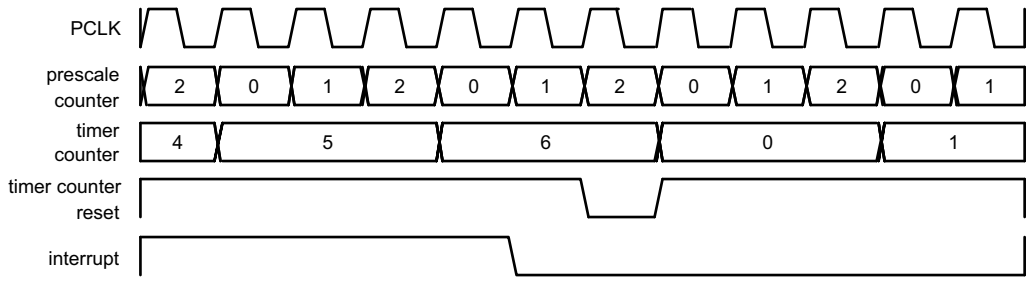


Fig 48. A timer cycle in which PR=2, MRx=6, and both interrupt and reset on match are enabled

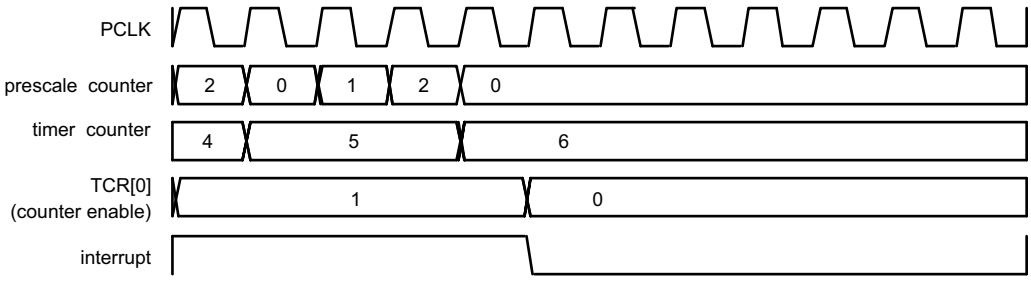


Fig 49. A timer cycle in which PR=2, MRx=6, and both interrupt and stop on match are enabled

19.7.1 Rules for single edge controlled PWM outputs

1. All single edge controlled PWM outputs go LOW at the beginning of each PWM cycle (timer is set to zero) unless their match value is equal to zero.
2. Each PWM output will go HIGH when its match value is reached. If no match occurs (i.e. the match value is greater than the PWM cycle length), the PWM output remains continuously LOW.
3. If a match value larger than the PWM cycle length is written to the match register, and the PWM signal is HIGH already, then the PWM signal will be cleared with the start of the next PWM cycle.
4. If a match register contains the same value as the timer reset value (the PWM cycle length), then the PWM output will be reset to LOW on the next clock tick after the timer reaches the match value. Therefore, the PWM output will always consist of a one clock tick wide positive pulse with a period determined by the PWM cycle length (i.e. the timer reload value).
5. If a match register is set to zero, then the PWM output will go to HIGH the first time the timer goes back to zero and will stay HIGH continuously.

Note: When the match outputs are selected to perform as PWM outputs, the timer reset and timer stop bits (MCR[MRnR] and MCR[MRnS]) must be set to zero except for the match register setting the PWM cycle length. For this register, set the MRnR bit to 1 to enable the timer reset when the timer value matches the value of the corresponding match register.

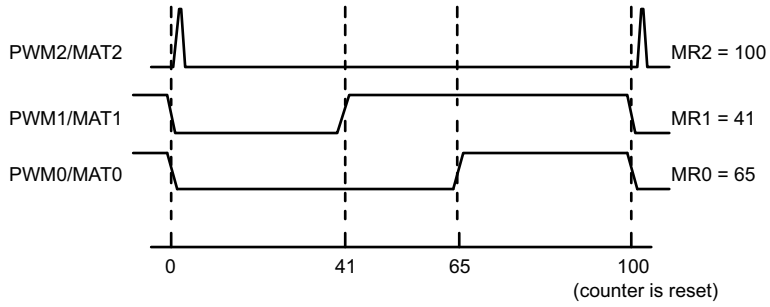


Fig 50. Sample PWM waveforms with a PWM cycle length of 100 (selected by MR3) and MAT3:0 enabled as PWM outputs by the PWCON register.

19.7.2 DMA operation

DMA requests are generated by a match of the Timer Counter (TC) register value to either Match Register 0 (MR0) or Match Register 1 (MR1). This is not connected to the operation of the Match outputs controlled by the EMR register. Each match sets a DMA request flag, which is connected to the DMA controller. In order to have an effect, the DMA controller must be configured correctly.

When a timer is initially set up to generate a DMA request, the request may already be asserted before a match condition occurs. An initial DMA request may be avoided by having software write a 1 to the interrupt flag location, as if clearing a timer interrupt. A DMA request will be cleared automatically when it is acted upon by the DMA controller.

Note: Because timer DMA requests are generated whenever the timer value is equal to the related Match Register value, DMA requests are always generated when the timer is running, unless the Match Register value is higher than the upper count limit of the timer. It is important not to select and enable timer DMA requests in the DMA block unless the timer is correctly configured to generate valid DMA requests.

Note: To ensure proper operation when using DMAs with this peripheral, the DMA clock must be configured to be greater than the peripheral clock.

19.8 Software control

To use the functionality of the CTIMER modules, it is recommended to use software functions from within fsl_ctimer.c.

20.1 How to read this chapter

The watchdog timer is available on all JN5189(T)/JN5188(T) devices.

20.2 Features

- Internally resets chip if not reloaded during the programmable time-out period.
- Optional windowed operation requires reload to occur between a minimum and maximum time-out period, both programmable.
- Optional warning interrupt can be generated at a programmable time prior to watchdog time-out.
- Programmable 24-bit timer with internal fixed pre-scaler.
- Selectable time period from 1,024 watchdog clocks ($T_{WDCLK} \times 256 \times 4$) to over 67 million watchdog clocks ($T_{WDCLK} \times 2^{24} \times 4$) in increments of 4 watchdog clocks.
- “Safe” watchdog operation. Once enabled, requires a hardware reset or a Watchdog reset to be disabled.
- Incorrect feed sequence causes immediate watchdog event if enabled.
- The watchdog reload value can optionally be protected such that it can only be changed after the “warning interrupt” time is reached.
- Flag to indicate Watchdog reset.
- The Watchdog clock (WDCLK) source is a selectable frequency in the range of 6 kHz to 1.5 MHz. The accuracy of this clock is limited to ±15% over temperature, voltage, and silicon processing variations. To determine the actual watchdog oscillator output, use the frequency measure block. See Section 37.6 “Frequency measurement”.
- The Watchdog timer can be configured to run in deep-sleep mode.
- Debug mode.

20.3 Basic configuration

Configuration of the WWDT is accomplished as follows:

- Enable the watchdog oscillator using the software APIs, see Section 6.4 “Clock control software functions”.
- Enable the register interface (WWDT bus clock): set the SYSCON_AHBCLKCTRL0[WWDT].
- For waking up from a WWDT interrupt, enable the watchdog interrupt for wake-up in the SYSCON_STARTER0[WDT_BOD].

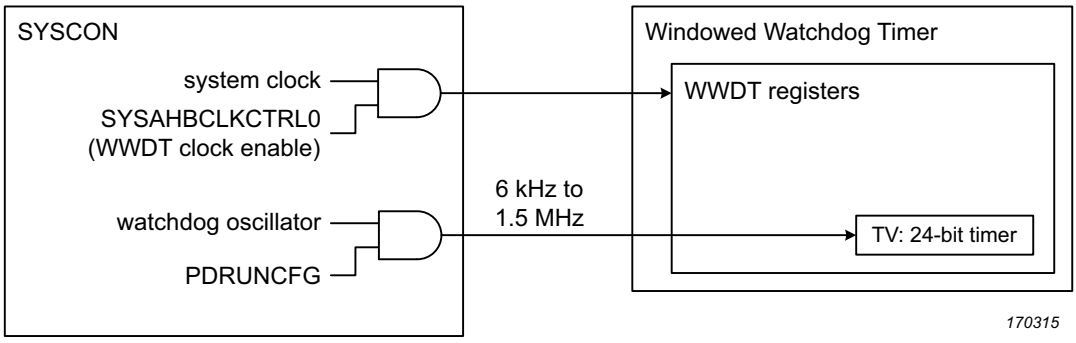


Fig 51. WWDT clocking

20.4 Pin description

The WWDT has no external pins.

20.5 General description

The purpose of the Watchdog Timer is to reset or interrupt the microcontroller within a programmable time if it enters an erroneous state. When enabled, a watchdog reset is generated if the user program fails to feed (reload) the Watchdog within a predetermined amount of time.

When a watchdog window is programmed, an early watchdog feed is also treated as a watchdog event. This allows preventing situations where a system failure may still feed the watchdog. For example, application code could be stuck in an interrupt service that contains a watchdog feed. Setting the window such that this would result in an early feed will generate a watchdog event, allowing for system recovery.

The Watchdog consists of a fixed (divide by 4) pre-scaler and a 24-bit counter which decrements when clocked. The minimum value from which the counter decrements is 0xFF. Setting a value lower than 0xFF causes 0xFF to be loaded in the counter. Hence the minimum Watchdog interval is ($T_{WDCLK} \times 256 \times 4$) and the maximum Watchdog interval is ($T_{WDCLK} \times 2^{24} \times 4$) in multiples of ($T_{WDCLK} \times 4$). The Watchdog should be used in the following manner:

- Enable and configure the Watchdog oscillator as described in Section 20.3 “Basic configuration”
- Set the Watchdog timer constant reload value in the TC register.
- Set the Watchdog timer operating mode in the WWDT_MOD register.
- Set a value for the watchdog window time in the WINDOW register if windowed operation is desired.
- Set a value for the watchdog warning interrupt in the WARNINT register if a warning interrupt is desired.
- Enable the Watchdog by writing 0xAA followed by 0x55 to the FEED register.

- The Watchdog must be fed again before the Watchdog counter reaches zero in order to prevent a watchdog event. If a window value is programmed, the feed must also occur after the watchdog counter passes that value.

When the Watchdog Timer is configured so that a watchdog event will cause a reset and the counter reaches zero, the CPU will be reset, loading the stack pointer and program counter from the vector table as for an external reset. The Watchdog time-out flag (MOD[WDTOF]) can be examined to determine if the Watchdog has caused the reset condition. The MOD[WDTOF] flag must be cleared by software.

When the Watchdog Timer is configured to generate a warning interrupt, the interrupt will occur when the counter is no longer greater than the value defined by the WARNINT register.

20.5.1 Block diagram

The block diagram of the Watchdog is shown below in the Figure 52. The synchronization logic (APB bus clock to WDCLK) is not shown in the block diagram.

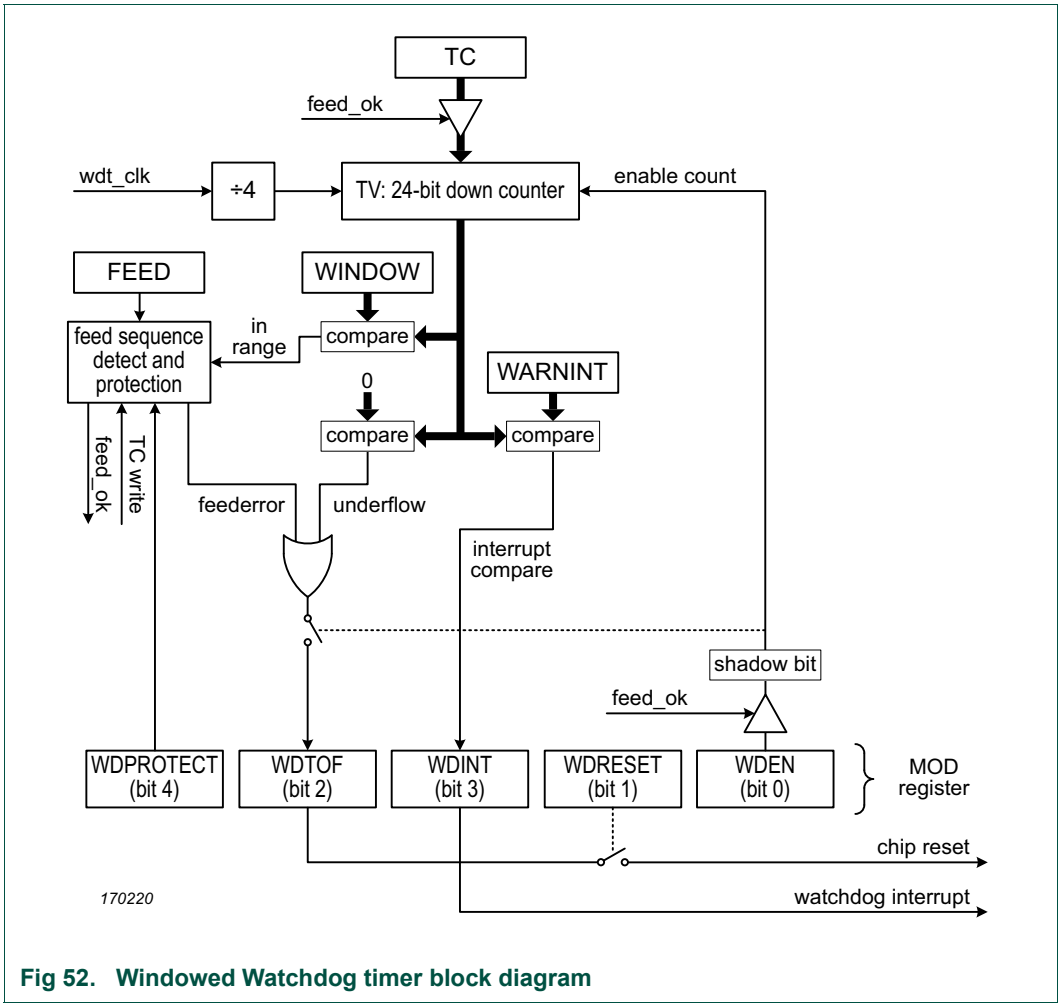


Fig 52. Windowed Watchdog timer block diagram

20.5.2 Clocking and power control

The watchdog timer block uses two clocks: APB bus clock and WDCLK. The APB bus clock is used for the APB accesses to the watchdog registers and is derived from the system clock (see Figure 10). The WDCLK is used for the watchdog timer counting and is derived from the watchdog oscillator.

The synchronization logic between the two clock domains works as follows: When the MOD and TC registers are updated by APB operations, the new value will take effect in 3 WDCLK cycles on the logic in the WDCLK clock domain.

When the watchdog timer is counting on WDCLK, the synchronization logic will first lock the value of the counter on WDCLK and then synchronize it with the APB bus clock, so that the CPU can read the TV register.

Remark: Because of the synchronization step, software must add a delay of three WDCLK clock cycles between the feed sequence and the time the MOD[WDPROTECT] is enabled. The length of the delay depends on the selected watchdog clock WDCLK.

20.5.3 Using the WWDT protect features

If MOD[WDPROTECT] is set, the watchdog time-out value (TC) can be changed only after the counter is below the value of WARNINT and WINDOW.

The reload overwrite lock mechanism can only be disabled by a reset of any type.

20.6 Functional description

The following figures illustrate several aspects of Watchdog Timer operation.

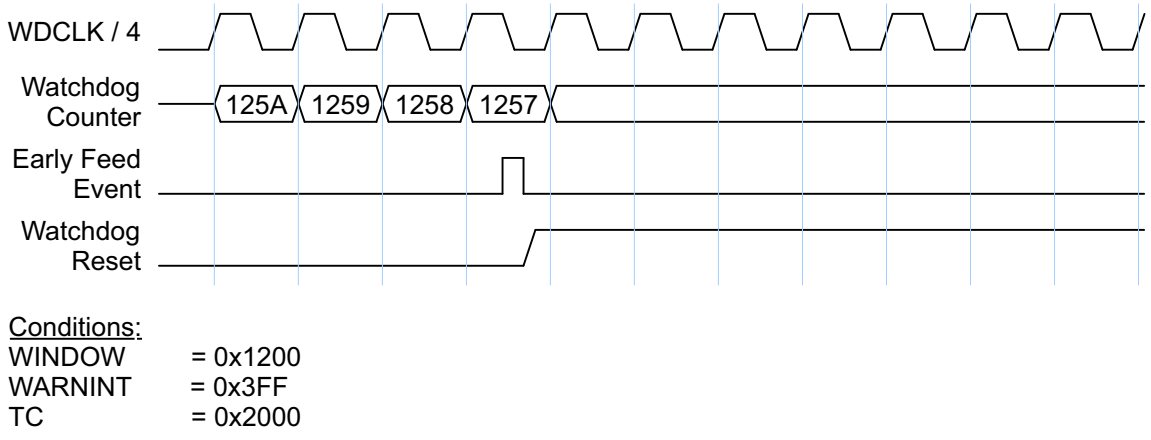


Fig 53. Early watchdog feed with windowed mode enabled

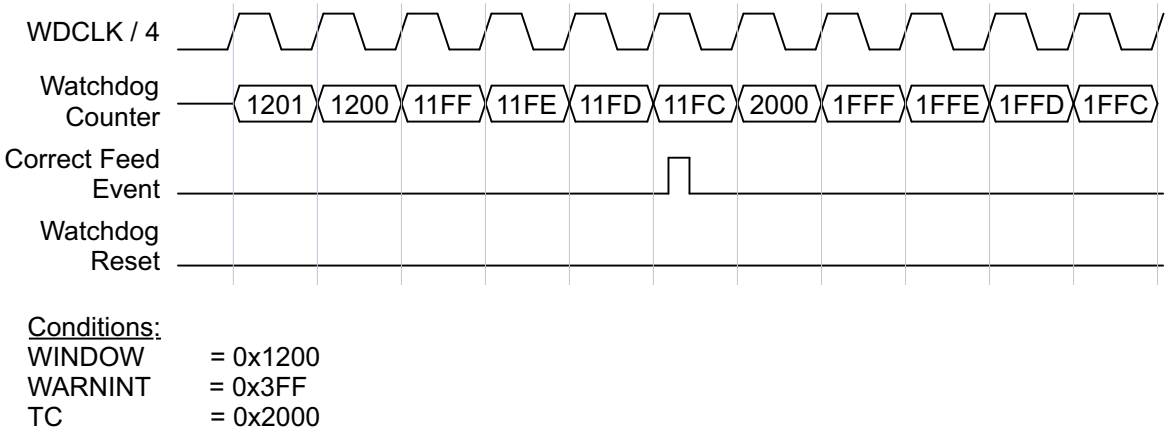


Fig 54. Correct watchdog feed with windowed mode enabled

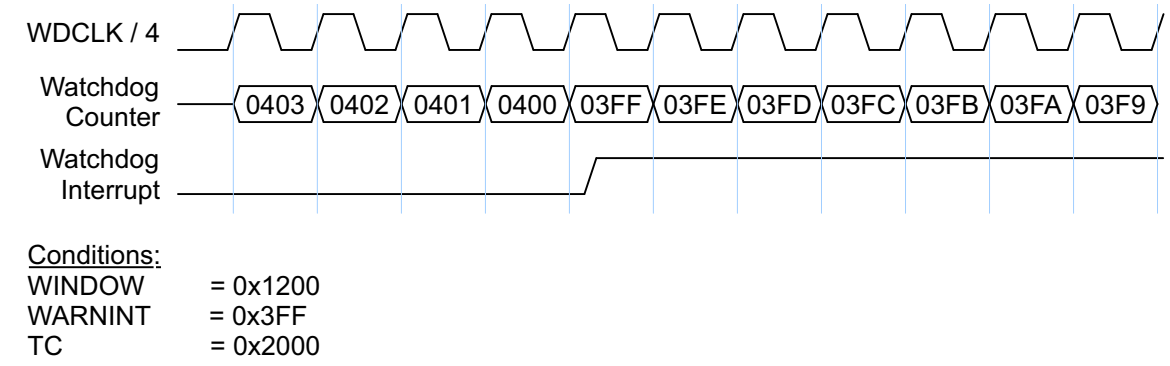


Fig 55. Watchdog warning interrupt

In all power modes except deep power-down mode, a Watchdog reset or interrupt can occur when the watchdog is running and has an operating clock source. The watchdog oscillator can be configured to keep running in sleep and deep-sleep modes.

If a watchdog interrupt occurs in sleep or deep-sleep mode, and the WWDT interrupt is enabled in the NVIC, the device will wake up. Note that in deep-sleep mode, the WWDT interrupt must be enabled in the SYSCON_STARTER0[WDT_BOD] in addition to the NVIC.

Table 38. Watchdog operating modes selection

WDEN	WDRESET	Mode of operation
0	X (0 or 1)	Debug/Operate without the Watchdog running.
1	0	Watchdog interrupt mode: the watchdog warning interrupt will be generated but watchdog reset will not. When this mode is selected, the watchdog counter reaching the value specified by WARNINT will set the WDINT flag and the Watchdog interrupt request will be generated
1	1	Watchdog reset mode: both the watchdog interrupt and watchdog reset are enabled. When this mode is selected, the watchdog counter reaching the value specified by WARNINT will set the WDINT flag and the watchdog interrupt request will be generated, and the watchdog counter reaching zero will reset the microcontroller. A watchdog feed prior to reaching the value of WINDOW will also cause a watchdog reset.

20.7 Software control

To use the functionality of the WWDT module it is recommended to use software functions from within fsl_wwdt.c.

21.1 How to read this chapter

The RTC is available on all JN5189(T)/JN5188(T) devices.

21.2 Features

- The RTC is driven by either the 32K FRO or XTAL
- From the selected 32 kHz clock lower speed clocks are generated:
  - 1 Hz clock for RTC timing.
  - 1 kHz clock for high-resolution RTC timing.
- Alarm timer: 32-bit, 1 Hz RTC counter and associated match register for alarm generation.
- Wake timer: separate 16-bit high-resolution/wake-up timer clocked at 1 kHz for 1 ms resolution; allowing for a maximum nominal period of 65 seconds.
- RTC Wake Timer can generate an interrupt for the CPU.
- Either Wake timer or the Alarm timer can wake up the part from sleep, deep sleep and power down modes.

21.3 Basic configuration

Configure the RTC as follows:

- Use the SYSCON_AHBCLKCTRL0[RTC] to enable the clock to the RTC register interface and peripheral clock.
- Enable the 32K clock source to be used, see Section 6.3 “Clock generation (CLK_GEN) module”
- Use the SYSCON_OSC32CLKSEL register to select the required 32 kHz clock source
- Configure the RTC clock divider to divide by 32 in order to generate the required 1 kHz clock
- For RTC software reset, use the CTRL register. The RTC is reset only by initial power-up of the device or when an RTC software reset is applied, it is not initialized by other system resets.
- The RTC provides an interrupt to the NVIC for the RTC_WAKE function.
- To enable the RTC interrupts for waking up from deep-sleep mode, keep the selected 32 kHz clock running in deep-sleep and enable the interrupts in the SYSCON_STARTER0 register and the NVIC.
- To enable the RTC interrupts for waking up from power down mode, keep the selected 32 kHz clock running in power down mode and enable the wakeup interrupt in the SYSCON_STARTER0 register.

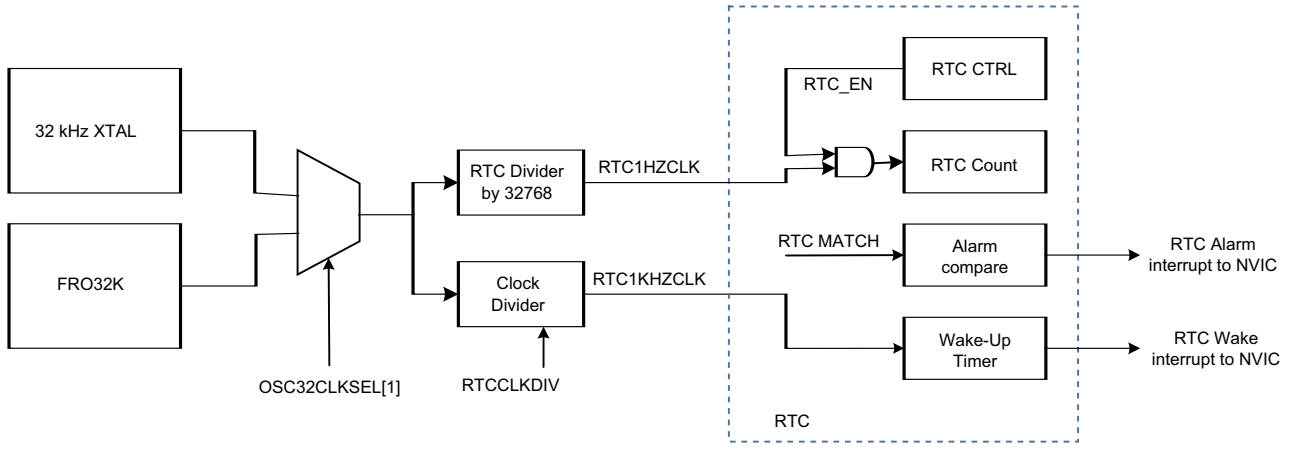


Fig 56. RTC clocking

21.3.1 RTC timers

The RTC contains two timers:

1. The main RTC timer. This 32-bit timer uses a 1 Hz clock and is intended to run continuously as a real-time clock. When the timer value reaches a match value, an interrupt is raised. The alarm interrupt can also wake up the part from sleep, deep-sleep and power-down modes if enabled.
2. The high-resolution/wake-up timer. This 16-bit timer uses a 1 kHz clock and operates as a one-shot down timer. Once the timer is loaded, it starts counting down to 0 at which point an interrupt is raised. The interrupt can wake up the part from sleep, deep-sleep and power down modes if enabled. This timer is intended to be used for timed wake-up from deep-sleep or power-down modes. The high-resolution wake-up timer can be disabled to conserve power if not used.

21.4 General description

21.4.1 Real-time clock

The real-time clock is a 32-bit up-counter which can be cleared or initialized by software. Once enabled, it counts continuously at a 1 Hz clock rate as long as the device is powered up and the RTC remains enabled.

The main purpose of the RTC is to count seconds and generate an alarm interrupt to the processor whenever the counter value equals the value programmed into the associated 32-bit MATCH register.

If the part is in one of the reduced-power modes (deep-sleep, power-down) an RTC alarm interrupt can also wake up the part to exit the power mode and begin normal operation.

21.4.2 High-resolution/wake-up timer

The time interval required for many applications, including waking the part up from a low-power mode, will often demand a greater degree of resolution than the one-second

minimum interval afforded by the main RTC counter. For these applications, a higher frequency secondary timer has been provided.

This secondary timer is an independent, stand-alone wake-up or general-purpose timer for timing intervals of up to 64 seconds with approximately one millisecond of resolution.

The High-Resolution/Wake-up Timer is a 16-bit down counter which is clocked at a 1 kHz rate when it is enabled. Writing any non-zero value to this timer will automatically enable the counter and launch a countdown sequence. When the counter is being used as a wake-up timer, this write can occur just prior to entering a reduced power mode.

When a starting count value is loaded, the High-Resolution/Wake-up Timer will turn on, count from the pre-loaded value down to zero, generate an interrupt and/or a wake-up command, and then turn itself off until re-launched by a subsequent software write.

21.5 Functional Description

The RTC is controlled from the CTRL register controls the clock enables within the block and the wakeup source at the block level.

The 1 Hz timer will generate an alarm when its 32-bit value equals the Match value in the MATCH register. The counter value can be read and configured using the COUNT register.

The 1 kHz 16-bit timer is configured and read through the WAKE register.

21.6 Software control

To use the functionality of the RTC module it is recommended to use software functions from within fsl_rtc.c.

22.1 How to read this chapter

The system tick timer (SysTick timer) is present on all JN5189(T)/JN5188(T) devices.

Refer to “Cortex-M4 TRM” for full details of the functionality and registers of this block.

22.2 Basic configuration

Configuration of the system tick timer is accomplished as follows:

1. Pins: The system tick timer uses no external pins.
2. Power: The system tick timer is enabled through the SysTick control register. The system tick timer clock is fixed to half the frequency of the system clock.
3. Enable the clock source for the SysTick timer in the SYST_CSR register and configuring the systick clock divider if required.

22.3 Features

- Simple 24-bit timer.
- Uses dedicated exception vector.
- Clocked internally by the system clock or the SYSTICKCLK.

22.4 General description

Block diagram of the SysTick timer for the CPU is shown in Figure 57.

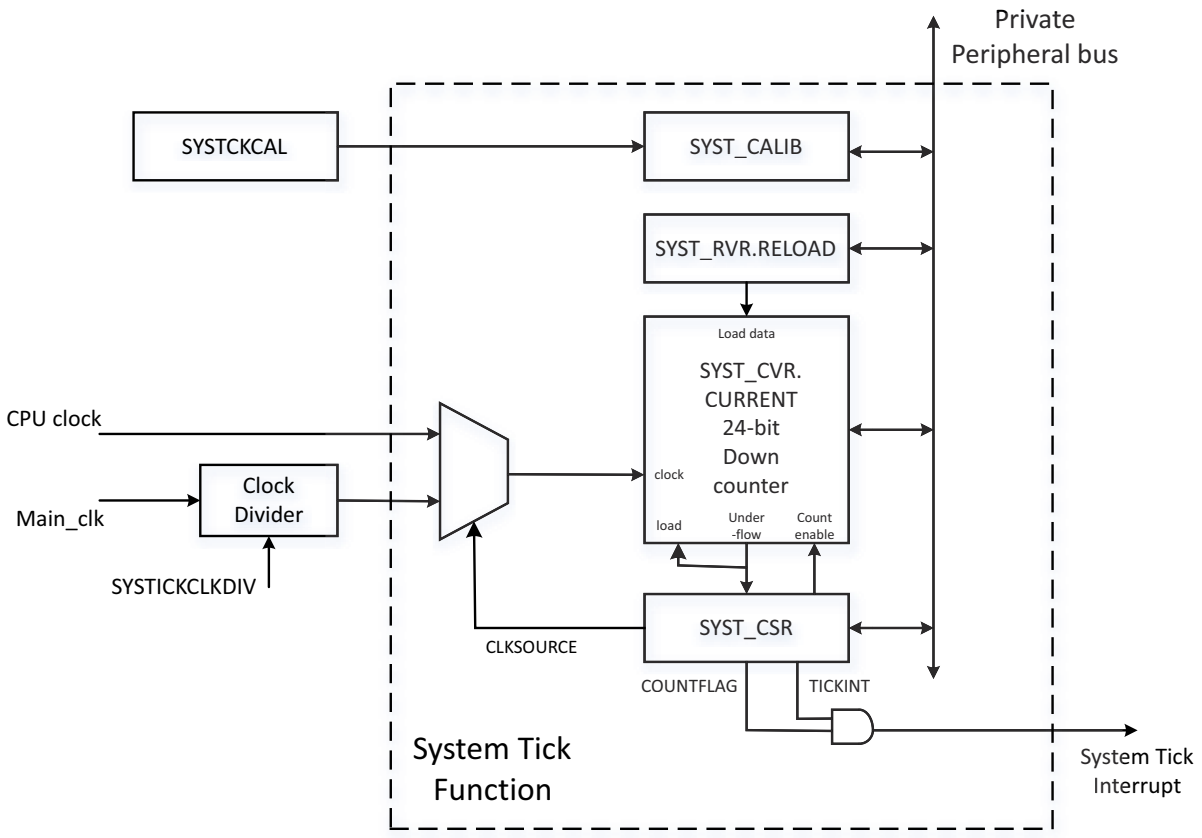


Fig 57. System tick timer block diagram

The SysTick timer is an integral part of the Cortex-M4. The SysTick timer is used to generate a fixed 10 ms interrupt for use by an operating system or other system management software.

Since the SysTick timer is a part of the CPU, it facilitates porting of software by providing a standard timer that is available on Arm Cortex-based devices. The SysTick timer can be used for:

- An RTOS tick timer which fires at a programmable rate (for example 100 Hz) and invokes a SysTick routine.
- A high-speed alarm timer using the core clock.
- A simple counter. Software can use this to measure time to completion and time used.
- An internal clock source control based on missing/meeting durations. The COUNTFLAG bit-field in the SYST_CSR control and status register can be used to determine if an action completed within a set duration, as part of a dynamic clock management control loop.

Refer to Ref. 1 “Cortex-M4 TRM” for full details of the functionality and registers of this block.

22.5 Functional description

The SysTick timer is a 24-bit timer that counts down to zero and generates an interrupt. The intent is to provide a fixed 10 millisecond time interval between interrupts. The SysTick timer is clocked from the CPU clock (the system clock, see Figure 57) [SYST_CSR.CLKSOURCE = 1] or from a generated clock [SYST_CSR.CLKSOURCE =0].

The generated clock is the output of the SYSTICK clock divider. Therefore its frequency depends on Main_Clk frequency and the clock divider settings See Figure 11 “CPU SYSTICK and TRACECLK clock generation” for details of the generation of the SYSTICK clock..

In order to generate recurring interrupts at a specific interval, the SYST_RVR register must be initialized with the correct value for the desired interval.

The Sys Tick register, SYST_CALIB, is a read only register. The field TENMS can be used to indicate the number of ticks needed for a 10 ms period. This field will read 0 until the SYST_CALIB values are written. To set this value write the required value to SYSCON_SYSTCKCAL register. This value is based on the clock settings previously described. SYST_RVR.RELOAD field should be set to the number of ticks that will be counted. Setting a value of 0 will cause the systick interrupt handler (void SysTick_Handler(void) to never fire.

The two further fields in SYST_CALIB are also driven from the SYSTCKCAL registers using the SKEW and NOREF fields.

22.6 Example timer calculations

To use the system tick timer, do the following:

1. Program the SYST_RVR register with the reload value calculated as shown below to obtain the desired time interval.
2. Clear the SYST_CVR register by writing to it. This ensures that the timer will count from the SYST_RVR value rather than an arbitrary value when the timer is enabled.

The following examples illustrate selecting SysTick timer reload values for different system configurations. All of the examples calculate an interrupt interval of 10 milliseconds, as the SysTick timer is intended to be used, and there are no rounding errors.

System tick timer clock = 24 MHz, System clock = 48 MHz

Program the SYST_CSR register with the value 0x3 which selects the clock from the system tick clock divider as the clock source and enables the SysTick timer and the SysTick timer interrupt. Use DIV of the SYSCON_SYSTICKCLKDIV setting to divide the system clock by 2 to create the 24 MHz clock for the SysTick timer.

SYST_RVR = (system tick timer clock frequency $\times$ 10 ms) -1 = (24 MHz $\times$ 10 ms) -1 = 240000-1 = 239999 = 0x0003 A97F

System clock = 12 MHz

Program the SYST_CSR register with the value 0x7 which selects the system clock as the clock source and enables the SysTick timer and the SysTick timer interrupt.

In this case the system clock is derived from the FRO 12 MHz clock.

SYST_RVR = (system clock frequency × 10 ms) −1 = (12 MHz × 10 ms) −1 = 120000 −1 = 119999 = 0x0001 D4BF

23.1 How to read this chapter

Two USART functions are available on all JN5189(T)/JN5188(T) devices.

23.2 Features

- 7, 8, or 9 data bits and 1 or 2 stop bits.
- Synchronous mode with master or slave operation. Includes data phase selection and continuous clock option.
- Multiprocessor/multidrop (9-bit) mode with software address compare.
- RS-485 transceiver output enable.
- Parity generation and checking: odd, even, or none.
- Software selectable oversampling from 5 to 16 clocks in asynchronous mode.
- One transmit and one receive data buffer.
- The USART function supports separate transmit and receive FIFO with 4 entries each.
- RTS/CTS for hardware signaling for automatic flow control. Software flow control can be performed using Delta CTS detect, Transmit Disable control, and any GPIO as an RTS output.
- Break generation and detection.
- Receive data is 2 of 3 sample "voting". Status flag set when one sample differs.
- Built-in Baud Rate Generator.
- Auto-baud mode for automatic baud rate detection.
- Special operating mode allows operation at up to 9600 baud using the 32 kHz RTC oscillator as the USART clock. This mode can be used while the device is in deep-sleep mode and can wake-up the device when a character is received.
- A fractional rate divider is shared among all USARTs.
- Interrupts available for FIFO receive level reached, FIFO transmit level reached, FIFO overflow or underflow, Transmitter Idle, change in receiver break detect, Framing error, Parity error, Delta CTS detect, and receiver sample noise detected (among others).
- USART transmit and receive functions can operated with the system DMA controller.
- Loopback mode for testing of data and flow control.

23.3 Basic configuration

Initial configuration of a USART peripheral is accomplished as follows:

- If needed, use the SYSCON_PRESETCTRL1 register to reset the USART0 or USART1 Interface.

- Enable the USART function by writing to the PSELID register of the related USART Interface.
- Configure the FIFOs for operation.
- Configure USART for receiving and transmitting data:
  - In the SYSCON_AHBCLKCTRL1 register, set the appropriate bit for the related USART Interface in order to enable the clock to the register interface.
  - Enable or disable the related USART Interface interrupt in the NVIC (see Table 17).
  - Configure the related USART Interface pin functions via IOCON, see Chapter 12.
  - Configure the USART Interface clock and USART baud rate. See Section 23.3.1.
- Configure the USART to wake up the part from low-power modes. See Section 23.3.2.
- Configure the USART to receive and transmit data in synchronous slave mode. See Section 23.3.2.

23.3.1 Configure the USART Interface clock and USART baud rate

Each USART Interface has a separate clock selection, which can include a shared fractional divider (also see Section 23.6.2.3 “32 kHz mode”). The function clock and the fractional divider for the baud rate calculation are set up in the SYSCON block as follows:

1. If a fractional value is needed to obtain a particular baud rate, program the fractional rate divider (FRG, controlled by SYSCON_FRGCTRL). The fractional divider value is the fraction of MULT/DIV. The MULT and DIV values are programmed in the SYSCON_FRGCTRL register. The DIV value must be programmed with the fixed value of 256.

   USART Interface clock = (FRG input clock) / (1+(MULT / DIV))

   The following rules apply for MULT and DIV:

   - Always set DIV to 256 by programming the FRGCTRL register with the value of 0xFF.
   - Set the MULT to any value between 0 and 255.
2. In asynchronous mode: configure the baud rate divider by BRG[BRGVAL]. The baud rate divider divides the USART Interface function clock (FCLK) to create the clock needed to produce the desired baud rate.

   Generally: baud rate = [FCLK / oversample rate] / BRG divide

   With specific register values: baud rate = [FCLK / (OSRVAL+1)] / (BRGVAL + 1)

   Generally: BRG divide = [FCLK / oversample rate] / baud rate

   With specific register values: BRGVAL = [[FCLK / (OSRVAL + 1)] / baud rate] - 1

   See Section 23.6.2.2 “Baud Rate Generator (BRG)”.
3. In synchronous master mode: The serial clock is Un_SCLK = FCLK / (BRGVAL+1).

The USART can also be clocked by the 32 kHz RTC oscillator. Set the CFG[MODE32K] bit to enable this 32 kHz mode. See also Section 23.6.2.3 “32 kHz mode”.

For details on the clock configuration see:

Section 23.6.2 “Clocking and baud rates”

23.3.2 Configure the USART for wake-up

A USART can wake up the system from sleep mode in asynchronous or synchronous mode on any enabled USART interrupt.

In deep-sleep mode, there are two options for configuring USART for wake-up:

- If the USART is configured for synchronous slave mode, the USART block can create an interrupt on a received signal even when the USART block receives no on-chip clocks - that is in deep-sleep mode.

  As long as the USART receives a clock signal from the master, it can receive up to one byte in the FIFORD[RXDAT] while in deep-sleep mode. Any interrupt raised as part of the receive data process can then wake up the part.
- If the 32 kHz mode is enabled, the USART can run in asynchronous mode using the 32 kHz RTC oscillator and create interrupts.

23.3.2.1 Wake-up from sleep mode

- Configure the USART in either asynchronous mode or synchronous mode.
- Enable the USART interrupt in the NVIC.
- Any enabled USART interrupt wakes up the part from sleep mode.

23.3.2.2 Wake-up from deep-sleep mode

- Configure the USART in synchronous slave mode. The SCLK function must be connected to a pin and also connect the pin to the master. Alternatively, the 32 kHz mode can be enabled and the USART operated in asynchronous mode with the 32 kHz RTC oscillator.
- Enable the USART interrupt in the SYSCON_STARTER0 register.
- Enable the USART interrupt in the NVIC.
- The USART wakes up the part from deep-sleep mode on all events that cause an interrupt and are enabled. Typical wake-up events are:
  - A start bit has been received.
  - Received data becomes available.
  - In synchronous mode, data is available in the FIFO to be transmitted, and a serial clock from the master has been received.
  - A change in the state of the CTS pin if the CTS function is connected.

    Remark: By enabling or disabling specific USART interrupts, you can customize when the wake-up occurs.

23.3.2.3 Wake-up from power down mode

Only the USART0 is possible to wake from power down mode.

- Configure the USART in synchronous slave mode. The SCLK function must be connected to a pin and also connect the pin to the master. Alternatively, the 32 kHz mode can be enabled and the USART operated in asynchronous mode with the 32 kHz RTC oscillator
- Enable the USART interrupt in the SYSCON_STARTER0 register

- The USART wakes up the part from power down mode on all events that cause an interrupt and are enabled. Typical wake-up events are:
  - A start bit is received.
  - received data becomes available. In synchronous mode, data is available in the FIFO to be transmitted, and a serial clock from the master has been received.
  - A change in the state of the CTS pin if the CTS function is connected
- Ensure that the Comm0 power domain will be active during the power cycle, using the power control APIs.
- Perform power down request. Note, the Low Power API should be used to perform the configuration and execution of power down cycles.

23.4 Pin description

The USART receive, transmit, and control signals are assigned to external pins through via IOCON. See the IOCON description (Chapter 12) to assign the USART functions to pins on the device package.

Table 39. USART pin description

Pin	Type	Name used in Pin Configuration chapter	Description
TXD	O	USARTn_TXD	Transmitter output for USART Interface n. Serial transmit data.
RXD	I	USARTn_RXD	Receiver input for USART Interface n. Serial receive data.
$\overline{\text{RTS}}$	O	USART0_RTS	Request To Send output for USART0 Interface. This signal supports inter-processor communication through the use of hardware flow control. This signal can also be configured to act as an output enable for an external RS-485 transceiver. RTS is active when the USART RTS signal is configured to appear on a device pin.
$\overline{\text{CTS}}$	I	USART0_CTS	Clear To Send input for USART0 Interface. Active low signal indicates that the external device that is in communication with the USART is ready to accept data. This feature is active when enabled by the CFG[CTSEN] bit and when configured to appear on a device pin. When deasserted (high) by the external device, the USART will complete transmitting any character already in progress, then stop until CTS is again asserted (low).
SCLK	I/O	USARTn_SCK	Serial clock input/output for USART Interface n in synchronous mode. Clock input or output in synchronous mode. **Remark:** When the USART is configured as a master, such that SCK is an output, it must actually be connected to a pin in order for the USART to work properly.

Recommended IOCON settings are shown in Table 40. See Chapter 12 for definitions of pin types.

Table 40: Suggested USART pin setting for standard GPIO IO

IOCON bit(s)	Field	Setting	Note
2:0	Func	Set for USART function	
4:3	Mode	2	No pullup or pull-down
5	Slew0	0	see slew1
6	Invert	0	No need to invert
7	Digimode	1	Digital mode

Table 40: Suggested USART pin setting for standard GPIO IO

IOCON bit(s)	Field	Setting	Note
8	FilterOff	1	Generally disable filter
9	Slew1	0	With slew0: generally set to 0. Settings to 1,2 or 3 at high usary rates may improve performance
10	OD	0	Normal GPIO mode
11	IO_clamp	0	Clamp Off

Table 41: Suggested USART pin setting for combo IO cells (PIO10 and PIO11)

IOCON bit(s)	Field	Setting	Note
2:0	Func	Set for USART function	
3	EGP	1	GPIO mode
4	ECS	0	Standard GPIO mode
5	EHS	0	Low speed GPIO. Setting to 1 may be beneficial with higher speed USART signals.
6	Invert	0	No need to invert
7	Digimode	1	Digital mode
8	FilterOff	1	Generally disable filter
9	Fsel	0	Not valid for GPIO mode
10	OD	0	Normal GPIO mode
11	Reserved	0	Entry should be blank
12	IO_clamp	0	Clamp Off

Software configuration of the IO cell setting can be achieved with the IOCON_PinMuxSet or IOCON_SetPinMuxing function. As an example for using USART RX and TX pins on PIO8 and PIO9 then the following SW functions could be used. To have USART on these IOs requires the setting of Function value of 2:

```
IOCON_PinMuxSet(IOCON, 0, 8,
IOCON_MODE_INACT |
IOCON_FUNC2 |
IOCON_DIGITAL_EN);
IOCON_PinMuxSet(IOCON, 0, 9,
IOCON_MODE_INACT |
IOCON_FUNC2 |
IOCON_DIGITAL_EN);
```

23.5 General description

The USART receiver block monitors the serial input line, Un_RXD, for valid input. The receiver shift register assembles characters as they are received, after which they are passed to the receiver FIFO to await access by the CPU or DMA controller.

The USART transmitter block accepts data written by the CPU or DMA controller to the transmit FIFO. When the transmitter is available, the transmit shift register takes that data, formats it, and serializes it to the serial output, Un_TXD.

The Baud Rate Generator block divides the incoming clock to create an oversample clock (typically 16x) in the standard asynchronous operating mode. The BRG clock input source is the shared Fractional Rate Generator that runs from the USART function clock. The 32 kHz operating mode generates a specially timed internal clock based on the RTC oscillator frequency.

In synchronous slave mode, data is transmitted and received using the serial clock directly. In synchronous master mode, data is transmitted and received using the baud rate clock without division.

Status information from the transmitter and receiver is provided via the STAT register. Many of the status flags are able to generate interrupts, as selected by software. The INTSTAT register provides a view of all interrupts that are both enabled and pending.

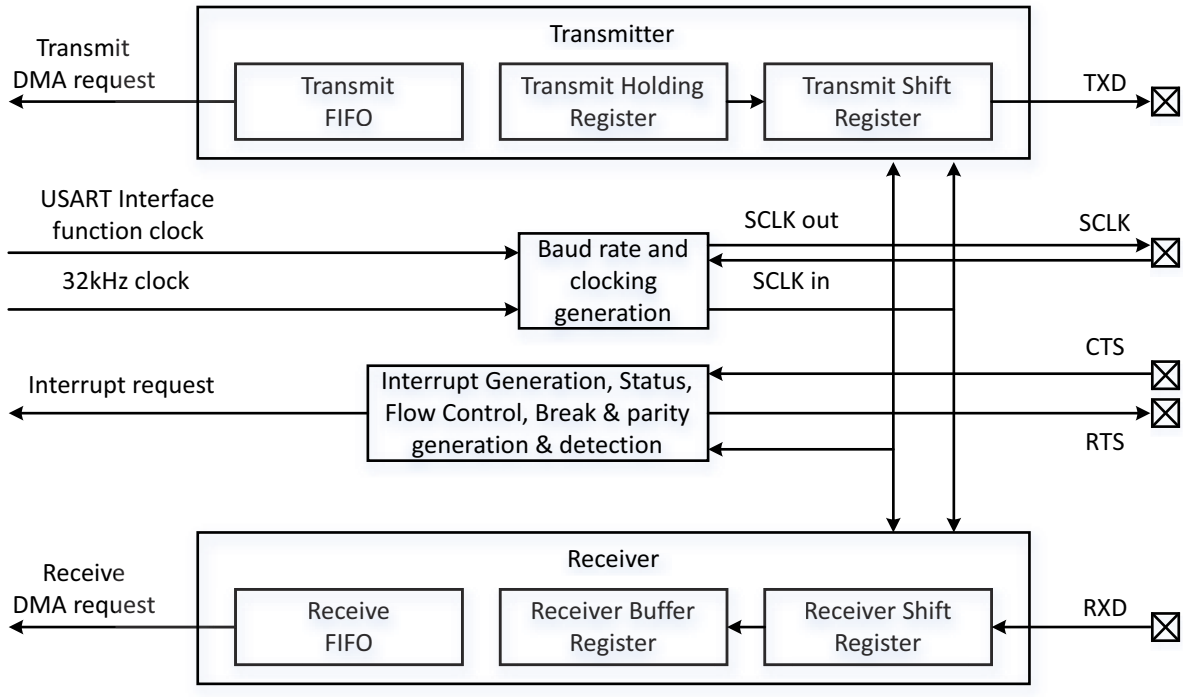


Fig 58. USART block diagram

23.6 Functional description

23.6.1 AHB bus access

The bus interface to the USART registers contained in the USART Interface supports only word writes. Byte and halfword writes are not supported in conjunction with the USART function.

23.6.2 Clocking and baud rates

In order to use the USART, clocking details must be defined such as setting up the clock source selection, the BRG, and setting up the FRG if it is the selected clock source.

Also see Section 23.3.1 “Configure the USART Interface clock and USART baud rate”.

23.6.2.1 Fractional Rate Generator (FRG)

The Fractional Rate Generator can be used to obtain more precise baud rates when the function clock is not a good multiple of standard (or otherwise desirable) baud rates.

The FRG is typically set up to produce an integer multiple of the highest required baud rate, or a very close approximation. The BRG is then used to obtain the actual baud rate needed.

The FRG register controls the Fractional Rate Generator, which provides the base clock that may be used by any USART Interface. The Fractional Rate Generator creates a lower rate output clock by suppressing selected input clocks. When not needed, the value of 0 can be set for the FRG, which will then not divide the input clock.

The FRG output clock is defined as the input clock divided by 1 + (MULT / 256), where MULT is in the range of 1 to 255. This allows producing an output clock that ranges from the input clock divided by 1+1/256 to 1+255/256 (just more than 1 to just less than 2). Any further division can be done specific to each USART block by the integer BRG divider contained in each USART.

The base clock produced by the FRG cannot be perfectly symmetrical, so the FRG distributes the output clocks as evenly as is practical. Since USARTs normally uses 16x overclocking, the jitter in the fractional rate clock in these cases tends to disappear in the ultimate USART output.

For setting up the fractional divider, see SYSCON_FRGCTRL register and Section 23.3.1 “Configure the USART Interface clock and USART baud rate”.

23.6.2.2 Baud Rate Generator (BRG)

The Baud Rate Generator (see BRG register) is used to divide the base clock to produce a rate 16 times the desired baud rate. Typically, standard baud rates can be generated by integer divides of higher baud rates.

Note that in 32 kHz mode, the baud rate generator is still used and must be set to 0 if 9600 baud is required.

Remark: In order to change a baud rate after a USART is running, the following sequence should be used:

1. Make sure the USART is not currently sending or receiving data.
2. Disable the USART by writing a 0 to the CFG[ENABLE] (0 may be written to the entire register).
3. Write the new BRG[BRGVAL].
4. Write 1 to the CFG[ENABLE].

23.6.2.3 32 kHz mode

In order to use a 32 kHz clock to operate a USART at any reasonable speed, a number of adaptations need to be made. First, 16x overclocking has to be abandoned. Otherwise, the maximum data rate would be very low. For the same reason, multiple samples of each data bit must be reduced to one. Finally, special clocking has to be used for individual bit times because 32 kHz is not particularly close to an even multiple of any standard baud rate.

When 32 kHz mode is enabled, clocking must come from the 32k XTAL. Hence the XTAL must be enabled and running; also SYSCON_OSC32CLKSEL[SEL32KHZ] must be set. The 32 kHz clock must be enabled to the USART using SYSCON_MODEMCTRL[BLE_LP_OSC32K_EN]. The FRG is bypassed, and the BRG can be used to divide down the default 9600 baud to lower rates. Other adaptations required to make the USART work for rates up to 9600 baud are done internally. Rate error will be less than one half percent in this mode, provided the XTAL is operating at the intended frequency of 32.768 kHz.

23.6.3 DMA

A DMA request is provided for each USART direction, and can be used in lieu of interrupts for transferring data by configuring the DMA controller and FIFO level triggering appropriately. The DMA controller provides an acknowledgement signal that clears the related request when it completes handling a that request. The transmitter DMA request is asserted when the transmitter can accept more data. The receiver DMA request is asserted when received data is available to be read.

When DMA is used to perform USART data transfers, other mechanisms can be used to generate interrupts when needed. For instance, completion of the configured DMA transfer can generate an interrupt from the DMA controller. Also, interrupts for special conditions, such as a received break, can still generate useful interrupts.

23.6.4 Synchronous mode

In synchronous mode, a master generates a clock as defined by the clock selection and BRG, which is used to transmit and receive data. As a slave, the external clock is used to transmit and receive data. There is no overclocking in either case.

23.6.5 Flow control

The USART0 supports hardware flow control. Both USART0 and USART1 support software flow control.

23.6.5.1 Hardware flow control

The USART supports hardware flow control using RTS and/or CTS signaling. If RTS is configured to appear on a device pin so that it can be sent to an external device, it indicates to an external device the ability of the receiver to receive more data. It can also be used internally to throttle the transmitter from the receiver, which can be especially useful if loopback mode is enabled.

If connected to a pin, and if enabled to do so, the CTS input can allow an external device to throttle the USART transmitter. Both internal and external CTS can be used separately or together.

Figure 59 shows an overview of RTS and CTS within the USART.

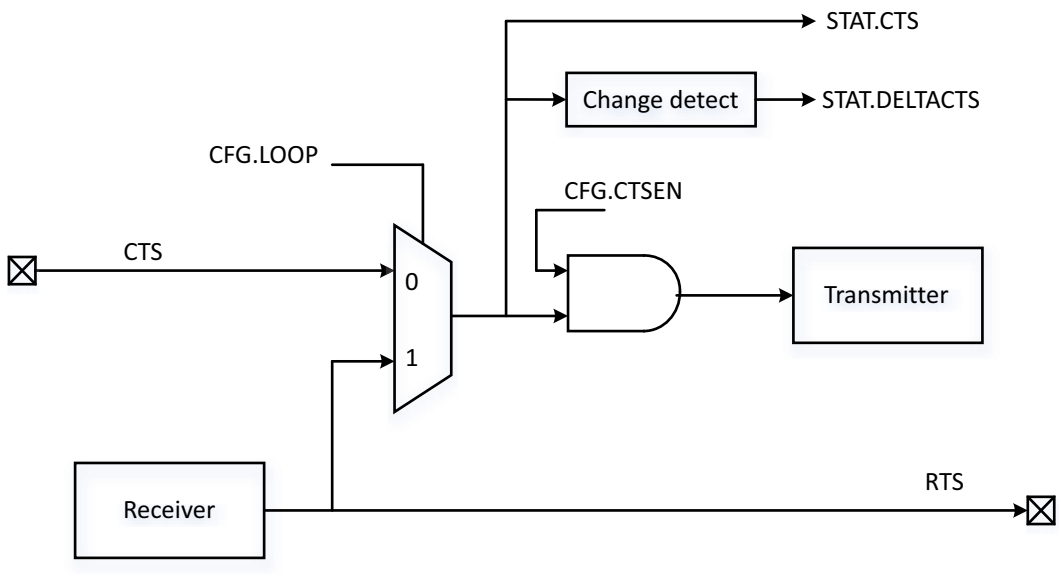


Fig 59. Hardware flow control using RTS and CTS

23.6.5.2 Software flow control

Software flow control could include XON / XOFF flow control, or other mechanisms. these are supported by the ability to check the current state of the CTS input, and/or have an interrupt when CTS changes state (via the STAT[CTS] and STAT[DELTACTS] bits), and by the ability of software to gracefully turn off the transmitter (via the CTL[TXDIS] bit).

23.6.6 Auto-baud function

The auto-baud functions attempts to measure the start bit time of the next received character. For this to work, the measured character must have a 1 in the least significant bit position, so that the start bit is bounded by a falling and rising edge. Before an auto-baud operation is requested, the BRG value must be set to 0. The measurement is made using the current clocking settings, including the oversampling configuration. The result is that a value is stored in the BRG register that is as close as possible to the correct setting for the sampled character and the current clocking settings. The sampled character is provided in the FIFORD[RXDATA], allowing software to double check for the expected character.

Auto-baud includes a time-out that is flagged by STAT[ABERR] if no character is received at the expected time. It is recommended that auto-baud only be enabled when the USART receiver is idle. Once enabled, either data will become available in the FIFO or ABERR will be asserted at some point, at which time software should turn off auto-baud.

Auto-baud has no meaning and should not be enabled when the USART is in synchronous mode.

23.6.7 RS-485 support

RS-485 support requires some form of address recognition and data direction control.

This USART has provisions for hardware address recognition (see the CFG[AUTOADDR] and the ADDR register), as well as software address recognition (see the CTL[ADDRDET] bit).

Automatic data direction control with the RTS pin can be set up using the CFG[OESEL], CFG[OEPOL], and CFG[[OETA] bits). Data direction control can also be implemented in software using a GPIO pin.

23.6.8 Oversampling

Typical industry standard USARTs use a 16x oversample clock to transmit and receive asynchronous data. This is the number of BRG clocks used for one data bit. The Oversample Select Register (OSR) allows this USART to use a 16x down to a 5x oversample clock. There is no oversampling in synchronous modes.

Reducing the oversampling can sometimes help in getting better baud rate matching when the baud rate is very high, or the function clock is very low. For example, the closest actual rate near 115,200 baud with a 12 MHz function clock and 16x oversampling is 107,143 baud, giving a rate error of 7%. Changing the oversampling to 15x gets the actual rate to 114,286 baud, a rate error of 0.8%. Reducing the oversampling to 13x gets the actual rate to 115,385 baud, a rate error of only 0.16%.

There is a cost for altering the oversampling. In asynchronous modes, the USART takes three samples of incoming data on consecutive oversample clocks, as close to the center of a bit time as can be done. When the oversample rate is reduced, the three samples spread out and occupy a larger proportion of a bit time. For example, with 5x oversampling, there is one oversample clock, then three data samples taken, then one more oversample clock before the end of the bit time. Since the oversample clock is running asynchronously from the input data, skew of the input data relative to the expected timing has little room for error. At 16x oversampling, there are several oversample clocks before actual data sampling is done, making the sampling more robust. Generally speaking, it is recommended to use the highest oversampling where the rate error is acceptable in the system.

23.6.9 Break generation and detection

A line break may be sent at any time, regardless of other USART activity. Received break is also detected at any time, including during reception of a character. Received break is signaled when the RX input remains low for 16 bit times. Both the beginning and end of a received break are noted by the STAT[DELTARXBRK] status flag, which can be used as an interrupt. See Section 23.6.10 for details of LIN mode break.

In order to avoid corrupting any character currently being transmitted, it is recommended that the USART transmitter be disabled by setting the CTL[TXDIS] bit, then waiting for the STST[TXDISSTAT] flag to be set prior to sending a break. Then a 1 may be written to the CTL[TXBRKEN] bit. This sends a break until TXBRKEN is cleared, allowing any length break to be sent.

23.6.10 LIN bus

The only difference between standard operation and LIN mode is that LIN mode alters the way that break generation and detection is performed (see Section 23.6.9 for details of the standard break). When a break is requested by setting the CTL[TXBRKEN], then sending a dummy character, a 13-bit time break is sent. A received break is flagged when the RX input remains low for 11-bit times. As for non-LIN mode, a received character is also flagged, and accompanied by a framing error status.

As a LIN slave, the auto-baud feature can be used to synchronize to a LIN sync byte, and will return the value of the sync byte as confirmation of success.

Wake-up for LIN can potentially be handled in a number of ways, depending on the system, and what clocks may be running in a slave device. For instance, as long as the USART is receiving internal clocks allowing it to function, it can be set to wake up the CPU for any interrupt, including a received start bit. If there are no clocks running, the GPIO function of the USART RX pin can be programmed to wake up the device.

23.7 Software control

To use the functionality of the USART module it is recommended to use software functions from within fsl_usart.c.

24.1 How to read this chapter

Two SPI functions (SPI0 and SPI1) are available on all JN5189(T)/JN5188(T) devices.

24.2 Features

- Master and slave operation.
- Data transmits of 4 to 16 bits supported directly.
- Larger frames supported by software.
- The SPI function supports separate transmit and receive FIFOs with 4 16-bit entries each.
- Supports DMA transfers: SPIn transmit and receive functions can be operated with the system DMA controller.
- Data can be transmitted to a slave without the need to read incoming data. This can be useful while setting up an SPI memory.
- Up to three Slave Select input/outputs, for SPI1; one slave select input/output for SPI0. Slave selects have selectable polarity.
- Options on assignment of SPI functions to IOs for flexible usage

Remark: Texas Instruments SSI and National Microwire modes are not supported.

24.3 Basic configuration

Initial configuration of an SPI peripheral is accomplished as follows:

- If needed, use the SYSCON_PRESETCTRL1 register to reset the SPI Interface that is about to be used.
- Configure PSELID register of the related SPI Interface.
- Configure the FIFOs for operation.
- Configure the SPI for receiving and transmitting data:
  - In the SYSCON_AHBCLKCTRL1 register, set the appropriate bit for the related SPI Interface in order to enable the clock to the register interface.
  - Enable or disable the related SPI Interface interrupts in the NVIC
  - Configure the required SPI Interface pin functions through IOCON.
  - Configure the SPI Interface clock and SPI data rate (see Section 24.6.4 “Clocking and data rates”).
  - Set the TXCTL[RXIGNORE] bit to only transmit data and not read the incoming data. Otherwise, the transmit halts when the FIFORD buffer is full.
- Enable the FIFO, then enable the SPI function.
- Configure the SPI function to wake up the part from low-power modes. See Section 24.3.1 “Configure the SPI for wake-up”.

24.3.1 Configure the SPI for wake-up

In sleep mode, any signal that triggers an SPI interrupt can wake the part, provided that the interrupt is enabled in the INTENSET register and the NVIC. As long as the SPI clock is configured to be active in sleep mode, the SPI can wake up the part independently of whether the SPI block is configured in master or slave mode.

In deep-sleep mode, the SPI clock is turned off. However, if the SPI is configured in slave mode and an external master on the provides the clock signal, the SPI can create an interrupt asynchronously and wake up the device. The appropriate interrupt(s) must be enabled in the SPI and the NVIC.

24.3.1.1 Wake-up from sleep mode

- Configure the SPI in either master or slave mode.
- Enable the SPI interrupt in the NVIC.
- Any enabled SPI interrupt wakes up the part from sleep mode.

24.3.1.2 Wake-up from deep-sleep mode

- Configure the SPI in slave mode. The SCK function must be connected to a pin and the pin connected to the slave.
- Enable the SPI interrupt in the SYSCON_STARTER0 register.
- Enable the SPI interrupt in the NVIC.
- Enable desired SPI interrupts. The following wake-up events are examples:
  - A change in the state of the SSEL pins.
  - Data available to be received.
  - Receive FIFO overflow.

24.3.1.3 Wake-up from power down

Only the SPI0 is possible to wake-up from power down mode.

- Configure the SPI in slave mode. The SCK function must be connected to a pin and the pin connected to the slave.
- Enable the SPI interrupt in the SYSCON_STARTER0 register
- Enable the desired SPI interrupts. The following wake-up events are examples:
  - A change in state of the SSEL pins
  - Data available to be received
  - Receive FIFO overflow
- Ensure that the Comm0 power domain will be active during the power cycle, using the power control APIs.
- Perform power down request. Note, the Low Power API should be used to perform the configuration and execution of power down cycles.

24.4 Pin description

The SPI interface signals are assigned to external pins via IOCON.

Table 42: SPI Pin Description

Function	Type	Pin name used in Pin Description chapter	Description
SCK	I/O	SPIn_SCK	Serial Clock for SPI on SPI Interface n. SCK is a clock signal used to synchronize the transfer of data. It is driven by the master and received by the slave. When the SPI interface is used, the clock is programmable to be active-high or active-low. SCK only switches during a data transfer. It is driven whenever the CFG[MASTER] = 1, regardless of the state of the Enable bit.
MOSI	I/O	SPIn_MOSI	Master Out Slave In for SPI on SPI Interface n. The MOSI signal transfers serial data from the master to the slave. When the SPI is a master, it outputs serial data on this signal. When the SPI is a slave, it clocks in serial data from this signal. MOSI is driven whenever the CFG[MASTER] equals 1, regardless of the state of the CFG[ENABLE] bit.
MISO	I/O	SPIn_MISO	Master In Slave Out for SPI on SPI Interface n. The MISO signal transfers serial data from the slave to the master. When the SPI is a master, serial data is input from this signal. When the SPI is a slave, serial data is output to this signal. MISO is driven when the SPI block is enabled, the CFG[MASTER] = 0, and when the slave is selected by one or more SSEL signals.
SSEL0	I/O	SPI0_SSELN or SPI1_SSELN0	Slave Select 0 for SPI on SPI Interface n. When the SPI interface is a master, it will drive the SSEL signals to an active state before the start of serial data and then release them to an inactive state after the serial data has been sent. By default, this signal is active low but can be selected to operate as active high. When the SPI is a slave, any SSEL in an active state indicates that this slave is being addressed. The SSEL pin is driven whenever the CFG[MASTER] = 1, regardless of the state of the CFG[ENABLE].
SSEL1	I/O	SPI1_SSELN1	Slave Select 1 for SPI on SPI Interface 1. This feature is not supported on SPI0.
SSEL2	I/O	SPI1_SSELN2	Slave Select 2 for SPI on SPI Interface 1. This feature is not supported on SPI0.

Recommended IOCON settings are shown in Table 44. See Chapter 12 for definitions of pin types.

The following table shows the pin locations that can be configured for SPI functionality. See IOCON configuration for FUNC settings associated with these mappings.

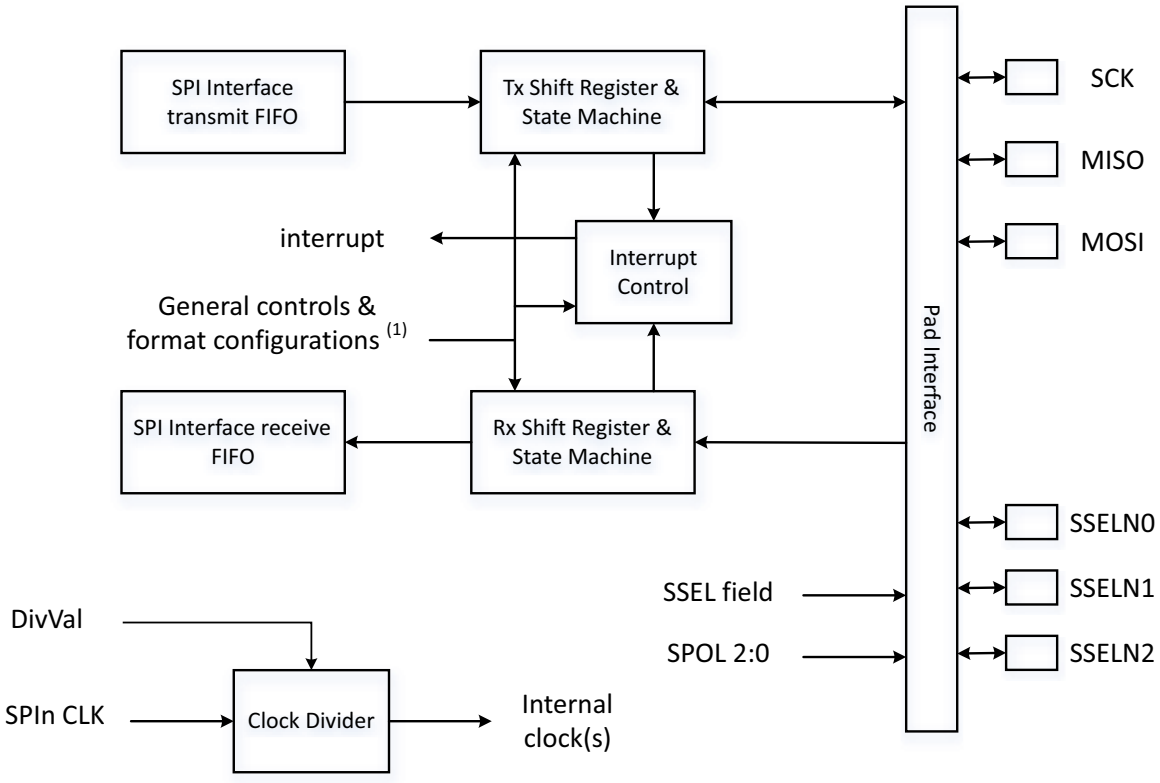
Table 43: Pins for SPI functionality

Pin	Possible Pin Assignments		
SPI0_SCK	GPIO2	GPIO6	GPIO10
SPI0_MISO	GPIO3	GPIO7	GPIO11
SPI0_MOSI	GPIO4	GPIO8	GPIO12
SPI0_SSELN	GPIO5	GPIO9	GPIO13
SPI1_SCK	GPIO15	GPIO0	
SPI1_MISO	GPIO18	GPIO5	GPIO1
SPI1_MOSI	GPIO17	GPIO2	
SPI1_SSELN0	GPIO16	GPIO3	
SPI1_SSELN1	GPIO14	GPIO4	
SPI1_SSELN2	GPIO13	GPIO5	

Table 44: Suggested SPI pin settings

IOCON bit (s)	Standard IO pin		GPIO10 or 11	
	Name	Setting	Name	Setting
12	—	—	IO_CLAMP	Set to 0, normal operation
11	IO_CLAMP	Set to 0, normal operation	SSEL	Set to 0
10	OD	Set to 0, unless open-drain output is desired	OD	Set to 0, unless open-drain output is desired
9	SLEW1	With SLEW0: Generally set to 0. Setting to 1,2,or 3 at higher SPI rates may improve performance	FSEL	Set to 0, unless input filtering is required
8	FILTEROFF	Generally set to 1	FILTEROFF	Generally set to 1
7	DIGIMODE	Set to 1	DIGIMODE	Set to 1
6	INVERT	Set to 0	INVERT	Set to 0
5	SLEW0	See SLEW1	EHS	Set to 0 unless at higher SPI rates when setting to 1 may improve performance
4	MODE[1]	With MODE[0]: Set to 0 (pull-down/pull-up resistor not enabled). Could be another setting if the input might sometimes be floating (causing leakage within the pin input).	ECS	Set to 0 when using standard GPIO mode
3	MODE[0]	See MODE[0]	EGP	Set to 1 for standard GPIO mode
2:0	FUNC[2:0]	Must select the correct function for this peripheral	FUNC[2:0]	Must select the correct function for this peripheral

24.5 General description



(1) Includes CPOL, CPHA, LSBF, LEN, master, enable, transfer_delay, frame_delay, pre_delay, post_delay, SOT, EOT, EOF, RXIGNORE and individual interrupt enables.

Fig 60. SPI block diagram

24.6 Functional description

24.6.1 AHB bus access

With the exception of the FIFOWR register, the bus interface to the SPI registers contained in the SPI Interface support only word writes. Byte and halfword writes are not supported in conjunction with the SPI function for those registers.

The FIFOWR register also supports byte and halfword (data only) writes in order to allow writing FIFO data without affecting the SPI control fields FIFOWR[31:16].

24.6.2 Operating modes: clock and phase selection

SPI interfaces typically allow configuration of clock phase and polarity. These are sometimes referred to as numbered SPI modes, as described in Table 45 and shown in Figure 61. CPOL and CPHA are configured by bits in the CFG register.

Table 45: SPI mode summary

CPOL	CPHA	SPI Mode	Description	SCK rest state	SCK data change edge	SCK data sample edge
0	0	0	The SPI captures serial data on the first clock transition of the transfer (when the clock changes away from the rest state). Data is changed on the following edge.	low	falling	rising
0	1	1	The SPI changes serial data on the first clock transition of the transfer (when the clock changes away from the rest state). Data is captured on the following edge.	low	rising	falling
1	0	2	Same as mode 0 with SCK inverted.	high	rising	falling
1	1	3	Same as mode 1 with SCK inverted.	high	falling	rising

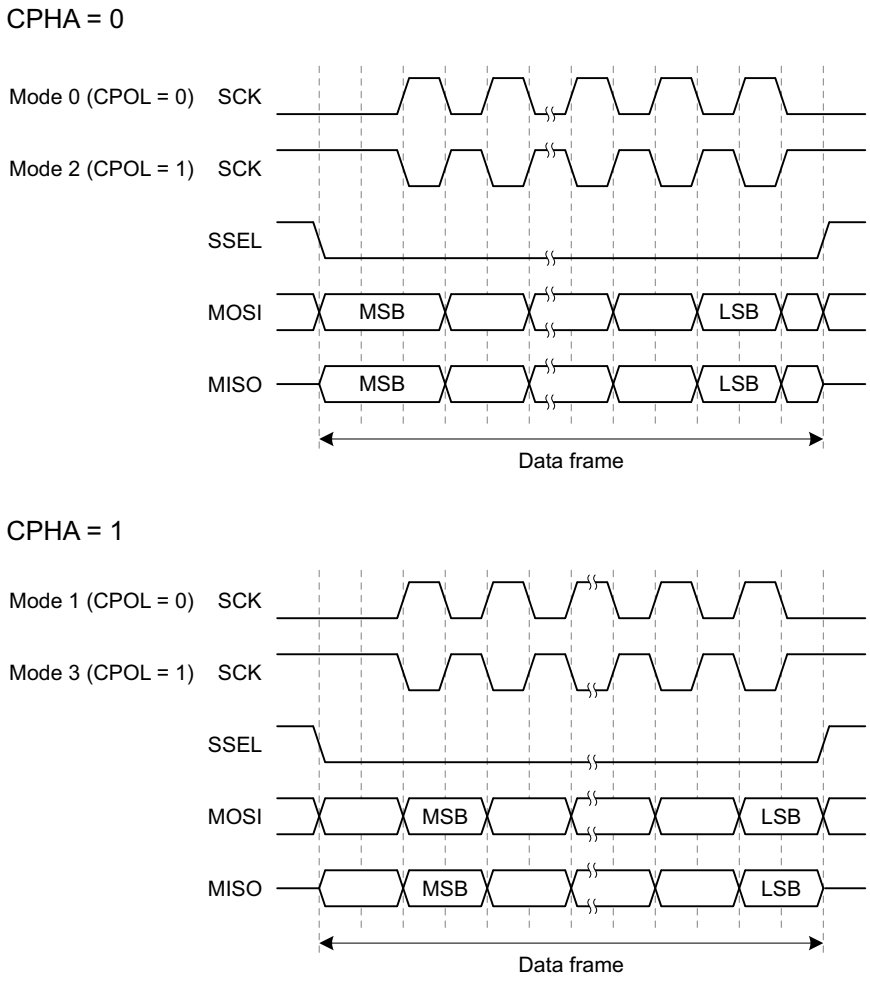


Fig 61. Basic SPI operating modes

24.6.3 Frame delays

Several delays can be specified for SPI frames. These include:

- Pre_delay: delay after SSEL is asserted before data clocking begins
- Post_delay: delay at the end of a data frame before SSEL is deasserted
- Frame_delay: delay between data frames when SSEL is not deasserted
- Transfer_delay: minimum duration of SSEL in the deasserted state between transfers

24.6.3.1 Pre_delay and Post_delay

Pre_delay and Post_delay are illustrated by the examples in Figure 62. The Pre_delay value controls the amount of time between SSEL being asserted and the beginning of the subsequent data frame. The Post_delay value controls the amount of time between the end of a data frame and the deassertion of SSEL.

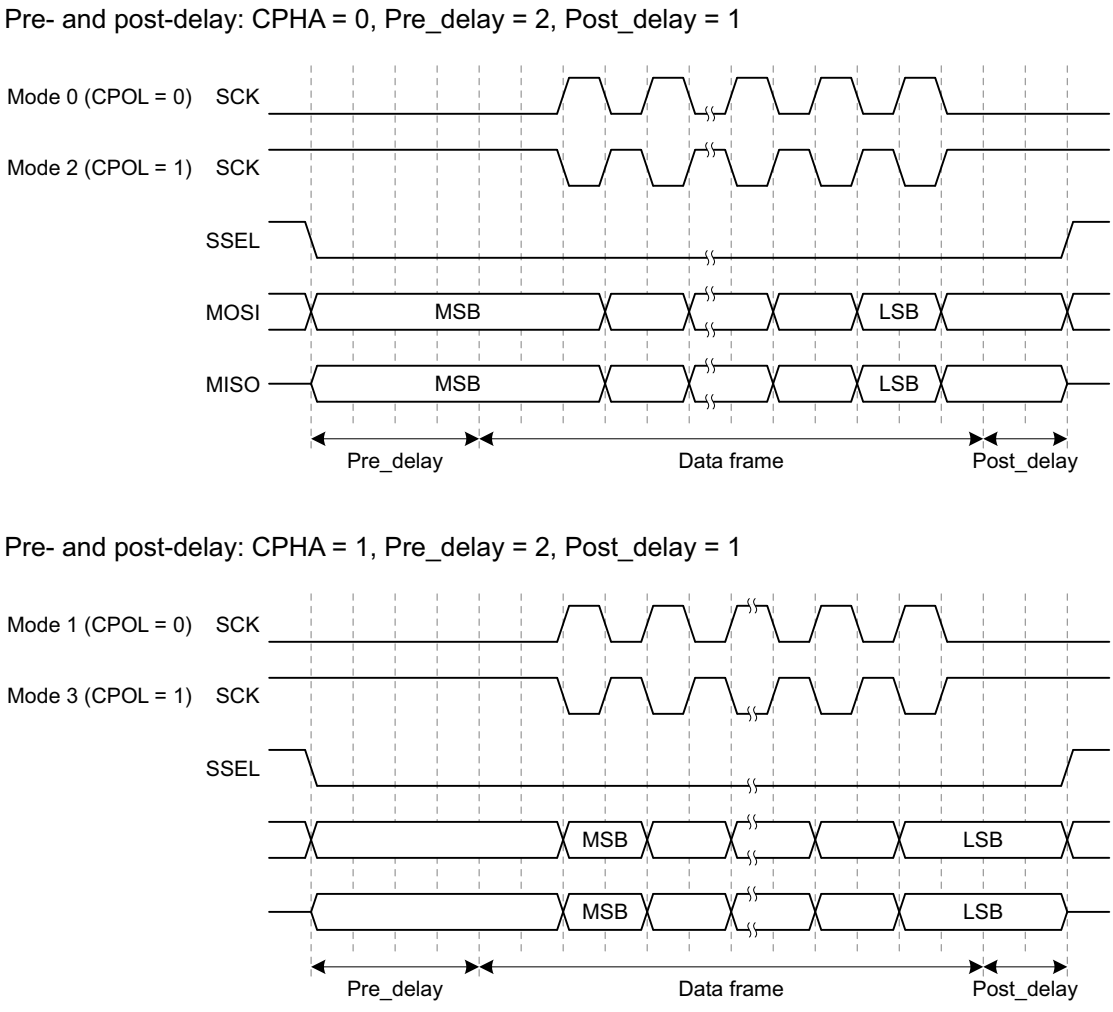


Fig 62. Pre_delay and Post_delay

24.6.3.2 Frame_delay

The Frame_delay value controls the amount of time at the end of each frame. This delay is inserted when the TXCTL[EOFR] = 1. Frame_delay is illustrated by the examples in Figure 63. Note that frame boundaries occur only where specified. This is because frame lengths can be any size, involving multiple data writes. See Section 24.6.7 for more information.

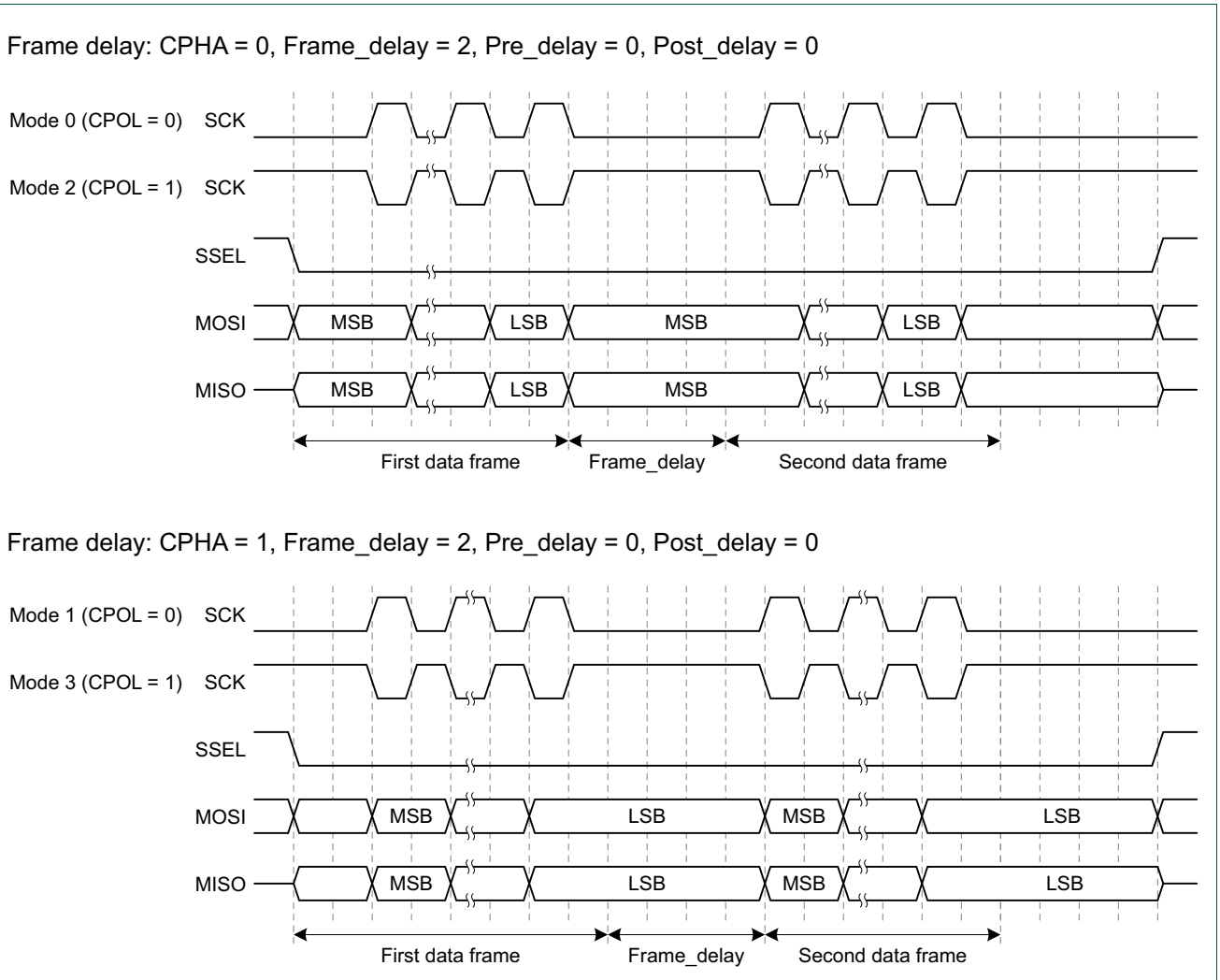


Fig 63. Frame_delay

24.6.3.3 Transfer_delay

The Transfer_delay value controls the minimum amount of time that SSEL is deasserted between transfers, because the FIFOWR[EOT] = 1. When Transfer_delay = 0, SSEL may be deasserted for a minimum of one SPI clock time. Transfer_delay is illustrated by the examples in Figure 64.

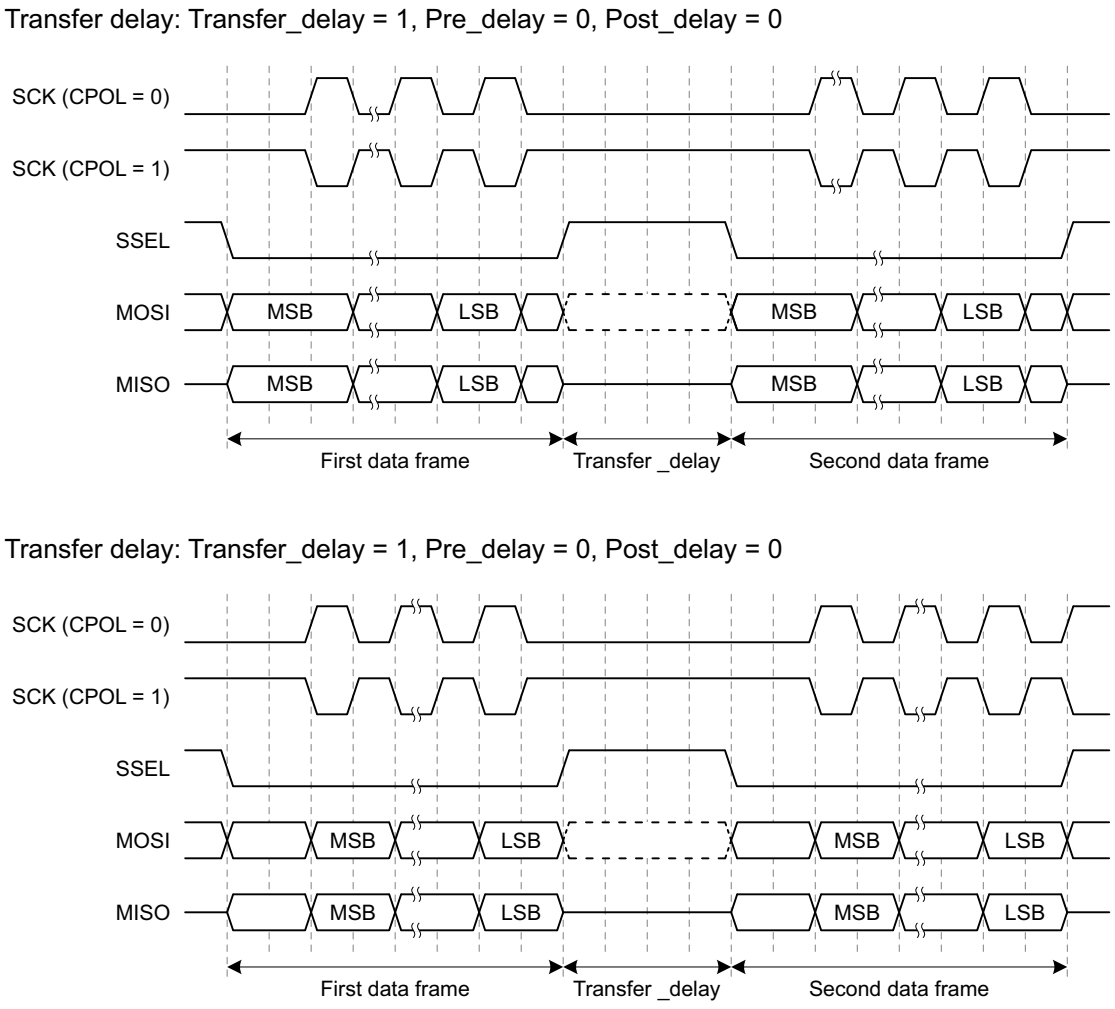


Fig 64. Transfer_delay

24.6.4 Clocking and data rates

In order to use the SPI, clocking details must be defined. The system clock must be selected with the SYSCON_SPICLKSEL; both SPI0 and SPI1 receive the same clock. Within each SPI peripheral a divider can be configured with the SPI DIV register.

24.6.4.1 Data rate calculations

The SPI interface is designed to operate asynchronously from any on-chip clocks, and without the need for overclocking.

In slave mode, this means that the SCK from the external master is used directly to run the transmit and receive shift registers and other logic.

In master mode, the SPI rate clock produced by the SPI clock divider is used directly as the outgoing SCK.

The SPI clock divider is an integer divider. The SPI in master mode can be set to run at the same speed as the selected SPICLK, or at lower integer divide rates. The SPI rate will be = SPICLK/ DIVVAL.

In slave mode, the clock is taken from the SCK input and the SPI clock divider is not used

24.6.5 Slave select

The SPI1 block provides for three Slave Select inputs in slave mode or outputs in master mode. Each SSEL can be set for normal polarity (active low), or can be inverted (active high). Representation of the 3 SSELs in a register is always active low. If an SSEL is inverted, this is done as the signal leaves/enters the SPI block.

In slave mode, any asserted SSEL that is connected to a pin will activate the SPI. Therefore care is needed to ensure that inactive SSEL lines return to inactive state; abnormal operation may result if this is not done.

In master mode, all SSELs that are connected to a pin will be output as defined in the SPI registers. In the latter case, the SSELs could potentially be decoded externally in order to address more than three slave devices. Note that at least one SSEL is asserted when data is transferred in master mode.

In master mode, Slave Selects come from the TXSSEL bits in the FIFOWR register. In slave mode, the state of all three SSELs is saved along with received data in the FIFORD[RXSSELn_N] field.

For SPI0 block only one slave select is supported.

24.6.6 DMA operation

A DMA request is provided for each SPI direction, and can be used in lieu of interrupts for transferring data by configuring the DMA controller appropriately. The DMA controller provides an acknowledgment signal that clears the related request when it completes handling that request.

The transmitter DMA request is asserted when Tx DMA is enabled and the transmitter can accept more data.

The receiver DMA request is asserted when Rx DMA is enabled and received data is available to be read.

24.6.6.1 DMA master mode End-Of-Transfer

When using polled or interrupt mode to transfer data in master mode, the transition to end-of-transfer status (drive SSEL inactive) is straightforward. The FIFOWR[EOT] bit would be set just before or along with the writing of the last data to be sent.

When using the DMA in master mode, the end-of-transfer status (drive SSEL inactive) can be generated in a number of ways:

1. Using DMA interrupt and a second DMA transfer:

   To use only 8 or 16 bit wide DMA transfers for all the data, a second DMA transfer can be used to terminate the transfer (drive SSEL inactive).

The transfer would be started by setting the control bits and then initiating the DMA transfer of all but the last byte/halfword of data. The DMA completion interrupt function must modify the control bits to set FIFOWR[EOT] and then set-up DMA to send the last data.

2. Using DMA and SPI interrupts (or background SPI status polling):

   To use only one 8 or 16 bit wide DMA transfer for all the data, two interrupts would be required to properly terminate the transfer (drive SSEL inactive).

   The SPI Tx DMA completion interrupt function sets the FIFOTRIG[TXLVL] to 0 and sets the interrupt enable bit in the FIFOINTENSET[TXLVL].

   The interrupt function handling the SPI TXLVL would set the STAT[ENDTRANSFER], to force termination after all data output is complete.

3. Using DMA linked descriptor:

   The DMA controller provides for a linked list of DMA transfer control descriptors. The initial descriptor(s) can be used to transfer all but the last data byte/halfword. These data transfers can be done as 8 or 16 bit wide DMA operations. A final DMA descriptor, linked to the first DMA descriptor, can be used to send the last data along with control bits to the FIFOWR register. The control bits would include the setting of the FIFOWR[EOT] bit.

   Note: The DMA interrupt function cannot set the STAT[ENDTRANSFER] bit. This may terminate the transfer while the FIFO still has data to send.

4. Using 32 bit wide DMA:

   Write both data and control bits to FIFOWR for all data. The control bits for the last entry would include the setting of the FIFOWR[EOT] bit. This also allows a series of SPI transactions involving multiple slaves with one DMA operation, by changing the TXSSELn_N bits.

24.6.7 Data lengths greater than 16 bits

The SPI interface handles data frame sizes from 4 to 16 bits directly. Larger sizes can be handled by splitting data up into groups of 16 bits or less. For example, 24 bits can be supported as 2 groups of 16 bits and 8 bits or 2 groups of 12 bits, among others. Frames of any size, including greater than 32 bits, can supported in the same way.

Details of how to handle larger data widths depend somewhat on other SPI configuration options. For instance, if it is intended for Slave Selects to be deasserted between frames, then this must be suppressed when a larger frame is split into more than one part. Sending 2 groups of 12 bits with SSEL deasserted between 24-bit increments, for instance, would require changing the value of the FIFOWR[EOF] bit on alternate 12-bit frames.

24.7 Data stalls

A stall for Master transmit data can happen in modes 0 and 2 when SCK cannot be returned to the rest state until the MSB of the next data frame can be driven on MOSI. In this case, the stall happens just before the final clock edge of data if the next piece of data is not yet available.

A stall for Master receive can happen when a FIFO overflow (see FIFOSTAT[RXERR]) would otherwise occur if the transmitter was not stalled. In modes 0 and 2, this occurs if the FIFO is full when the next piece of data is received. This stall happens one clock edge earlier than the transmitter stall.

In modes 1 and 3, the same kind of receiver stall can occur, but just before the final clock edge of the received data. Also, a transmitter stall will not happen in modes 1 and 3 because the transmitted data is complete at the point where a stall would otherwise occur, so it is not needed.

Stalls are reflected in the STAT[STALLED], which indicates the current SPI status. The transmitter will be stalled until data is read from the receive FIFO. Use the RXIGNORE control bit setting to avoid the need to read the received data.

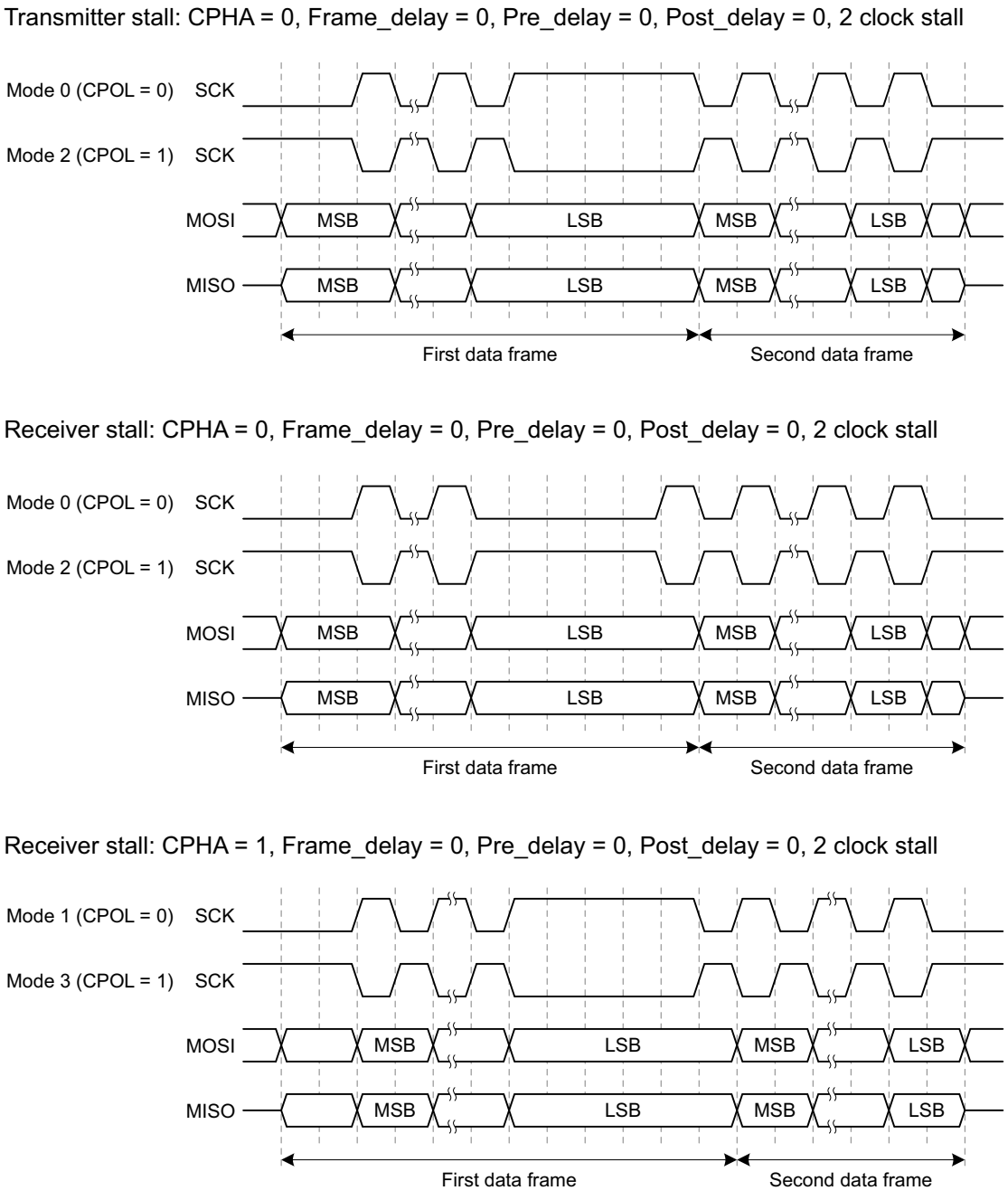


Fig 65. Examples of data stalls

24.8 Software control

To use the functionality of the SPI module it is recommended to use software functions from within fsl_spi.c.

25.1 How to read this chapter

Two I2C functions (I2C0 and I2C1) are available on all JN5189(T)/JN5188(T) devices. Additionally, on the JN5189T and JN5188T device a further I2C interface (I2C2) is provided to interface to the internal NTAG device.

25.2 Features

- Independent Master, Slave, and Monitor functions.
- Bus speeds supported:
  - Standard mode, up to 100 kbits/s.
  - Fast-mode, up to 400 kbits/s.
  - Fast-mode Plus, up to 1 Mbits/s (on pins PIO0_10 and PIO0_11 that include specific I2C support).
  - High speed mode, 3.4 Mbits/s as a Slave only (on pins PIO0_10 and PIO0_11 that include specific I2C support).
- Supports both Multi-master and Multi-master with Slave functions.
- Multiple I2C slave addresses supported in hardware.
- One slave address can be selectively qualified with a bit mask or an address range in order to respond to multiple I2C bus addresses.
- 10-bit addressing supported with software assist.
- Supports System Management Bus (SMBus).
- Separate DMA requests for Master, Slave, and Monitor functions.
- No chip clocks are required in order to receive and compare an address as a Slave, so this event can wake up the device from deep-sleep mode. Additionally, I2C0 can optionally generate a wake-up from power down.
- Automatic modes optionally allow less software overhead for some use cases.

25.3 Pin description

The I2C pins are fixed-pin functions and enabled through IOCON. Refer to the IOCON settings in Table 48 and in Section 12.4.2.

Table 46. I2C-bus pin description

Function	Type	Pin name used in data sheet Pin Description	Description
SCL	I/O	I2Cn_SCL	I2C serial clock
SDA	I/O	I2Cn_SDA	I2C serial data

The following table shows the pin locations that can be configured for I2C functionality. See IOCON configuration for FUNC settings associated with these mappings.

Table 47. I2C bus pin assignments

Pin	Possible Pin Assignment	
I2C0_SCL	PIO0_10[1]	PIO0_15
I2C0_SDA	PIO0_11[1]	PIO0_16
I2C1_SCL	PIO0_6	PIO0_12
I2C1_SDA	PIO0_7	PIO0_13
I2C2_SCL	Internal connection	
I2C2_SDA	Internal connection	

[1] Dedicated I2C IO cells

Table 48: Suggested I2C pin settings

IOCON bit(s)	Standard IO pin		PIO10 or 11	
	Name	Setting	Name	Setting
12	—	—	IO_CLAMP	Set to 0, normal operation
11	IO_CLAMP	Set to 0, normal operation	—	—
10	OD	Set to 1 for pseudo open-drain output function	OD	Set to 0, using I2C mode in IO cell
9	SLEW1	With SLEW0: Set to 0 so lowest speed which is adequate to I2C speeds	FSEL	Set to 0, unless input filtering is required
8	FILTEROFF	Generally, set to 1, unless filtered inputs are required then set to 0 for a 10ns filter	FILTEROFF	Generally, set to 1, unless filtered inputs are required then set to 0 and set FSEL and EGP correctly to get 3,10 or 50ns filter.
7	DIGIMODE	Set to 1	DIGIMODE	Set to 1
6	INVERT	Set to 0	INVERT	Set to 0
5	SLEW0	See SLEW1	EHS	Set to 0 for I2C mode
4	MODE[1]	With MODE[0]: Set to 0 (pull-up enabled). If internal pull is not required then set to 0x2.	ECS	Set to 0 for I2C mode
3	MODE[0]	See MODE[0]	EGP	Set to 0 for I2C mode
2:0	FUNC[2:0]	Must select the correct function for this peripheral, see IOCON configuration	FUNC[2:0]	Must select the correct function for this peripheral, see IOCON configuration

25.4 Basic configuration

Configure the I2C and related clocks as follows:

- If needed, use the SYSCON_PRESETCTRL1 register to reset the required I2C function.
- Configure the I2C for the desired functions:
  - In the SYSCON_AHBCLKCTRL1 register, set the appropriate bit for the related I2C Interface in order to enable the clock to the register interface.
  - Enable or disable the related I2C Interface interrupt in the NVIC.

- Configure the related I2C Interface pin functions via IOCON, see Chapter 12 “I/O Pin Configuration (IOCON)”.
- Configure the I2C clock and data rate. This includes the CLKDIV register for both master and slave modes, and MSTTIME register for master mode. Also see Section 25.6.2 “Bus rates and timing considerations”.

25.4.1 I2C transmit/receive in master mode

In this example, I2C Interface 1 is configured as an I2C master. The master sends 8 bits to the slave and then receives 8 bits from the slave.

I2C1 interface uses standard IO cells, only PIO0_10 and PIO0_11 use special I2C IO cells. Configure the IO cells for the correct settings in IOCON.

The transmission of the address and data bits is controlled by the STAT[MSTPENDING] status bit. Whenever the status is Master pending, the master can read or write to the MSTDAT register and go to the next step of the transmission protocol by writing to the MSTCTL register.

Configure the I2C bit rate:

- The I2C system clock must be 8 MHz. Hence, configure the clock using SYSCON_I2CCLKSEL and the CLKDIV[DIVVAL] setting within the I2C block to get 8 MHz clock results.
- Set the SCL high and low times to complete the bus rate setup.

25.4.1.1 Master write to slave

This example uses polling to control operations and is not interrupt driven. Configure the I2C as a master: set the CFG[MSTEN] bit to 1. Then wait for the pending status to be set (STAT[MSTPENDING] = 1) by polling the STAT register.

Check the status register STAT[MSTSTATE] is indicating IDLE. If not, then an error has occurred.

Write data to the slave:

1. Write the 7-bit slave address with the $\overline{RW}$ bit set to 0 to the Master data register MSTDAT.
2. Start the transmission by setting the MSTCTL[MSTSTART] bit to 1. The following happens:
   - The pending status is cleared and the I2C-bus is busy.
   - The I2C master sends the start bit and address with the $\overline{RW}$ bit to the slave.
3. Wait for the pending status to be set (STAT[MSTPENDING] = 1) by polling the STAT register. The STAT[MSTSTATE] can be checked that it indicates TX by reading the STAT register. If not in this state then an error has occurred.
4. Write 8 bits of data to the MSTDAT register.
5. Continue with the transmission of data by setting the MSTCTL[MSTCONTINUE] bit to 1. The following happens:
   - The pending status is cleared and the I2C-bus is busy.
   - The I2C master sends the data bits to the slave address.

6. Wait for the pending status to be set (STAT[MSTPENDING] = 1) by polling the STAT register.
7. Stop the transmission by setting the MSTCTL[MSTSTOP] bit to 1.
8. Wait for the pending status to be set (STAT[MSTPENDING] = 1) by polling the STAT register.

25.4.1.2 Master read from slave

This example uses polling to control the sequence and does not use interrupts. Configure the I2C as a master: set the CFG[MSTEN] bit to 1. Then wait for the pending status to be set (STAT[MSTPENDING] = 1) by polling the STAT register.

Check the status register STAT[MSTSTATE] is indicating IDLE. If not then an error has occurred.

Read data from the slave:

1. Write the slave address with the $\overline{RW}$ bit set to 1 to the Master data register MSTDAT.
2. Start the transmission by setting the MSTCTL[MSTSTART] bit to 1. The following happens:
   - The pending status is cleared and the I2C-bus is busy.
   - The I2C master sends the start bit and address with the $\overline{RW}$ bit to the slave.
   - The slave sends 8 bit of data.
3. Wait for the pending status to be set (STAT[MSTPENDING] = 1) by polling the STAT register. The status register STAT[MSTSTATE] can be checked that it indicates RX. If not in this state then an error has occurred.
4. Read 8 bits of data from the MSTDAT register.
5. Stop the transmission by setting the MSTCTL[MSTSTOP] bit to 1.

25.4.2 I2C receive/transmit in slave mode

In this example, I2C1 is used as an I2C slave. The slave receives 8 bits from the master and then sends 8 bits to the master. The SCL and SDA functions must be enabled on suitable pins, see Table 48.

The pins should be configured as required for the I2C-bus mode.

The transmission of the address and data bits is controlled by the STAT[SLVPENDING] status bit. Whenever the status is Slave pending, the slave can acknowledge (“ack”) or send or receive an address and data. The received data or the data to be sent to the master are available in the SLVDAT register. After sending and receiving data, continue to the next step of the transmission protocol by writing to the SLVCTL register.

25.4.2.1 Slave read from master

This example uses polling to control the sequence and does not use interrupts. Configure the I2C as a slave with address x:

1. Write the slave address x to the address 0 match register.
2. Set the CFG[SLVEN] bit to 1.

Read data from the master:

1. Wait for the pending status to be set (SLVPENDING = 1) by polling the STAT register. Check the STAT[SLVSTATE] is indicating ADDR. If not then an error has occurred.
2. Acknowledge (“ack”) the address by setting SLVCTL[SLVCONTINUE] = 1 in the slave control register.
3. Wait for the pending status to be set (STAT[SLVPENDING] = 1) by polling the STAT register. Check the status register SLVSTATE is indicating RX by reading the STAT register. If not then an error has occurred.
4. Read 8 bits of data from the SLVDAT register.
5. Acknowledge (“ack”) the data by setting SLVCTL[SLVCONTINUE] = 1 in the slave control register.

25.4.2.2 Slave write to master

This example uses polling to control the sequence and does not use interrupts. Configure the I2C as a slave with address x:

1. Write the slave address x to the address 0 match register.
2. Set the CFG[SLVEN] bit to 1.

Write data to the master:

1. Wait for the pending status to be set (STAT[SLVPENDING] = 1) by polling the STAT register. Check the status register STAT[SLVSTATE] indicating ADDR. If not then an error has occurred.
2. ACK the address by setting SLVCTL[SLVCONTINUE] = 1 in the slave control register.
3. Wait for the pending status to be set (STAT[SLVPENDING] = 1) by polling the STAT register. Check the status register STAT[SLVSTATE] is indicating TX. If not then an error has occurred.
4. Write 8 bits of data to SLVDAT register.
5. Continue the transaction by setting SLVCTL[SLVCONTINUE] = 1 in the slave control register.

25.4.3 Configure the I2C for wake-up

In sleep mode, any activity on the I2C-bus that triggers an I2C interrupt can wake up the part, provided that the interrupt is enabled in the INTENSET register and the NVIC. As long as the I2C Interface clock remains active in sleep mode, the I2C can wake up the part independently of whether the I2C interface is configured in master or slave mode.

In deep-sleep mode, the I2C clock is turned off as are all peripheral clocks. However, if the I2C is configured in slave mode and an external master on the I2C-bus provides the clock signal, the I2C interface can create an interrupt asynchronously. This interrupt, if enabled in the NVIC and in the I2C interface INTENCLR register, can then wake up the core.

25.4.3.1 Wake-up from sleep mode

- Enable the I2C interrupt in the NVIC.
- Enable the I2C wake-up event in the INTENSET register. Wake-up on any enabled interrupts is supported (see the INTENSET register). Examples are the following events:

- Master pending
- Change to idle state
- Start/stop error
- Slave pending
- Address match (in slave mode)
- Data available/ready

25.4.3.2 Wake-up from deep-sleep mode

- Enable the I2C interrupt in the NVIC.
- Enable the I2C interrupt in the SYSCON_STARTER0 register to create the interrupt signal asynchronously while the core and the peripheral are not clocked.
- Configure the I2C in slave mode.
- Enable the I2C interrupt in the INTENSET register which configures the interrupt as wake-up event. The following events are examples:
  - Slave deselect
  - Slave pending (wait for read, write, or ACK)
  - Address match
  - Data available/ready for the Monitor function

25.4.3.3 Wake-up from power down mode

Only I2C0 is possible to wake-up from power down mode.

- Enable the I2C interrupt in the SYSCON_STARTER0 register to create the wake-up signal asynchronously while the core and the peripheral are not clocked
- Configure the I2C is slave mode
- Enable the I2C interrupt in the INTENSET register which configures the interrupt as a wake-up event. The following events are examples:
  - slave deselect
  - slave pending (wait for read, write or ACK)
  - address match
  - Data available/ready for the monitor function
- Ensure that the Comm0 power domain will be active during the power cycle, using the power control APIs.
- Perform power down request. Note, the Low Power API should be used to perform the configuration and execution of power down cycles.

25.5 General description

The architecture of the I2C-bus interface is shown in Figure 66.

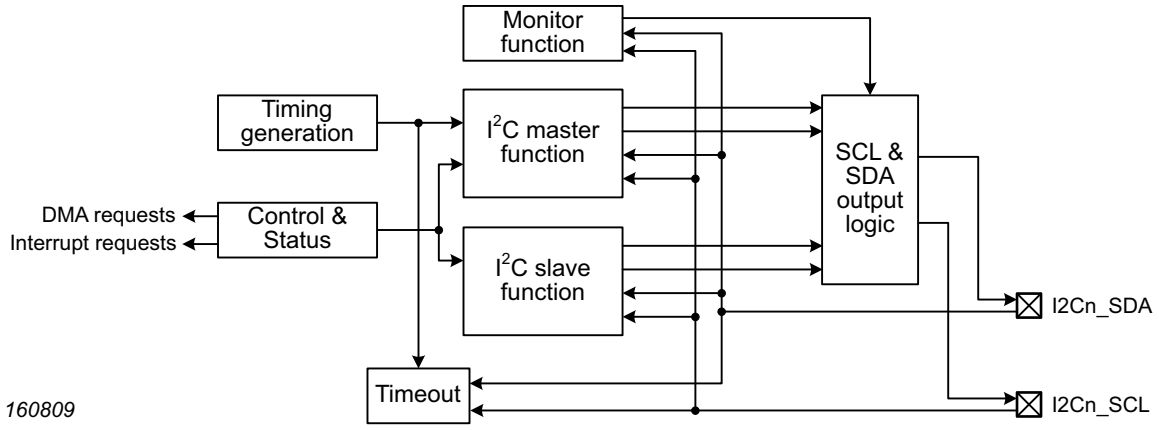


Fig 66. I2C block diagram

25.6 Functional description

25.6.1 AHB bus access

The I2C registers support only word writes. Byte and halfword writes are not supported in conjunction with the I2C function.

25.6.2 Bus rates and timing considerations

Due to the nature of the I2C bus, it is generally not possible to guarantee a specific clock rate on the SCL pin. On the I2C-bus, the clock can be stretched by any slave device, extended by software overhead time, etc.

In a multi-master system, the master that provides the shortest SCL high time will cause that time to appear on SCL as long as that master is participating in I2C traffic (i.e. when it is the only master on the bus, or during arbitration between masters).

In addition, I2C implementations generally base subsequent actions on what actually happens on the bus lines. For instance, a bus master allows SCL to go high. It then monitors the line to make sure it actually did go high (this would be required in a multi-master system). This results in a small delay before the next action on the bus, caused by the rise time of the open drain bus line.

Rate calculations give a base frequency that represents the fastest that the I2C bus could operate if nothing slows it down.

25.6.2.1 Rate calculations

Master timing

SCL high time (in I2CCLK function clocks) =
I2C clock divider * SCL high multiplier

SCL low time (in I2CCLK function clocks) =
I2C clock divider * SCL low multiplier

Nominal SCL rate =
I2CCLK function clock rate / (SCL high time + SCL low time)

Remark: DIVVAL must be ≥ 1.

Remark: For 400 kHz clock rate, the clock frequency after the I2C divider (divval) must be ≤ 6.5 MHz. Table 49 shows the recommended settings for 400 kHz clock rate.

Table 49. Settings for I2C baud rate

I2CCLK	DIVVAL	Interal I2C CLK	MSTSCL-HIGH	MSTSCL-LOW	I2C Baud
32 MHz	3	8 MHz	2	2	1 Mb/s Fast-mode plus
48 MHz	5	8 MHz	2	2	1 Mb/s Fast-mode plus
32 MHz	4	6.4 MHz	6	6	400 kb/s Fast Mode
48 MHz	7	6.0 MHz	6	5	400 kb/s Fast Mode
32 MHz	19	1.6 MHz	6	6	100 kb/s Fast Mode
48 MHz	31	1.5 MHz	6	5	100 kb/s Fast Mode

Slave timing

Most aspects of slave operation are controlled by SCL received from the I2C bus master. However, if the slave function stretches SCL to allow for software response, it must provide sufficient data setup time to the master before releasing the stretched clock. This is accomplished by inserting one clock time of CLKDIV at that point.

If CLKDIV is already configured for master operation, that is sufficient. If only the slave function is used, CLKDIV should be configured such that one clock time is greater than the Data set-up time $t_{SU;DAT}$ value noted in the I2C bus specification for the I2C mode that is being used.

25.6.2.2 Bus rate support

The I2C interface can support 4 modes from the I2C bus specification:

- Standard-mode (SM, rate up to 100 kbits/s)
- Fast-mode (FM, rate up to 400 kbits/s)
- Fast-mode Plus (FM+, rate up to 1 Mbits/s)
- High-speed mode (HS, rate up to 3.4 Mbits/s)

For operation of the I2C interface, an external pull-up resistor is required on the clock and data lines. For high speed mode, a 2.7 kΩ pull-up resistor is recommended; for slower speeds, a 1 kΩ resistor is recommended.

Refer to Ref. 2 “UM10204” for details of I2C modes and other details.

The I2C interface supports Standard-mode, Fast-mode, and Fast-mode Plus with the same software sequence, which also supports SMBus. High-speed mode is intrinsically incompatible with SMBus due to conflicting requirements and limitations for clock stretching, and therefore requires a slightly different software sequence.

25.6.2.2.1 High-speed mode support

High-speed mode requires different pin filtering, somewhat different timing, and a different drive strength on SCL for the master function. The changes needed for the handling of the acknowledge bit mean that SMBus cannot be supported when the I2C is configured to be HS capable. This limitation is intrinsic to the SMBus and High-speed I2C specifications.

Because of the timing of changes to pin drive strength and filtering, the I2C interface is designed to directly control those pin characteristics when configured to be HS capable. The I2C also recognizes HS master codes and responds to programmed addresses when HS capable.

For software consistency, the changes required for handling of acknowledge and address recognition, and which affect when interrupts occur, are always in effect when the I2C is configured to be HS capable. This means that software does not need to know if a particular transfer is actually in HS mode or not.

25.6.2.2.2 Clock stretching

The I2C interface automatically stretches the clock when it does not have sufficient information on how to proceed, i.e. software has not supplied data and/or instructions to generate a start or stop. In principle, at least, I2C can allow the clock to be stretched by any bus participant at any time that SCL is low, in SM, FM, and MF+ modes.

In practice, the I2C interface described here may stretch SCL at the following times, in SM, FM, and MF+ modes:

- As a Slave:
  - after an address is received that complies with at least one slave address (before the address is acknowledged)
  - as a slave receiver, after each data byte received (software then acknowledges the data)
  - as a slave transmitter, after each data byte is sent and the matching acknowledge is received from the master
- As a master:

  after each
  - address is sent and the acknowledge bit has been received
  - as a master receiver, after each data byte is received (software then acknowledges the data)
  - as a master transmitter, after each data byte is sent and the matching acknowledge bit has been received from the slave

In HS mode:

- As a Slave (only slave functions in HS mode are supported on this device)
  - as a slave receiver, after each data byte is received and automatically acknowledged
  - as a slave transmitter, after each data byte is sent and the matching acknowledge is received from the master

In each case, the relevant pending flag (STAT[MSTPENDING] or STAT[SLVPENDING]) is set at the point where clock stretching occurs.

25.6.3 Time-out

A time-out feature on an I2C interface can be used to detect a “stuck” bus and potentially do something to alleviate the condition. Two different types of time-out are supported. Both types apply whenever the I2C interface and the time-out function are both enabled. Master, Slave, or Monitor functions do not need to be enabled.

In the first type of time-out, reflected by the STAT[EVENTTIMEOUT] flag, the time between bus events governs the time-out check. These events include Start, Stop, and all changes on the I2C clock (SCL). This time-out is asserted when the time between any of these events is longer than the time configured in the TIMEOUT register. This time-out could be useful in monitoring an I2C bus within a system as part of a method to keep the bus running of problems occur.

The second type of I2C time-out is reflected by the STAT[SCLTIMEOUT] flag. This time-out is asserted when the SCL signal remains low longer than the time configured in the TIMEOUT register. This corresponds to SMBus time-out parameter $T_{TIMEOUT}$. In this situation, a slave could reset its own I2C interface in case it is the offending device. If all listening slaves (including masters that can be addressed as slaves) do this, then the bus will be released unless it is a current master causing the problem. Refer to the SMBus specification for more details.

Both types of time-out are generated only when the I2C bus is considered busy, i.e. when there has been a Start condition more recently than a Stop condition.

25.6.4 Slave addresses

When operating as a slave it is possible to configure four independent slave addresses that will be used to match received addresses against.

In addition it is possible to extend the first slave address, SLVADR0, to be a range of addresses from SLVADR0 to SLVQUAL0 if the slave address qualifying feature is enabled. In this case the address matches when SLVADR0[7:1] ≤ received address ≤ SLVQUAL0[7:1].

25.6.5 Ten-bit addressing

Ten-bit addressing is accomplished by the I2C master sending a second address byte to extend a particular range of standard 7-bit addresses. In the case of the master writing to the slave, the I2C frame simply continues with data after the 2 address bytes. For the master to read from a slave, it needs to reverse the data direction after the second address byte. This is done by sending a Repeated Start, followed by a repeat of the same standard 7-bit address, with a Read bit. The slave must remember that it had been addressed by the previous write operation and stay selected for the subsequent read with the correct partial I2C address.

For the Master function, the I2C is simply instructed to perform the 2-byte addressing as a normal write operation, followed either by more write data, or by a Repeated Start with a repeat of the first part of the 10-bit slave address and then reading in the normal fashion.

For the Slave function, the first part of the address is automatically matched in the same fashion as 7-bit addressing; the last 8 bits of the address must be checked by software. For 10-bit addressing and first address byte will be 1110XX where XX are the first two bits of the 10-bit address. The slave address matching can be performed using the standard

slave address matching or the slave address qualified feature, see Section 25.6.4 “Slave addresses”. In the case of Slave Receiver mode, data is received in the normal fashion after software matches the first data byte to the remaining portion of the 10-bit address. The Slave function should record the fact that it has been addressed, in case there is a follow-up read operation.

For Slave Transmitter mode, the slave function responds to the initial address in the same fashion as for Slave Receiver mode, and checks that it has previously been addressed with a full 10-bit address. If the address matched is address 0, and address qualification is enabled, software must check that the first part of the 10-bit address is a complete match to the previous address before acknowledging the address.

25.6.6 Clocking and power considerations

The Master function of the I2C always requires a peripheral clock to be running in order to operate. The Slave function can operate without any internal clocking when the slave is not currently addressed. This means that reduced power modes up to deep-sleep mode can be entered, and the device will wake up when the I2C Slave function recognizes an address. Monitor mode can similarly wake up the device from a reduced power mode when information becomes available.

25.6.7 Interrupt handling

The I2C provides a single interrupt output that handles all interrupts for Master, Slave, and Monitor functions.

25.6.8 DMA

DMA with the I2C is done only for data transfer, DMA cannot handle control of the I2C. Once DMA is transferring data, I2C acknowledges are handled implicitly. No CPU intervention is required while DMA is transferring data.

Generally, data transfers can be handled by DMA for Master mode after an address is sent and acknowledged by a slave, and for Slave mode after software has acknowledged an address. In either mode, software is always involved in the address portion of a message. In master and slave modes, data receive and transmit data can be transferred by the DMA. The DMA supports three DMA requests: data transfer in master mode, slave mode, and Monitor mode.

DMA may be used in connection with Automatic Operation in order to minimize software overhead time for I2C handling.

A received NACK (from a slave in Master mode, or from a master in Slave mode) will cause DMA to stop and an interrupt to be generated. A Repeated Start sensed on the bus will similarly cause DMA to stop and an interrupt to be generated.

The Monitor function may be used with DMA if a channel is available See Section 17.5.1.1.1 “DMA with I2C monitor mode” for how DMA channels are used with the Monitor function.

25.6.8.1 DMA as a Master transmitter

A basic sequence for a Master transmitter:

- Software sets up DMA to transmit a message.

- Software causes a slave address with write command to be sent and checks that the address was acknowledged.
- Software turns on DMA mode in the I^2C.
- DMA transfers data and eventually completes the transfer.
- Software causes a stop (or repeated start) to be sent.

Software will be invoked to handle any exceptions to the standard transfer, such as the slave sending a NACK before the end of the transfer.

25.6.8.2 DMA as a Master receiver

A basic sequence for a Master receiver:

- Software sets up DMA to receive a message.
- Software causes a slave address with read command to be sent and checks that the address was acknowledged.
- Software starts DMA.
- DMA completes.
- Software causes a stop or repeated start to be sent.

Software will be invoked to handle any exceptions to the standard transfer.

25.6.8.3 DMA as a Slave transmitter

A basic sequence for a Slave transmitter:

- Software acknowledges an I^2C address.
- Software sets up DMA to transmit a message.
- Software starts DMA.
- DMA completes.

25.6.8.4 DMA as a Slave receiver

A basic sequence for a Slave receiver:

- Software receives an interrupt for a slave address received, and acknowledges the address.
- Software sets up DMA to receive a message, less the final data byte.
- Software starts DMA.
- DMA completes.
- Software sets SLVNACK prior to receiving the final data byte.
- Software receives the final data byte.

25.6.9 Automatic operation

Automatic operation modes provide a way to reduce software overhead for I^2C slave functions with some limitations. They are intended to be used primarily in conjunction with slave DMA. Related control bits are SLVCTL[SLVDMA], SLVCTL[AUTOACK], and

SLVCTL[AUTOMATCHREAD], and the SLVADR0[AUTONACK]. Table 50 shows how these controls may be used. These cases apply when an address matching SLVADR0, qualified by SLVQUAL0, is received.

Table 50: Automatic operation cases

Conditions:			Response:		
AUTONACK bit	**AUTOACK bit**	**Received R/W bit matches AUTOMATCHREAD**	**SLVPENDING interrupt generated?**	**ACK/NACK on I2C bus**	**Description**
0	0	x	Yes	None	Normal, non-automatic operation.
0	1	No	Yes	None	Automatic slave DMA: unexpected read/write case. Same as normal non-automatic operation.
x	1	Yes	No	ACK	Automatic slave DMA: expected read/write case. When the automatic Ack is sent, the SLVDMA bit is set and the AUTOACK bit is cleared.
1	0	x	No	NACK	Bus is ignored until software changes the setup.
1	1	No	No	NACK	Bus is ignored until software changes the setup.

25.6.10 Master and slave states

Within the Status Register, STAT, there are fields for both the master and slave state. The following two tables show the state descriptions, actions possible and indicate if DMA is allowed in that state. This is presented for the master and slave functions.

Table 51. Master function state codes (MSTSTATE)

MSTSTATE	Descriptions	Actions	DMA allowed
0x0	Idle. The master function is available to be used for a new transaction.	Send a start or disable MSTPENDING interrupt if the master function is not needed currently.	No
0x1	Received data is available (master receiver mode). Address plus read was previously sent and acknowledged by slave.	Read data and either continue, send a stop, or send a repeated start.	Yes
0x2	Data can be transmitted (master transmitter mode). Address plus write was previously send and acknowledged by slave.	Send data and continue, or send a stop or repeated start.	Yes
0x3	Slave NACKed address	Send a stop or repeated start	No
0x4	Slave NACKed transmitted data.	Send a stop or repeated start	No

Table 52. Slave function state codes (SLVSTATE)

Master state		Descriptions	Actions	DMA allowed
0	SLVST_ADDR	Address plus R/W received. At least one of the 4 slave addresses has been matched by hardware	Software can further check the address if needed, for instance, if a subset of addresses qualified by SLVQUAL0 is to be used. Software can ACK or NACK the address by writing 1 to either SLVCTL[SLVCONTINUE] or SLVCTL[SLVNACK]. Also see regarding 10-bit addressing.	No
1	SLVST_RX	Received data is available (slave receiver mode).	Read data, reply with an ACK or a NACK	Yes
2	SLVST_TX	Data can be transmitted (slave transmitter mode).	Send data. Note that when the master NACKs data transmitted by the slave, the slave becomes de-selected.	Yes

25.6.11 Recovery from illegal bus condition

If the I2C clock signal is driven low illegally by something connected to the I2C clock pin, it is possible for the I2C master to be prevented from starting or completing a transaction correctly. In this case, it is necessary for the master to be disabled and then re-enabled in the CFG register. To identify this situation, a software timeout could be used; see the I2C driver software for a demonstration of this.

25.7 Software control

To use the functionality of the I2C module it is recommended to use software functions from within fsl_i2c.c.

26.1 How to read this chapter

The DMIC subsystem, including the dual-channel digital PDM microphone interface (DMIC) and hardware voice activity detector (HWVAD), is available on all JN5189(T)/JN5188(T) parts.

26.2 Features

- DMIC (dual/stereo digital microphone interface)
  - PDM (Pulse-Density Modulation) data input for left and/or right channels on 1 or 2 buses.
  - Flexible decimation.
  - 16 entry FIFO for each channel.
  - DC blocking or unaltered DC bias can be selected.
- HWVAD (Hardware-based voice activity detector):
  - Optimized for PCM signals with 16 kHz sampling frequency.
  - Configurable detection levels.
  - Noise envelope estimator register output for further software analysis.

26.3 Basic configuration

The DMIC is configured as follows:

- Clock:
  - Enable the clock source that will be used, if it is not already running (most oscillators may be turned off when not needed in order to save power).
  - Select the clock source that will be used in the DMICCLKSEL register.
  - Set up the clock divider (DMICCLKDIV) that follows the clock source selection mux to obtain the desired clock rate.
  - Enable clock to the peripheral in the SYSCON_AHBCLKCTRL1 register.
- Reset: The peripheral may be specifically reset using the SYSCON_PRESETCTRL1 register, but must be removed from the reset state before continuing.
- Pins: Configure pins that will be used for this peripheral in the IOCON register block. See Chapter 12.
- Interrupts: If interrupts will be used with this peripheral, enable them in the NVIC. See Chapter 9.
- Wake-up: Enable interrupts for waking up from deep-sleep mode, enable the interrupts in the SYSCON_STARTER1 register.
- PDM internal setup:
  - Enable DMIC PDM channels via the EN_CH0/1 bits in the CHANEN register.

- Set up the internal clock dividers for the PDM channels used via the DIVHFCLK0/1 registers.
- If interrupts will be used with this peripheral, enable them in the NVIC. See Chapter 9.
- If DMA will be used with the PDM data flow, the related channels of the DMA controller must be set up. See Chapter 17. DMA must also be enabled via the DMAEN bit in the FIFO_CTRL register.
- Set up other functional configurations and controls for this peripheral as needed.

- HWVAD internal setup:
  a. The HWVAD is active when the DMIC interface is active.
  b. Reset the filters with a '1' pulse of bit HWVADRSTT[RSTT].
  c. Wait for a few milliseconds to let the filter converge.
  d. Enable the HWVAD interrupt with the appropriate NVIC bit. See Chapter 9.
  e. Start the HWVAD process by toggling bit HWVADST10[ST10] from '0' to '1' and back. This also clears the interrupt flag inside the HWVAD block.
  f. In the HWVAD interrupt service routine take appropriate action.
  g. Restart the HWVAD by adjusting the HWVADST10[ST10] bit. A precedent reset of the filters using the control of HWVADRSTT[RSTT] described above is optional..

26.4 Pin description

Table 53 gives a brief summary of each of the PDM pins used by the DMIC subsystem.

Table 53. DMIC subsystem PDM pin description

Pin	Type	Description
PDM0_CLK	O	Clock output to digital microphone on PDM interface 0.
PDM0_DATA	I	Data input from digital microphone on PDM interface 0.
PDM1_CLK	O	Clock output to digital microphone on PDM interface 1.
PDM1_DATA	I	Data input from digital microphone on PDM interface 1. Also PDM clock input in bypass mode.

Recommended IOCON settings are shown in Table 54. See Chapter 12 for definitions of pin types.

Table 54: Suggested DMIC pin setting for standard GPIO pin

IOCON bit(s)	Field name	Setting	Comment
11	SSEL (IO_CLAMP)	0	No clamping
10	OD	0	Standard driven IO
9	SLEW1	0	See slew0
8	FILTEROFF	1	No input filtering
7	DIGIMODE	1	Digital pin
6	INVERT	0	No inversion
5	SLEW0	1	Slew of {0,1} to give sharper edges for clock output

Table 54: Suggested DMIC pin setting for standard GPIO pin

IOCON bit(s)	Field name	Setting	Comment
4:3	MODE	2	No Pull-up or Pull-down
2:0	FUNC		Must select the correct function for this peripheral, see Table 19 “IOMUX functions” for details.
General comment			A good choice for PDM signals

Table 55. Suggested DMIC pin setting for combo GPIO/I²C pin

IOCON bit(s)	Field name	Setting	Comment
12	IO_CLAMP	0	No clamping
10	OD	0	See slew0
9	FSEL	0	Not relevant when filtering is off
8	FILTEROFF	1	No input filtering
7	DIGIMODE	1	Digital pin
6	INVERT	0	No inversion
5	EHS	1	Enable high speed to give sharper edges for clock output
4	ECS	0	Not valid, keep low
3	EGP	1	GPIO mode
2:0	FUNC		Must select the correct function for this peripheral, see Table 19 “IOMUX functions” for details.
General comment			A good choice for PDM signals

The PDM interface provides options to support 2 single-channel microphones or a single stereo microphone. The general connections are shown in Figure 67. Specific use examples are shown in Figure 68 through Figure 70.

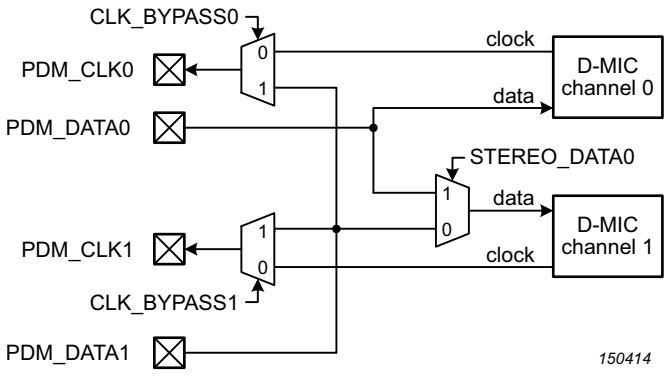
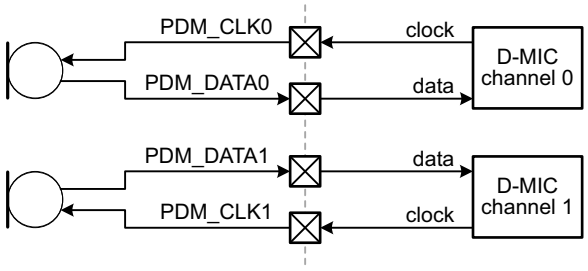
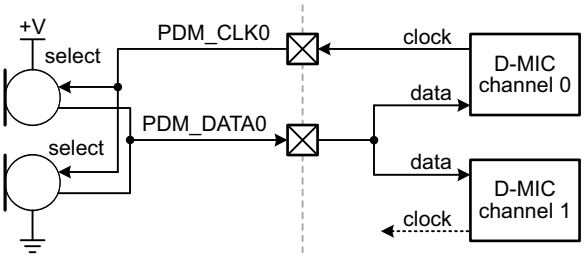


Fig 67. DMIC subsystem pin multiplexing



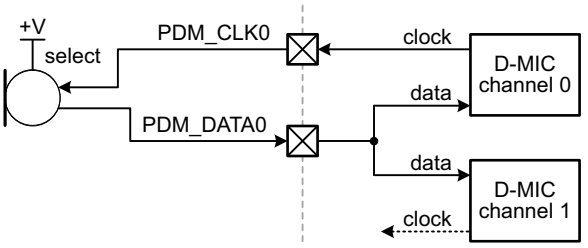
IOCFG[CLK_BYPASS0] = 0; IOCFG[CLK_BYPASS1] = 0; IOCFG[STEREO_DATA0] = 0

Fig 68. Typical connection to two independent microphones



IOCFG[CLK_BYPASS0] = 0; IOCFG[CLK_BYPASS1] = 0; IOCFG[STEREO_DATA0] = 1

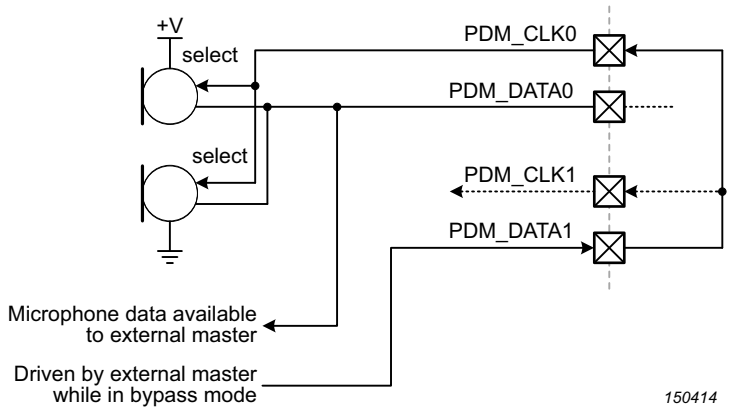
Fig 69. Typical connection to two microphones sharing a data line



IOCFG[CLK_BYPASS0] = 0; IOCFG[CLK_BYPASS1] = 0; IOCFG[STEREO_DATA0] = 1

Fig 70. Typical connection to a stereo microphone

The PDM interface also provides the possibility of an external codec or other PDM master to take over the PDM interface on this device. An example of this using dual microphones sharing one data line is shown in Figure 71.



IOCFG[CLK_BYPASS0] = 1

Fig 71. Bypass mode with an external device taking over microphone access

26.5 General description

The hardware voice activity detector (HWVAD) implements a wave envelope detector and a floor noise envelope detector. It provides an interrupt when the delta between the two detectors is larger than a predefined value. The input signal for the HWVAD can come from DMIC channel 0.

The basic detection of a voice activity can be the starting point for a more sophisticated task like for example voice recognition. As with the DMA for the DMIC subsystem, the HWVAD can be active during deep-sleep mode and therefore provide lowest power operation, compared with a software based implementation.

The DMIC receives PDM data, typically from one or two digital microphones, and produces a data stream that can be read by the CPU.

Detailed descriptions of both blocks can be found in Section 26.6 “Functional description”.

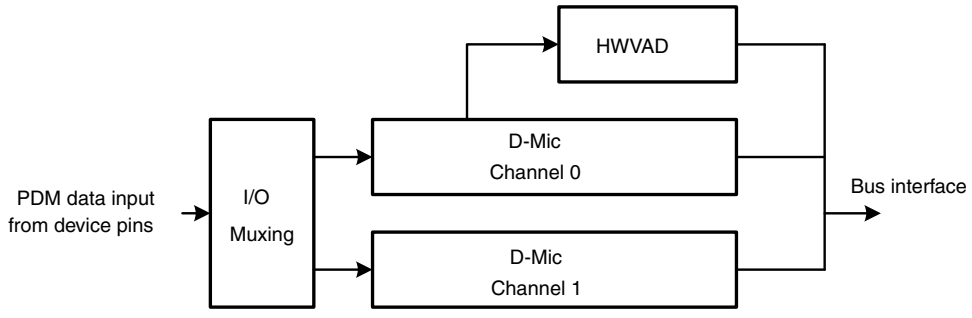


Fig 72. DMIC subsystem block diagram

26.6 Functional description

26.6.1 HWVAD

The hardware voice activity detector (HWVAD) analyses the PCM data from DMIC channel 0 by means of a filter block. Both the noise floor and the signal wave are examined and result in separate filter outputs. The HWVAD interrupt is issued when a specific delta between the signal and the noise result is detected

Gain levels for the input signal as well as for the signal and noise filter outputs can be set independently from each other, in order to adapt the HWVAD to different acoustic situations.

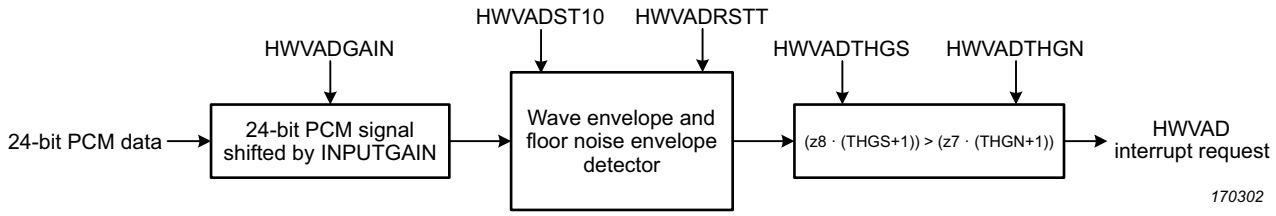
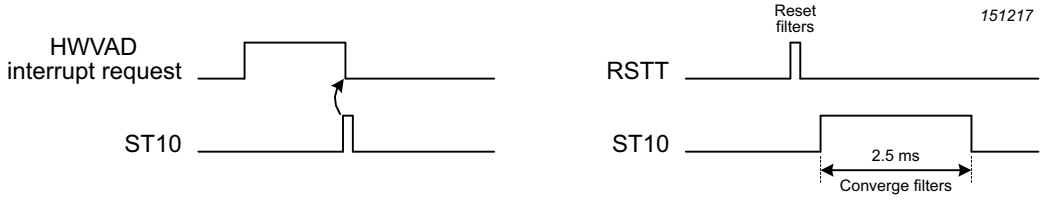


Fig 73. HWVAD block diagram

Because of the non-uniqueness of the input signal, which includes normally noise and voice with various frequency components and different volume, there is no one-and-only operation mode for the HWVAD. The few parameters as well as the chronology can play an important role for a good performance.

26.6.1.1 Basic operations

There are some basic operations for the HWVAD, which can be combined differently in order to achieve different behavior.



1) The internal HWVAD interrupt flag is reset with a short pulse on ST10
2) A pulse on RSTT resets all detection filters
3) Keeping ST10 high for a period causes a special filter convergence. The 2.5ms period is valid for 800 kHz DMIC sample rate. For 1 MHz sample rate the period is 2 ms.

Fig 74. Use of ST10 and RSTT controls (in the HWVADST10 and HWVADRSTT registers)

With bit HWVADST10[ST10], the HWVAD can be prepared for an interrupt. The internal flag is reset with the rising edge of ST10 and the HWVAD waits for the next event. In case the application involves some post-processing after a HWVAD event (outside of the interrupt service routine), the flag should only be cleared at the end of this processing. The interrupt status on NVIC level is not affected by this bit setting

With bit HWVADRSTT[RSTT], all filters can be reset. After this reset, the HWVAD filters need to converge, so for the first few milliseconds the result is not reliable. The HWVAD interrupt should be masked on NVIV level during this time frame. The wait period depends on the sample rate of the incoming data, at 1 MHz DMIC sample rate, the filters need about 2 ms to converge, for 800 kHz the period is 2.5 ms.

If it makes sense to reset the filters before starting into a new detection process depends on the use case. For a voice application, the filters can adapt continuously to the background environment, between the voice events there is normally enough time to let the filters converge to a changed background noise situation.

Keeping ST10 on high level during the convergence period enables a special mode. If the filters should adapt to a current background noise floor (without voice), then this can be done during this period. With ST10 returning to low level, the filter calculation is then based on a different filter pre-setting. This could be an advantage in special type of applications, where the signal is not continuously delivered to the HWVAD. In a DMIC system with continuous sampling, this convergence period is not required, bit HWVADST10[ST10] is just used to clear the interrupt flag.

A complete setup sequence for standard operation looks like this:

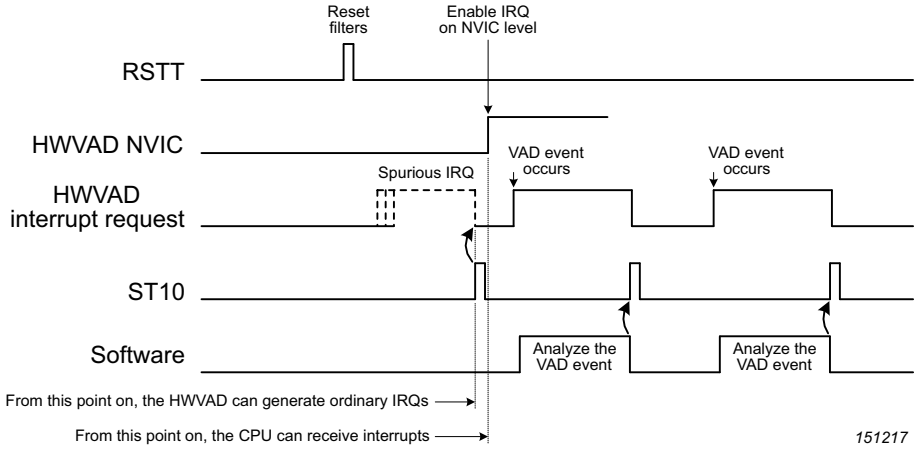


Fig 75. Complete HWVAD setup

1. Reset filters with bit HWVADRSTT[RSTT] and provide some time to let the filter converge to the signal conditions.
2. Pulse bit HWVADST10[ST10] to clear any spurious interrupts which were generated during bad filter conditions.
3. Enable HWVAD IRQ on NVIC level.
4. Process the VAD event in case of an interrupt, when finished clear the interrupt flag with a high pulse of ST10.

26.6.1.2 Extended operation

There are a few parameters which can be set to influence the behavior of the HWVAD. There is also an intermediate filter result value available, which can be used for proprietary software-based analysis.

26.6.1.2.1 Input gain setting

The 24-bit PCM input signal can be shifted left or right with the gain setting in the register HWVADGAIN. This increases or decreases the volume of the input signal for the HWVAD processing. Note that the reset value 0x05 equals a gain factor of 1, the signal is not shifted in either direction.

26.6.1.2.2 Filter result gain setting

The output values for the final equation can also have a gain factor within the hardware for determining the HWVAD result.

If [z8 * (THGS+1)] > [z7 * (THGN+1)], HWVAD_RESULT =1; else HWVAD_RESULT = 0;

These gain factors determine the proportion between the results of the signal and the noise filters. The values depend on the audio signal and noise environment, the reset values HWVADTHGN[THGN] = 0 and HWVADTHGS[THGS] = 4 are more suitable for a low noise environment. For noisy environment, the gain for THGN and THGS needs to be increased. In a typical voice recognition application HWVADTHGN[THGN] = 3 and HWVADTHGS[THGS] = 6 is a good starting point.

26.6.1.2.3 High pass filter setting

The setting in register HWVADHPFS can be used to adapt the filters to different background noise situations. In order to find the best setting, software could perform a rough spectral analysis of the audio signal.

For a background with more low-frequency content, HWVADHPFS[HPFS] should be set to 0x1. This is the standard use case. For environments where the low-frequency content is small, the filter can be set to 0x2.

26.6.1.2.4 Noise floor evaluation

The register HWVADLOWZ contains 2 bytes of the output of filter stage z7, which computes the noise floor. The characteristic of the filter block for voice applications is best for a value of 500 … 1000 in LOWZ. Software can tune the input gain to get the LOWZ value into this region.

Note: For power saving reasons, this register is not synchronized to the AHB bus clock domain. To ensure correct data is read, the register should be read twice. If the data is the same, then the data is correct, if not, the register should be read one more time. The noise floor is a slowly moving calculation, so several reads in a row can guarantee that register value being read can be assured to not be in the middle of a transition.

26.6.2 DMIC

The DMIC interface receives PDM data from one or two digital microphones and processes it to produce 24-bit PCM data. This data can be read by the CPU or DMA. Many aspects of DMIC operation can be controlled. A block diagram of one DMIC channel is shown in Figure 76.

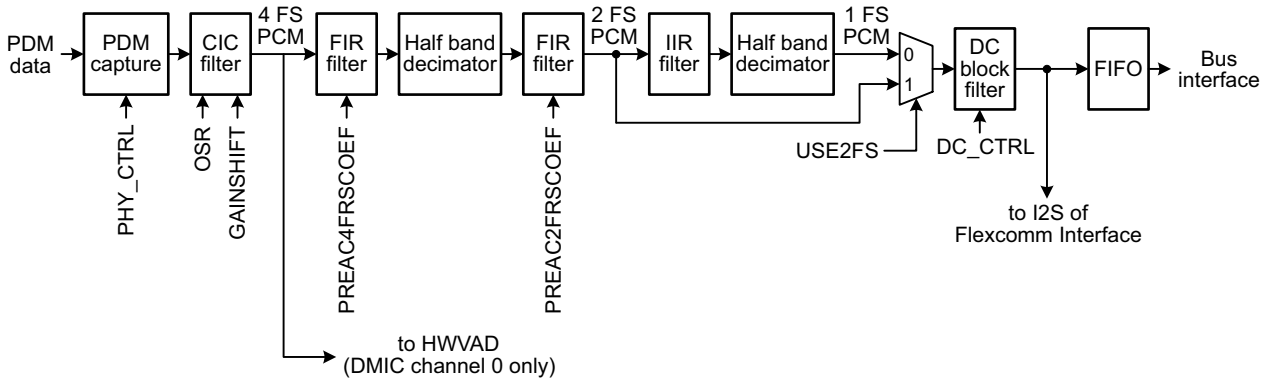


Fig 76. DMIC channel block diagram

26.6.2.1 Clocking and DMIC data rates

The DMIC interface operation is determined by 3 clock domains:

- DMIC interface base clock: supply clock for the peripheral block
- DMIC clock: sample clock for the digital microphone
- PCM sample rate: sample rate of the PCM data resulting from the PDM to PCM conversion

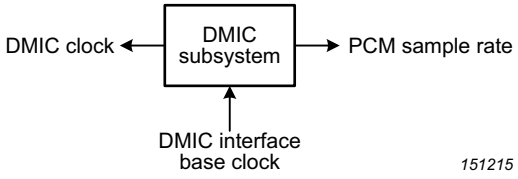


Fig 77. DMIC interface clock domains

The source for the base clock can be set in register SYSCON_DMICCLKSEL (see Chapter 6 “Clock Distribution”). Note that all of these clock sources may be divided by a factor of up to 256 by the DMIC clock divider, controlled by SYSCON_DMICCLKDIV (Chapter 6 “Clock Distribution”). The functions CLOCK_AttachClk and CLOCK_SetClkDiv can be used to configure the clocks.

Table 56. Base clock sources for DMIC interface peripheral

Source	Range	DMIC interface base clock	Note
FRO1MHz	1 MHz	≤ 1 MHz	
FRO12MHz	12 MHz	≤ 12 MHz	
FRO48MHz	48 MHz	≤ 24 MHz	
OSC32KHz	32 kHz	≤ 32 kHz	
MAIN_CLK	12-48 MHz	≤ 24 MHz	See Chapter 6 “Clock Distribution”
MCLK_IN	≤ 2 MHz	≤ 2 MHz	External clock

For the DMIC clock, the base clock divider values can be set in registers DIVHFCLK[0:1].

However, for power consumption reasons, it is preferable that the division to the required DMIC clock be done outside of the DMIC interface block (for example using register SYSCON_DMICCLKDIV).

The DMIC peripheral block is designed to run at a DMIC clock speed no faster than 6.144 MHz and with an input frequency no faster than 4 * 6.144 MHz = 24.576 MHz. With regards to power consumption the lowest possible frequency should be selected. This frequency very much depends on the application requirements. For a simple voice activity detection a sample rate of 200 kHz for the DMIC might be sufficient, for a good quality voice tag recognition the DMIC should be clocked at least with 800 kHz. Depending on the current operating mode of the application, the clocks can be set dynamically from one sample rate to the other.

For a glitch free reduction of the DMIC clock rate by factor 2 the DMIC interface contains dedicated circuitry. By setting bit PHY_CTRL0[PHY_HALF] and PHY_CTRL1[PHY_HALF], the DMIC clock is divided to half the frequency internally used for the filters. This enables an on-the-fly switching of the DMIC clock without affecting the operation of the filters. As long as the sample quality on half of the frequency is good enough for the application, for example in listening mode only, this helps to decrease the power consumption of the external digital microphone.

If the PDM interface operates during deep-sleep mode (always listening), then the presence of the clock source in this mode must be taken into account as well. For example, the PLL output is not present during deep-sleep mode, but the 12 MHz FRO is there.

In general, other clocks such as the 48 MHz FRO or the watchdog oscillator are available in deep-sleep mode. It depends on the use case whether a faster or slower clock provides any advantage to the system. At high PDM data rates, for example at 6 MHz, a 48 MHz clock will shorten the “awake” periods compared to 12 MHz operation. A trade-off between the sleep and the active periods, and the internal voltage required at the chosen clock rate, that determine which of the clocks perform better in terms of average power consumption.

The watchdog oscillator low-power operation can help to drive the power consumption down in simple voice detection setups. By running the DMIC interface on the slow watchdog oscillator frequency, the HWVAD feature can provide a first audio detection trigger signal to the system. Hereafter the sample rate as well as the processor performance is increased in order to run more sophisticated voice detection and/or voice recognition algorithms.

26.6.2.2 PDM to PCM conversion

The filter block for PDM to PCM conversion consists of four stages. It begins with a CIC filter (Cascaded-Integrator Comb filter) filter which is an optimized finite impulse response (FIR) filter combined with a decimator.The CIC filter converts the PDM stream from the digital microphone into PCM data with a given oversampling rate, set in registers OSR[0] and OSR[1] for each of the two channels. The second block performs a decimation by 2 and compensates for a roll-off at the upper limit of the audio band. The third block decimates the signal again by half, resulting in a PCM signal with the desired sample rate. A final DC filter removes any unwanted DC component in the audio signal.

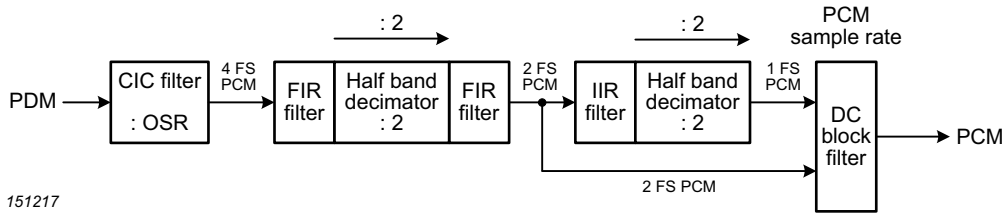


Fig 78. Principle structure of the PDM to PCM conversion

To achieve lower power consumption, the DC filter can be supplied with the 2FS instead of the 1FS signal, bypassing the second half band decimator filter. This reduces the required DMIC base clock by a factor of 2. This is done by setting the USE2FS[USE2FS] bit.

The PDM to PCM conversion block is designed for providing best results for PCM output signals with a 16 kHz sample rate, covering the enhanced 8 kHz speech band widely used in communication systems. However, other sample rates can be realized as well.

The final relation between the DMIC clock rate and the PCM audio sample rate is:

Table 57. DMIC input and output clock rates

2 FS mode	1 FS mode
PCM Sample rate = DMIC clock rate / (2 * OSR)	PCM Sample rate = DMIC clock rate / (4 * OSR)

Example: DMIC clock = 800 kHz, OSR = 25, 2 FS used

The 800 kHz DMIC data is downsampled by 25 times to 32 kHz. With the following half-band filter the final PCM sample rate is 16 kHz.

The FIR filter configurations are controlled by PREAC4FSCOEFx and PREAC2FSCOEFx. The following diagram shows the filter response for these filters.

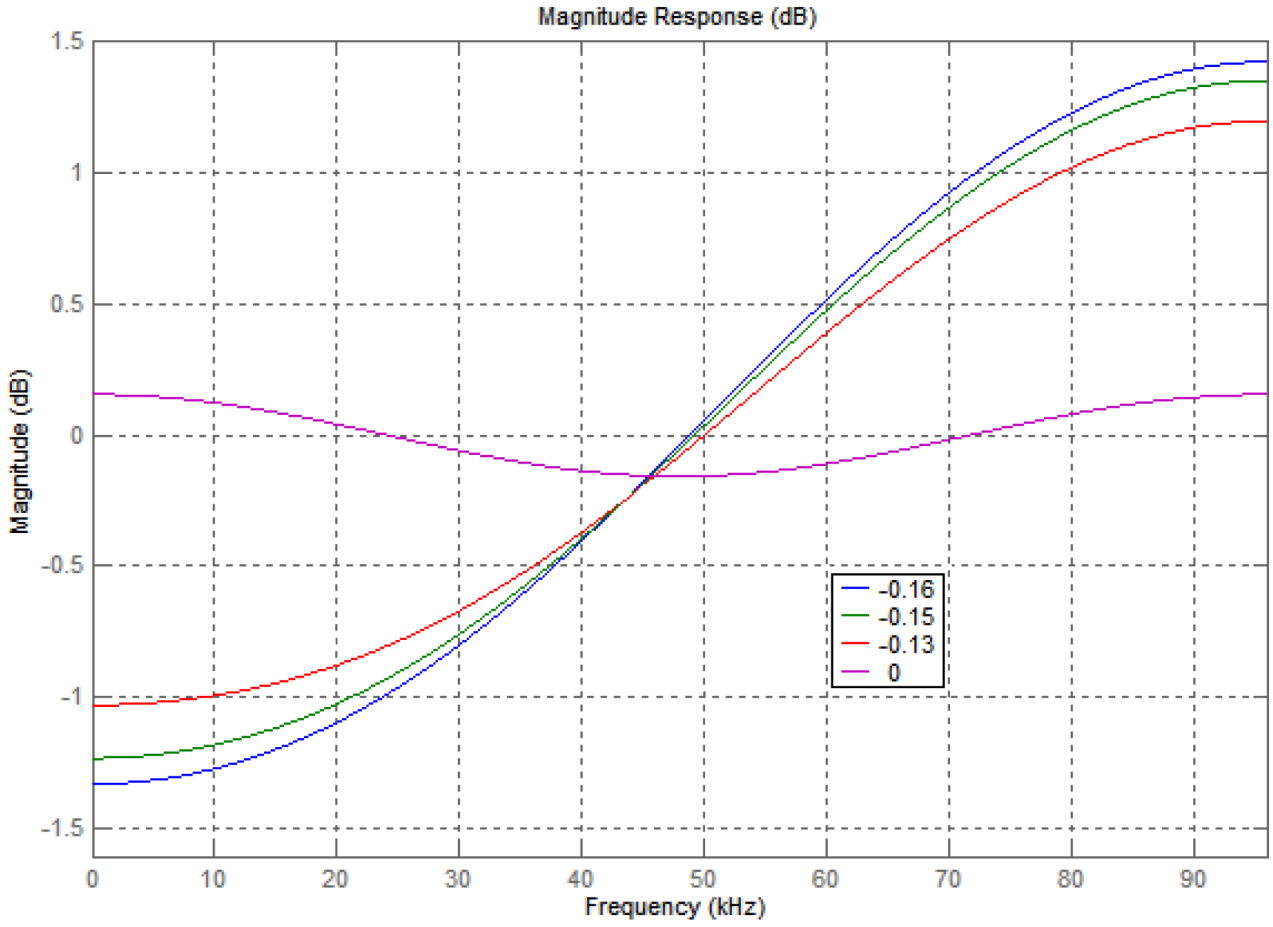


Fig 79. Pre-emphasis filter quantized response at 96 kHz

26.6.2.3 FIFO and DMA operation

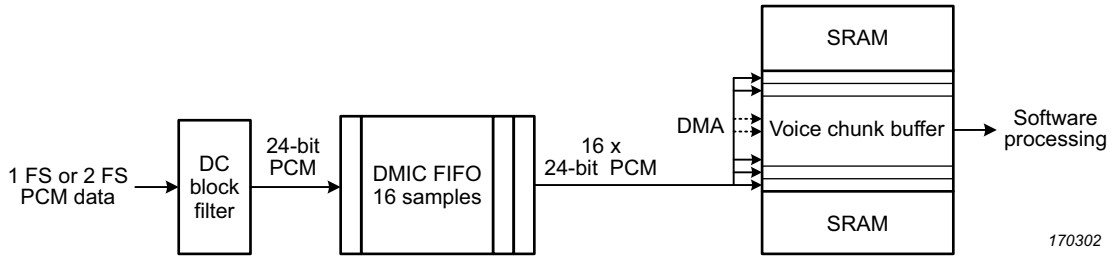


Fig 80. DMIC FIFO and DMA

The 24-bit wide FIFO of the DMIC interface consists of 16 entries for each of the two channels. The trigger level for the FIFO can be set in register FIFO_CTRLx individually for each channel.

The trigger level interrupt for the DMIC interface needs to be enabled on NVIC level and with bit INTEN in register FIFO_CTRLx. Bit DMAEN enables DMA operation. With each FIFO trigger level event the DMA performs a copy of the data from the FIFO into SRAM.

This data batching works without contribution of the core. When reaching the defined chunk buffer size, the DMA issues an interrupt to the Arm core for further processing of the data.

This also works when the device is in deep-sleep mode, as the FIFO event is able to wake up the required part of the hardware. After the DMA finished the job, the device will return into deep-sleep mode. The two DMIC channel DMA requests are connected to the DMA request input #15 and #16, see Table 25 “DMA requests & trigger muxes”.

Since each DMIC channel provides a separate DMA request, the most obvious configuration of DMA is to have left and right data in separate memory buffers. However, it is possible to configure the DMA controller to interleave left and right data if that is preferable in the application. To do this, the DMA is set up with a data size of halfword, but the next address written to is a word address distance away. The two descriptors would be started on consecutive halfwords. Data is delivered by the DMIC as left channel followed by right channel for each PCM stereo sample.

If more history data is required for a software algorithm, another DMA request can be set up, which copies the current chunk into a larger ring buffer structure. For algorithms like voice detection or voice recognition, this is key, in order to converge the software filters to the current background noise situation.

For operation without DMA, the dedicated DMIC FIFO interrupt can be enabled in order to inform the Arm core about the FIFO status.

Example:

- PCM output sample rate is 16 kHz
- The 32-bytes FIFO gets full every 1ms
- The DMA copies every 1ms the 32-bytes content of the DMIC FIFO to SRAM
- The DMA is configured to move 512 bytes (= 256 PCM samples) from the DMIC FIFO to SRAM before issuing a DMA interrupt
- Every 16 ms the 256 PCM samples are processed by the Arm core

26.6.2.4 Usage of the DMIC interface in power save modes

The DMIC interface can batch the serial PDM stream from a digital microphone in deep-sleep mode. This requires an appropriate base clock which is active in the respective power saving mode. The best fit for this base clock is the 12 MHz FRO, which provides a good trade-off between power consumption and performance. For lower power operation the watchdog oscillator can be used, taking into account that the clock is relatively inaccurate and the DMIC sample rate is rather low. At any time an external low-power clock, connected to pin MCLK, can be used.

In combination with the HWVAD, this provides lowest power consumption in listening mode. Except for the short periods with DMA activity, the MCU can remain in deep-sleep mode until the wave envelope detector of the HWVAD identifies an energy change event and issues an interrupt. With the DMA set to larger transfer sizes (maximum is 1024 transfers), there is quite some history data available for any type of software-based analysis of the data causing the HWVAD event.

This also enables the system to realize different strategies for dealing with a HWVAD event. A concrete analysis of the data could for example just be started when the HWVAD detected events over a longer time frame. This would avoid that the Arm core gets active on spurious noise. In case the decision has been taken to take the next step in data analysis, the history buffer still contains the complete PCM data sampled since the first event, nothing got lost. In average, the system can stay longer in power save mode if spurious events can be filtered out.

26.7 Software control

To use the functionality of the DMIC module, it is recommended to use software functions from within fsl_dmic.c.

27.1 How to read this chapter

The ADC controller is available on all JN5189(T)/JN5188(T) devices.

27.2 Features

- 12-bit successive approximation analog to digital converter.
- Input multiplexing among up to 8 pins (6 external inputs, 1 temperature sensor and V_{BAT}).
- A configurable conversion sequencer with configurable trigger
- Optional automatic high/low threshold comparison and “zero crossing” detection.
- 12-bit conversion rate of 190 kHz. Options for reduced resolution at higher conversion rates.
- Burst conversion mode for single or multiple inputs.
- Asynchronous operation. Asynchronous mode allows choosing ADC clock from FRO12M or XO32M.
- A temperature sensor is connected to ADC channel 7, see Chapter 28 “Temperature Sensor” for further details.
- Supply monitor is connected to ADC channel 6; this monitors V_{BAT}.

27.3 Basic configuration

Configure the ADC as follows:

- Set up the CTRL register.
- Use the ADC API to start up the ADC.
- The ADC block creates three interrupts which are connected to the NVIC: ADC_SEQ, and ADC_THCMP_OVR. The ADC_THCMP_OVR interrupt at the NVIC combines ADC_THCMP and ADC_OVR conditions from the ADC as described in this chapter. See Table 17 for interrupt numbers. The sequence interrupts can also be configured as DMA triggers through the INPUT MUX for each DMA channel and as inputs to the SCT.
- Use IOCON (Chapter 12) to connect and enable analog function on the ADC input pins.
- Pre-determined calibration data must be written into the ADC configuration registers after every reset or power cycle of the ADC. The ADC API function may be used for this. Note that the ADC may be power cycled in deep-sleep mode if it is not requested to stay on when these modes are invoked by the Chip_POWER_EnterPowerMode API.
- There are two options in the CTRL register to clock ADC conversions:
  - Use the system clock to clock the ADC in synchronous mode. This option allows exact timing of triggers.

– Use the ADC clock, determined by the SYSCON_ADCCLKSEL register and the SYSCON_ADCCLKDIV register. ADC clock should be at 4 MHz. Some clock sources are independent of the system clock, and may require extra time to synchronize ADC trigger inputs.

Configure the temperature sensor as follows:

- Select the temperature sensor as source for channel 7 of the ADC by writing the SEQ_CTRL[CHANNELS] bits to 0x80. In order to return ADC channel 7 to measuring its related device pin, write the SEQ_CTRL[CHANNELS] bits to 0x80.
- The digital temperature reading is available after an analog-to-digital conversion of ADC channel 7.

Remark: To convert the ADC conversion result into a temperature reading, use the API provided. This uses device specific calibration data stored in the device to increase the accuracy of the temperature reading.

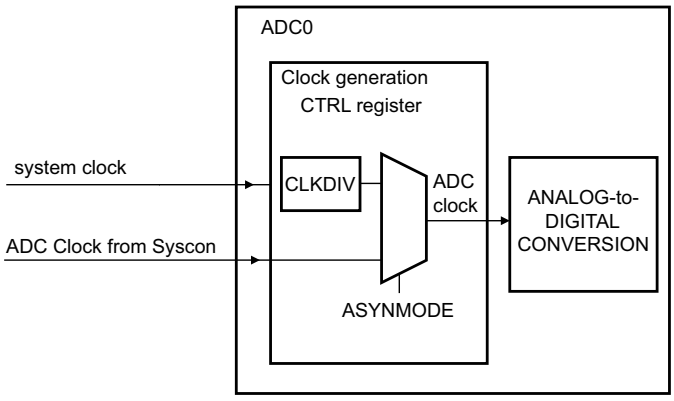


Fig 81. ADC clocking

27.4 Pin description

The ADC can measure the voltage on any of the input signals on the analog input channel. Digital signals must be disconnected from the ADC input pins when the ADC function is to be used by setting PIOx[DIGIMODE] = 0 on those pins. Additionally, the pull-up and pull-down resistors must be disabled.

Warning: If the ADC is used, signal levels on analog input pins must not be above the level of V_{BAT} at any time. Otherwise, ADC readings will be invalid. If the ADC is not used in an application, then the pins associated with ADC inputs can be used as digital I/O pins.

The ADC can be triggered by the Pin Interrupt PINT0 signal. This can be associated with a digital pin, see Chapter 15 “Pin Interrupt and Pattern Match (PINT)”. In addition to assigning the pin trigger to a pin, it must also be selected in the conversion sequence registers for each ADC conversion sequence defined.

The ADC can also be triggered by two of the outputs of the PWM module, PWM_OUT8 and PWM_OUT9. See Chapter 18 “Pulse Width Modulation (PWM)” for details of how these PWM signals are generated. In addition to enabling the PWM function it must also be selected in the conversion sequence registers for each ADC conversion sequence defined.

The processor can also generate an ADC trigger by asserting the transmit event, TXEV, signal by executing the SEV command. The conversion sequence register must also be configured to use this TXEV trigger.

Before the ADC can be configured and used, it is necessary to enable the two LDOs needed for the ADC and also enable the clocks to the ADC. This is performed by the ADC initialization function provided in the SDK. A delay of 230 µs is required to allow settling of the ADC LDOs to achieve full ADC accuracy.

Quicker startup is possible but with reduced accuracy. For a reduction of each bit of accuracy the startup time is reduced by 20 µs; this is possible down to a 7-bit ADC value. The driver code within the SDK will demonstrate how the initialization can be performed.

The ADC has 8 channels, the channel mappings are shown in Table 58.

Table 58. ADC channels

ADC channel	Function	Device Pin
0	ADC0	PIO14
1	ADC1	PIO15
2	ADC2	PIO16
3	ADC3	PIO17
4	ADC4	PIO18
5	ADC5	PIO19
6	Supply monitor	Internal function
7	Temperature sensor	Internal function

Recommended IOCON settings are shown in Table 59. See Chapter 12 for definitions of pin types.

Table 59: Suggested ADC input pin settings

IOCON bit(s)	ADC5 - ADC0 pins
11	SSEL: Set to 0
10	OD: Set to 0.
9	Slew1: set to 0.
8	FILTEROFF: Set to 1.
7	DIGIMODE: Set to 0.
6	INVERT: Set to 0.
5	Slew0: set to 0.
4:3	MODE: Set to 2.
2:0	FUNC: Select GPIO as the pin function.
General comment	Configure for analog input without pull-up or pull-down.

27.5 General description

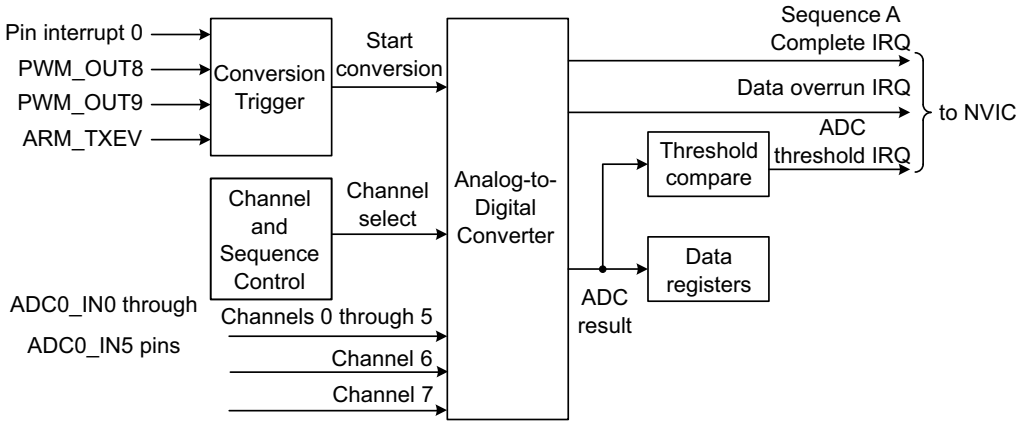


Fig 82. ADC block diagram

The ADC controller provides a great deal of flexibility in launching and controlling sequences of ADC conversions using the associated 12-bit, successive approximation ADC converter. ADC conversion sequences can be initiated under software control or in response to a selected hardware trigger.

Once the triggers are set up (software and hardware triggers can be mixed), the ADC runs through the pre-defined conversion sequences converting a sample whenever a trigger signal arrives until the sequence is disabled.

The ADC controller uses the system clock as a bus clock. The system clock or the asynchronous ADC clock (see Figure 81) can be used to create the ADC clock which drives the successive approximation process:

- In the asynchronous mode, an independent clock source is used as the ADC clock source without any further divider in the ADC. The ADC clock rate is 4 MHz as well.

The full scale output of the ADC is obtained when the ADC pin is at 3.6 V. However, voltage on the ADC pins must not exceed V_{BAT}. So to obtain the largest dynamic range of the ADC, it is necessary to operate with the maximum supply voltage possible. To convert from the ADC value to a voltage, it is necessary to multiply the ADC value by 3.6/4095.

27.6 Functional description

27.6.1 Conversion Sequences

A conversion sequence is a single pass through a series of ADC conversions performed on a selected set of ADC channels. Software can configure the conversion sequence which can be triggered by software or by a transition on one of the hardware triggers.

An optional single-step mode allows advancing through the channels of a sequence one at a time on each successive occurrence of a trigger.

27.6.2 Hardware-triggered conversion

Software can select which hardware trigger will launch each conversion sequence and it can specify the active edge for the selected trigger independently for each conversion sequence.

For each conversion sequence, if a designated trigger event occurs, one single cycle through that conversion sequence will be launched unless:

- The SEQ_CTRL[BURST] for this sequence is set to 1.
- The requested conversion sequence is already in progress.

If any of these conditions is true, the new trigger event will be ignored and will have no effect.

In addition, if the single-step bit for a sequence is set, each new trigger will cause a single conversion to be performed on the next channel in the sequence rather that launching a pass through the entire sequence.

27.6.2.1 Avoiding spurious hardware triggers

Care should be taken to avoid generating a spurious trigger when writing to the SEQ_CTRL register to change the trigger selected for the sequence, switch the polarity of the selected trigger, or to enable the sequence for operation.

In general, the SEQ_CTRL[TRIGGER] and SEQ_CTRL[TRIGPOL] should only be written when the sequence is disabled (while the SEQ_CTRL[SEQ_ENA] = 0). The SEQ_CTRL[SEQ_ENA] itself should only be set when the selected trigger input is in its INACTIVE state (as designated by the SEQ_CTRL[TRIGPOL] bit). If this condition is not met, a trigger will be generated immediately upon enabling the sequence - even though no actual transition has occurred on the trigger input.

27.6.3 Software-triggered conversion

There are two ways that software can trigger a conversion sequence:

1. **Start Bit:** Setting the corresponding SEQ_CTRL[START] . The response to this is identical to occurrence of a hardware trigger on that sequence. Specifically, one cycle of conversions through that conversion sequence will be immediately triggered except as indicated above.
2. **Burst Mode**: Set the SEQ_CTRL[BURST] bit. As long as this bit is 1 the designated conversion sequence will be continuously and repetitively cycled through. Any new software or hardware trigger on this sequence will be ignored.

27.6.4 Interrupts

The following interrupts can be generated by the ADC:

- Conversion-Complete or Sequence-Complete interrupt for sequencer
- Threshold-Compare Out-of-Range Interrupt
- Data Overrun Interrupt

Any of these interrupt requests may be individually enabled or disabled in the INTEN register. Note that the threshold and overrun interrupts share a slot in the NVIC.

27.6.4.1 Conversion-Complete or Sequence-Complete interrupts

An interrupt/DMA trigger can either be asserted at the end of each ADC conversion performed as part of that sequence or when the entire sequence of conversions is completed. The SEQ_CTRL[MODE] selects between these alternative behaviors.

If the SEQ_CTRL[MODE] bit for a sequence is 0 (conversion-complete mode), then the interrupt flag/DMA request for that sequence will reflect the state of the SEQ_GDAT[DATAVALID] bit. In this case, reading the SEQ_GDAT register will automatically clear the interrupt/DMA trigger.

If the SEQ_CTRL[MODE] bit for the sequence is 1 (sequence-complete mode) then the interrupt flag/DMA request must be written-to by software to clear it (except when used as a DMA trigger, in which case it will be cleared in hardware by the DMA engine).

27.6.4.2 Threshold-Compare Out-of-Range Interrupt

Every conversion performed on any channel is automatically compared against a designated set of low and high threshold levels specified in the THRn_HIGH and THRn_LOW registers. The results of this comparison on any individual channel(s) can be enabled to cause a threshold-compare interrupt if that result was above or below the range specified by the two thresholds or, alternatively, if the result represented a crossing of the low threshold in either direction. The mode is configured in the INTEN register.

This flag must be cleared by a software write to clear the individual FLAGS[THCMPn] flags.

27.6.4.3 Data Overrun Interrupt

This interrupt/DMA trigger will be asserted if any of the FLAGS[OVERRUNn] bits are set, visible in the OVERRUN0 to OVERRUN7 status bits in the FLAGS register. In addition, the SEQ_GDAT[OVERRUN] bit will cause this interrupt/DMA trigger if the SEQ_CTRL [MODE] bit is set to 0 (conversion-complete mode).

This flag will be cleared when the OVERRUN bit that caused it is cleared via reading the register containing it, e.g. if OVERRUN5 is set then it is necessary to read data from DAT[5] register to clear the overrun flag.

Note that the SEQ_GDAT[OVERRUN] bit is cleared when data related to that channel is read from either of the global data registers as well as when the individual data registers themselves are read.

27.6.5 Optional Operating Modes

There are three optional modes of ADC operation which may be selected in the CTRL register.

Four alternative ADC accuracy settings are available ranging from 12 bits down to 6 bits of resolution. Lowering the ADC resolution results in faster conversion times. A single ADC conversion (including one conversion in a burst or sequence) requires (resolution+3) ADC clocks when the minimum sampling period is selected. When reduced accuracy is selected, the unused LSBs of result data will automatically be forced to zero.

Two clocking modes are available, synchronous mode and asynchronous mode. The synchronous clocking mode uses the system clock in conjunction with an internal programmable divider. The main advantage of this mode is determinism. The start of ADC sampling is always a fixed number of system clocks following any ADC trigger. The alternative asynchronous mode (on chips where this mode is supported) uses an independent clock source. In this mode the user has greater flexibility in selecting the ADC clock frequency to better achieve the maximum ADC conversion rate without restricting the clock rate for other peripherals. The penalty for using this mode may be longer latency and greater uncertainty in response to a hardware trigger.

27.6.6 Offset and gain calibration algorithm

Before using the ADC, some initialization software is required to apply the pre-calculated calibration data words for ADC offset and ADC gain, stored in flash, need to be copied into dedicated control register.

The ADC cannot be utilized until the startup routine has completed. This initialization is performed the software initialization API function provided.

27.6.7 ADC vs. digital receiver

The analog ADC input must be selected via IOCON registers in order to get accurate voltage readings on the monitored pin. In the IOCON, the pull-up and pull-down resistors must be both disabled using the PIOn[MODE] bits. For a pin hosting an ADC input, it is not possible to have a have the digital function enabled and yet get valid ADC readings. Software must write a 0 to the related PIOn[DIGIMODE].

27.6.8 DMA control

The conversion sequence complete interrupt may also be used to generate a DMA transfer trigger. To generate a DMA transfer, the same conditions must be met as the conditions for generating an interrupt (see Section 27.6.4).

Remark: If DMA is used for a sequence, the corresponding sequence interrupt must be disabled in the INTEN register.

For DMA transfers, only burst requests are supported. The burst size can be set to one in the DMA_CFGn register. If the number of ADC channels is not equal to one of the other DMA-supported burst sizes (applicable DMA burst sizes are 1, 4, 8), set the burst size to one.

The DMA transfer size determines when a DMA interrupt is generated. The transfer size can be set to the number of ADC channels being converted. Non-contiguous channels can be transferred by the DMA using the scatter/gather linked lists.

27.6.9 ADC hardware trigger inputs

An analog-to-digital conversion can be initiated by a hardware trigger. The trigger can be selected for the conversion sequencer in the ADC SEQ_CTRL register by programming the hardware trigger input # into the TRIGGER bits.

Related registers:

- SEQ_CTRL register

Table 60. ADC0 hardware trigger inputs

Input #	Source	Description
0	PINT0	See Chapter 16 “Group GPIO Input Interrupt (GINT)”
1	PWM_OUT8	See Chapter 18 “Pulse Width Modulation (PWM)”
2	PWM_OUT9	See Chapter 18 “Pulse Width Modulation (PWM)”
3	ARM_TXEV	Transmit Event output from CPU

27.6.10 Sample and conversion time

The analog input from the selected channel is sampled at the start of each new A/D conversion. The default (and shortest) duration of this sample period is 2.5 ADC clock cycles. Under some conditions, longer sample times may be required. A variety of factors including operating conditions, the ADC clock frequency, the selected ADC resolution, and the impedance of the analog source will influence the required sample period.

The conversion time of the ADC is given by:

Tconv= [(1+ GPADC_TSAMP [4:0]) +7+nb_resol] .Tclk

For nb_resol = 12 (12 bit) , Tconv = (20 + GPADC_TSAMP [4:0]).Tclk

For the typical vale of GPADC_TSAMP [4:0]=0x14h (equivalent to 20d in decimal) Tconv = 40.Tclk (40 ADC clock cycle)

For the first conversion it will take 2 * Tconv. If the ADC input mux is unchanged (gpadc_sel_in_1v1_set[2:0]) then a following conversion will only take Tconv. However, each time the ADC input mux is changed, 2 * Tconv is needed.

27.7 Examples

The following examples are intended to show some of the ADC conversion operations and features supported. The APIs provided will help develop applications using this functionality. See Section 27.8 “Software control” for information on the SW driver.

27.7.1 Perform a single ADC conversion triggered by software

Remark: When ADC conversions are triggered by software only and hardware triggers are not used in the conversion sequence, follow these steps to avoid spurious conversions:

1. Before changing the trigger set-up, disable the conversion sequence by setting the SEQ_CTRL[SEQ_ENA] bit to 0.
2. Set the trigger source to an unused setting using the SEQ_CTRL[TRIGGER] bits. The value 3, for example, is not used on this device.
3. Set the SEQ_CTRL[TRIGPOL] bit to 1.

Once the sequence is enabled again, the ADC converts a sample whenever the SEQ_CTRL[START] bit is written to.

The ADC converts an analog input signal VIN on the ADC0 to ADC5 pins.

To perform a single ADC conversion for channel 1 using the analog signal on pin ADC1, follow these steps:

1. Enable the analog function on pin ADC1 via IOCON.
2. Configure the system clock to be 48 MHz and configure SYSCON_ADCCLKSEL[SEL] to 00b (XO32M), configure SYSCON_ADCCLKDIV[DIV] to 111b (8, to have 4 MHz).
3. Select the asynchronous mode by the CTRL[ASYNMODE].
4. Select ADC channel 1 to perform the conversion by setting the SEQ_CTRL[CHANNELS] bits to 0x2.
5. Set the SEQ_CTRL[TRIGPOL] bit to 1 and the SEQ_CTRL[SEQ_ENA] bit to 1.
6. Set the SEQ_CTRL[START] bit to 1.
7. Read the SEQ_GDAT[RESULT] bits for the conversion result. The SEQ_GDAT[DATAVALID] bit can be used to indicate when the ADC result is valid.

27.7.2 Perform a sequence of conversions triggered by an external pin

The ADC can perform conversions on a sequence of selected channels. Each individual conversion of the sequence (single-step) or the entire sequence can be triggered by hardware. Hardware triggers are either a signal from an external pin or an internal signal. See Section 27.6.9.

To perform a single-step conversion on the first four channels of ADC0 triggered by rising edges on pin PIO0, follow these steps:

1. Enable the analog function on pin ADC0 to ADC3 via IOCON.
2. Configure PINT0 to respond to PIO0, see Chapter 15 “Pin Interrupt and Pattern Match (PINT)” for details.
3. Configure the system clock and select asynchronous mode for ADC with FRO12M (SYSCON_ADCCLKSEL[SEL] to 01b) and SYSCON_ADCCLKDIV[DIV] to have clock at 4 MHz
4. Select the asynchronous mode by CTRL[ASYNMODE].
5. Select ADC channels 0 to 3 to perform the conversion by setting the SEQ_CTRL[CHANNELS] bits to 0xF.
6. Select trigger PINT0 by writing 0x1 to the SEQ_CTRL[TRIGGER].
7. To generate one interrupt at the end of the entire sequence, set the SEQ_CTRL[MODE] bit to 1.
8. Select single-step mode by setting the SEQ_CTRL[SINGLESTEP] bit to 1.
9. Enable the Sequence A by setting the SEQ_CTRL[SEQ_ENA] bit.

   A conversion on ADC channel 0 will be triggered whenever the pin PIO0 goes from LOW to HIGH. The conversion on the next channel (initially channel 1) is triggered on the next 0 to 1 transition of PINT0. The ADC0 interrupt is generated when the sequence has finished after four 0 to 1 transitions of PINT0.
10. Read the SEQ_GDAT[RESULT] bits for the conversion result. The SEQ_GDAT[DATAVALID] bit can be used to indicate when the ADC result is valid.

27.7.3 Perform a conversion in full speed

The goal is to have a full speed conversion of 1 input of ADC in channel 0.

1. Enable analog function for ADC0
2. Configure the system clock for ADC, 4 MHz clock is required and enable axync_adc_clk
3. Configure for shortest sampling time by setting GPADC_CTRL0[GPADC_TSAMP]=0x1. This requires the ADC source to have a low source impedance.
4. Select asynchronous mode by CTRL[ASYNMODE]
5. Select start_behavior mode by SEQ_CTRL[START_BEHAVIOUR]
6. Select ADC channel 0 to perform the conversion by setting the SEQ_CTRL[CHANNELS] to 0x1
7. Set the SEQ_CTRL[TRIGPOL] bit to 1 and the SEQ_CTRL[SEQ_ENA] bit to 1.
8. Select burst mode by SEQ_CTRL
9. At each new interrupt, the SEQ_GDAT is updated.

27.8 Software control

To use the functionality of the ADC module, it is recommended to use software functions from within fsl_adc.c.

28.1 How to read this chapter

The temperature sensor is available on all JN5189(T)/JN5188(T) devices.

28.2 Features

- Linear temperature sensor.
- Sensor output internally connected to the ADC channel 7 for temperature monitoring

28.3 Basic configuration

This section explains how the Temperature Sensor can be used. For a functional example see lpc_adc_basic.

- Enable the power to the temperature sensor by setting the ASYNC_SYSCON_TEMPSENSORCTRL[ENABLE] .
- Configure temperature sensor common mode output voltage setting ASYNC_SYSCON_TEMPSENSORCTRL[CM] = 0x2 for proper default operation
- To monitor the temperature continually, select the temperature sensor as source for channel 7 of ADC0. See Chapter 27. The digital temperature reading is available after an analog-to-digital conversion.
- The ADC reading must be converted into a temperature reading. To increase accuracy the sensor and ADC are calibrated during production, An API is provided to produce a temperature value; this performs the best configuration of the ADC for the purpose of the temperature sensor. The calibration data and other characteristics of the temperature and ADC are used to produce a high accuracy results.
- For highest accuracy, set ADCCLK mux source to be 32 MHz XTAL with a divider setting of 7, to give an ADCCLK of 4 MHz.
- The voltage range of operation of the ADC is set by ADC_GPADC_CTRL0[TEST]. In normal mode, the ADC can take an input voltage of 0 to 3.6 V, to V_{BAT} if this is lower. For the temperature sensor, the ADC must be configured in Unity Gain mode when the input voltage range is 0 to 0.9 V. Since the temperature sensor voltage output is within this range, the best accuracy is achieved. A consequence of this is that the temperature sensor can not be combined with the other ADC inputs as part of sequencer configuration. Also, safe practice is to set the mode back to normal mode after using the ADC with the temperature sensor.

28.3.1 Perform a single ADC conversion with the temperature sensor as ADC input

As mentioned in the previous chapter, the API should be used when performing temperature measurements. As a simple example of obtaining a temperature measurement, the following steps can be performed. In this case, the accuracy is not as high as that using the API.

To perform a single ADC conversion for ADC0 channel 7 using the temperature sensor output:

1. Enable the temperature sensor output as input to ADC channel 7.
2. Configure the system clock and the ADC for operation.
3. Select the asynchronous mode in the ADC_CTRL register.
4. Select ADC channel 7 to perform the conversion by setting the ADC_SEQ_CTRL[CHANNELS] bits to 0x80.
5. Set the ADC_SEQ_CTRL[START] bit to 1.
6. Read the SEQ_GDAT[RESULT] bits for the conversion result.
7. The AHI software may be used to generate the temperature value. In fact, the example driver will perform this sequencing as well as making corrections due to the calibration data, and using averaging to give the best result.

28.4 Pin description

The temperature sensor has no configurable pins.

29.1 How to read this chapter

Serial Wire Debug functionality is available on all JN5189(T)/JN5188(T) devices.

29.2 Features

- Supports Arm Serial Wire Debug mode for the Cortex-M4.
- Trace port provides Cortex-M4 CPU instruction trace capability. Output via a Serial Wire Viewer.
- Direct debug access to all memories, registers, and peripherals.
- No target resources are required for the debugging session.
- Breakpoints: the Cortex-M4 includes 6 instruction breakpoints that can also be used to remap instruction addresses for code patches. Two literal comparators that can also be used to remap addresses for patches to literal values.
- Watchpoints: the Cortex-M4 includes 4 data watchpoints that can also be used as triggers.
- Instrumentation Trace Macrocell allows additional software controlled trace for the Cortex-M4.

29.3 Basic configuration

The serial wire debug pins, SWCLK and SWDIO, are enabled by default.

29.4 Pin description

The tables below indicate the various pin functions related to debug. Some of these functions share pins with other functions which therefore may not be used at the same time. Trace using the Serial Wire Output has limited bandwidth.

Table 61. Serial Wire Debug pin description

Function	Type	Connect to	Description
SWCLK	In	PIO0_12	**Serial Wire Clock.** This pin is the clock for SWD debug logic when in the Serial Wire Debug mode (SWD). This pin is pulled up internally.
SWDIO	I/O	PIO0_13	**Serial wire debug data input/output.** The SWDIO pin is used by an external debug tool to communicate with and control the part. This pin is pulled up internally.
SWO	Out	PIO0_14, PIO0_17 or PIO0_21	**Serial Wire Output.** The SWO pin optionally provides data from the ITM for an external debug tool to evaluate.

The following setup is required to enable SWO output on GPIO PIO0_14 (FUNC5), PIO_17 (FUNC3) or PIO0_21 (FUNC6):

1. Write 0x0 to SYSCON_TRACECLKDIV to enable the Trace clock divider.

2. If the clock to the IOCON block is not already enabled, write 1 to SYSCON_AHBCLKCTRLSET0[IOCON_CLK_SET]. The clock must be enabled in order to access any IOCON registers.
3. Configure the IOCON function for the required GPIO which will be used; for instance configure PIO0_17 to be FUNC3 by writing FUNC=3 in the PIO[17] register.
4. Enable SWD by setting register SYSCON_CODESECURITYPROT[SEC_CODE]=0x87654320

29.5 General description

Serial wire debug functions are integrated into the CPU, with up to four breakpoints and two watchpoints.

Trace on the Cortex-M4 is supported via the Serial Wire Output.

29.6 Functional description

29.6.1 Debug limitations

Important: Due to limitations of the CPU, the part cannot wake up in the usual manner from deep-sleep mode during debugging.

When using debug mode, some reduced power modes do not reach their normal low-power state. Therefore power measurements should not be made while debugging, power consumption is higher than that during normal operation. During a debugging session, the watchdog does not stop. Therefore, if the watchdog is active, it will interrupt the debug session. Therefore, any debug build must not have any references to the WWDT block.

Also, during a debugging session, the System Tick Timer is automatically stopped whenever the CPU is stopped. Other peripherals are not affected.

29.6.2 Access to SWD

In some circumstances, it is not possible to access the debug port. Conditions that can lead to access being blocked are:

- application is running and flash control bit, F_DIS_SWD=1
- flash field JTAG_DIS=1
- application is running and flash control bit, F_DIS_SWD=0, but application has not enabled debug

There are some special device modes which are configured by the boot code. SWD is controlled in some of these modes, refer to Chapter 7 “Reset, Boot and Wakeup”.

Access to the processors using SWD is shown in the following figure.

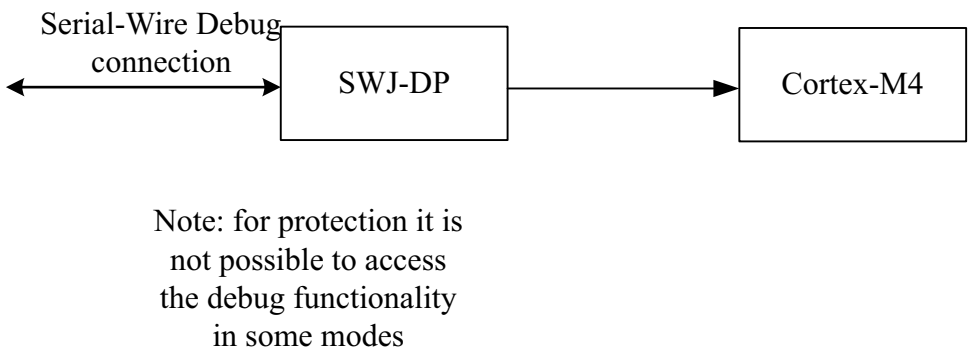


Fig 83. Serial Wire Debug connections

30.1 Introduction

This chapter describes the Flash Controller of the JN5189(T)/JN5188(T) device.

30.2 Features

- 32-bit AHB interface, to access the memory
- 32-bit APB registers interface (same clock domain as AHB)
- Auto initialization after reset
- ECC management, including single bit correction and error correction logging
- Supports automatic signature calculation of an address range
- Interrupts for end of command, ECC events, command error/failure

30.3 Basic configuration

The application does not need to directly control the flash controller. General configuration is managed by the boot code. Flash programming utilities are provided for modifying the flash contents.

The flash contents can be read as code or data from the flash region of the memory map. The register in the flash controller are accessible in a separate region of the address map.

When the flash is performing certain functions the flash memory cannot be accessed for other purpose, and an attempted read will cause a wait state to be asserted until the function is complete. These functions include erase, program and signature generation.

Access to the flash is zero wait state for CPU clock speeds of 32MHz of less. At 48MHz one wait state is required. Wait state configuration requires use of the SET_READ_MODE command.

To set clock to 48M FRO, set the wait state before changing clock frequency:

```
FLASH_SetReadMode(FLASH, true);

CLOCK_AttachClk(kFRO48M_to_MAIN_CLK);
```

To set lower speeds set the wait state after changing the clock frequency, for example for 32M FRO:

```
CLOCK_AttachClk(kFRO32M_to_MAIN_CLK);

FLASH_SetReadMode(FLASH, false);
```

31.1 How to read this chapter

Up to 2 SRAM controllers are available on all JN5189(T)/JN5188(T) devices. The number of SRAM controllers are dependent of JN5189(T)/JN5188(T) version.

31.2 Features

The two controllers are connected as the following:

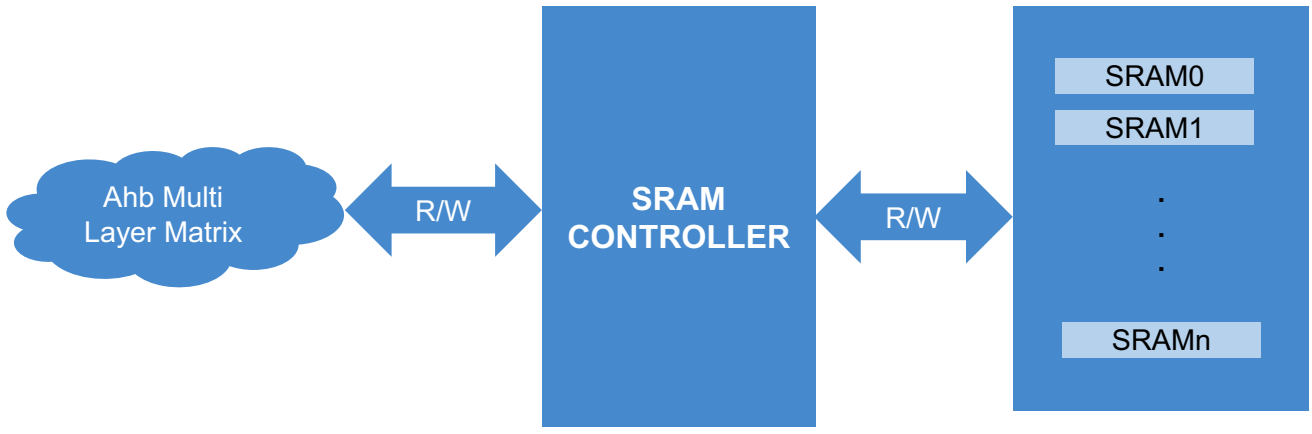


Fig 84. SRAM controller connection

The SRAM controllers have the following features:

- Zero wait state access (for all CPU frequencies)
- Controls SRAM memories in ACTIVE and POWERDOWN modes
- Can be configured through SYSCON_AHBCLKCTRL0[SRAMCTRL0] and SYSCON_AHBCLKCTRL0[SRAMCTRL1]

31.3 Basic configuration

Set the SYSCON_AHBCLKCTRL0[SRAMCTRL0] and SYSCON_AHBCLKCTRL0[SRAMCTRL1] to enable the clock to the SRAM controllers.

31.4 General description

31.4.1 SRAM controllers

The two RAM controllers are identical and are zero wait state memories (i.e no RY handshake or multi-cycle access). It includes a write buffer in order to boost performance (no stall upon write-read back to back).

The Table 62 lists the settings of each controller

Table 62. SRAM controller setting

Instance Name	Type	Slave Port/Slave Number	Description	Aperture Size	Start Address	End Address	Power Domain
SRAM-CTRL0	Slave	2/0	SRAM0 to SRAM7	88 KB	0x0400_0000	0x0401_5FFF	MCU
SRAM-CTRL1[1]	Slave	30	SRAM8 to SRAM11	64 KB	0x0402_0000	0x0402_FFFF	MCU

[1] SRAM-CTRL1 is disabled for JN5188 and JN5188T. To disable this controller, the PMC maintained it to RESET state.

SRAM Controller 0 controls 8 SRAMs composed as follow:

- SRAM0 (16 KB)
- SRAM1 (16 KB)
- SRAM2 (16 KB)
- SRAM3(16 KB)
- SRAM4 (8 KB)
- SRAM5 (8 KB)
- SRAM6 (4 KB)
- SRAM7 (4KB)

SRAM controller 1 controls 4 SRAMs composed as follow:

- SRAM8 (16 KB)
- SRAM9 (16 KB)
- SRAM10 (16KB)
- SRAM11 (16KB)

The Figure 1 “Main memory map” shows the JN5189(T)/JN5188(T) system memory map.

31.5 Power description

31.5.1 JN5189(T)/JN5188(T) IC power domains

The two SRAM controllers are on MCU power domain (PD_MCU) while memories are on different power domains PD_MEMx (see Section 31.5.2 “Memories power domains” and Figure 1 “Chip block diagram”.

31.5.2 Memories power domains

12 power domains are defined for memories:

- PD_MEM0 for SRAM0
- PD_MEM1 for SRAM1
- PD_MEM2 for SRAM2
- PD_MEM3 for SRAM3
- PD_MEM4 for SRAM4

- PD_MEM5 for SRAM5
- PD_MEM6 for SRAM6
- PD_MEM7 for SRAM7
- PD_MEM8 for SRAM8
- PD_MEM9 for SRAM9
- PD_MEM10 for SRAM10
- PD_MEM11 for SRAM11

In ACTIVE mode, SRAM memories are powered by LDO_CORE. If retention is needed (optional mode), SRAM memories are powered by LDO_MEM in power down mode.

The Table 15 “Possible power modes and state of power domains” shows more precisely the power domains states according to the power modes.

In active and sleep modes, controllers and memories are supplied and clocked by AHB clock.

In deep sleep mode, SRAM controllers are supplied but memories are in retention mode or shut-off (depending on the power mode type).

In power down mode, the internal SRAM values can be maintained if memory retention mode is enabled.

In deep power down, memories are shut off.

32.1 How to read this chapter

The ROM controller is available on all JN5189(T)/JN5188(T) devices.

32.2 Features

The ROM controller is connected as the following figure

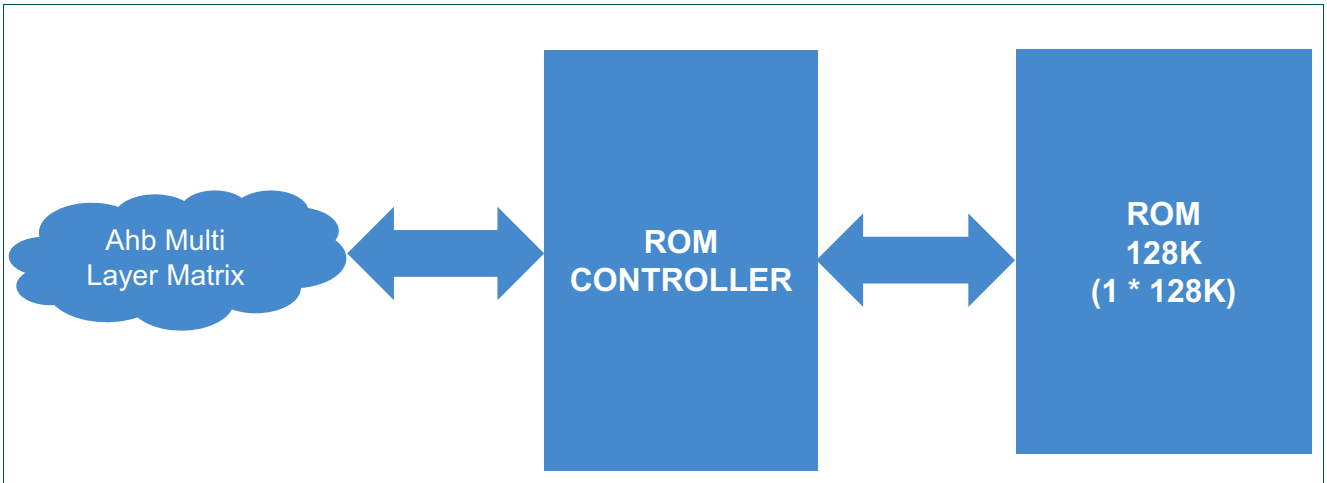


Fig 85. ROM controller connection

The ROM controller has the following features:

- Zero wait state access (for all CPU frequencies)
- Controls ROM memory in Active mode

32.3 Basic configuration

The application does not need to perform any configuration to use the ROM. Any required set-up is managed by the boot code.

32.4 General description

32.4.1 ROM controller

The ROM controller is zero wait state memory (i.e no RY handshake or multi-cycle access). It is the same controller as SRAM controllers except that it does not include a write buffer. So, it is fastest on timing and has a smallest gate count.

The Table 63 lists the settings of ROM:

Table 63. ROM setting

Instance Name	Type	Slave Port/Slave Number	Description	Aperture Size	Start Address	End Address	Power Domain
ROM	Slave	1/0	Code	128 KB	0x0300_0000	0x0401_FFFF	MCU

33.1 Introduction

The Hash-Crypt peripheral is divided into 3 parts, with details of each, controlled by configuration:

- The register interface, including two 512-bit buffers
- An AHB Master, used to read and in some cases write data.
- Hardware engines to perform specific symmetric crypto algorithms, including hashing and en/decryption:
  - SHA1 - 160 bit hash on 512-bit blocks
  - SHA2-256 - 256 bit hash on 512-bit blocks

The Hash block can generate triggers to the DMA block so that DMA data transfer can be efficiently performed.

33.1.1 Hashing

Hashing is used for 4 primary purposes (explained below):

- It is the core of a digital signature model, including certificates, such as for secure update.
- It is used with an HMAC to support a challenge/response or to validate a message.
- It can be used in a secure boot model, which verifies code integrity.
- It can be used to verify external memory has not been compromised.

A hash takes an arbitrarily large message/image (e.g. 1K or 1MB or more) and forms a relatively small fixed size "unique" number called a digest. The digest is unique in 3 ways:

- There is only one solution (digest) for a given input.
- Small and large changes will result in very different numbers.
- There is no predictable way to modify one input to result in a specific digest. That is, an attacker cannot add/insert/modify a message and get the same hash in any direct way; it is an extremely complex problem to add mods that result in the same digest.

The hash model works where data is fed to the Hash block and it forms a digest of size 160-bit or 256-bit. The data is fed 16 words at a time (512 bits) with the last block formatted per the SHA model:

- The last data must be 447 bits or less. If more, then an extra block must be created.
- After the last bit of data, a 1 bit is appended.
- Then, as many 0 bit are appended to take it to 448 bits long (so, 0 or more).
- Finally, the last 64 bits contain the length of the whole message, in bits, formatted as a word.

The data is fed by words from the processor, DMA, or hosted access; the words are converted from little-endian (Arm standard) to big-endian (SHA standard) by the block. So, no extra work is needed.

33.2 SHA hashing

Hashing is a way to reduce arbitrarily large amounts of data to a fixed size semi-unique result, called a digest. The SHA1 hash produces a 160-bit digest (5 words), and the SHA2 256 hash produces a 256-bit digest (8 words). The digest output has two notable aspects:

- Even a small change to the input message will cause a major change in the digest output.
- It is extremely hard to find changes that will produce the same digest. It is even harder to find changes to a message that preserve its size.

The above two properties make it useful for verifying whether a message is valid - whether corrupted intentionally or unintentionally. When used in conjunction with a public key infrastructure, it allows verifying correctness of the message along with correctness of the sender (author).

Hashing processes 512-bit blocks (16 words) and performs the hash in 80 cycles (SHA1) or 64 cycles (SHA2 256) per block. As many blocks as needed may be processed. The last block is special as explained above. It is up to the application to manage the last block.

The words are always big-endian. Since the Cortex-M processor uses little endian, this block reverses the bytes in the words written to the data register. This is because a hash is on bytes, so a string such as "abcd", when read as a word by the processor (or DMA) would be reversed into "dcba". The block reverses these to process correctly.

SHA hashing is explained in detail in the NIST/FIPS spec, see Ref. 3.

33.2.1 Performance of this block

This block has 4 possible use models. The difference in performance is often very dependent on the memory (e.g. flash with wait states vs. RAM) and then somewhat activity on the system buses. The 6 use models are:

- Single buffer with data loaded by the processor (Cortex-M).
  - The core writes 16 words (can use LDM and STM instructions for max speed) to kick off each SHA hashing. The block uses alias registers to support use of STM (writing to contiguous locations).
  - The hashing then runs for 64 or 80 cycles.
  - Interrupts or WFI or polling can be used to then feed the next 16 words until done.
- Double buffered load by processor.
  - Identical to the single buffer case above, except the processor can load the next 16 during the 64 or 80 cycles of hashing.
  - Unless high wait states, this would allow non-stop hashing.
- Single buffer with data loaded by DMA
  - The DMA loads the 16 words based on requests.

  - The hashing then runs for 64 or 80 cycles.
  - The DMA is then requested to load another 16. When the DMA is done, it interrupts the processor.
- Double buffered load by DMA
  - Identical to the single buffer case above, except the DMA can load the next 16 during the 64 or 80 cycles of hashing.
  - For the DMA using 4 cycles per transfer (at 0 wait state), this allows non-stop hashing.
- Single buffer with data loaded by an AHB bus master
  - The AHB master loads 16 words from Flash or RAM.
  - The hashing then runs for 64 or 80 cycles.
  - If the block count is not 0, it loads the next 16 words. This is repeated until the count reaches 0.
- Double buffered AHB bus master
  - Identical to the single buffer case above, except the master can load the next 16 during the 64 or 80 cycles of hashing.
  - Even at 3 wait states, this can perform non-stop hashing with SHA-2.
  - Even with 4 wait states, this can perform non-stop hashing with SHA-1.

33.3 Registers

All registers are to be accessed in word mode. The control register is used to enable the block, start a new hash, and to control use of DMA. The status and interrupts are linked. Interrupts are held until the condition is resolved, including waiting for data and completion. DMA is used to process data and will keep asking for more after every block - the DMA length should be used to control the number of blocks.

33.4 Initialization and Use

To perform hashing, the 1st step is to choose among 3 possible ways to get the data into the Hash block:

- Using Cortex-M4 using interrupts.
  - The WAITING (as well as ERROR) interrupt is configured.
  - On WAITING interrupt, the block is loaded by copying in the 16 words.
  - If more blocks to load, then the WAITING interrupt is retained. If last block, then the DIGEST interrupt is enabled instead.
- Using SDMA
  - The DMA is configured for up to 1K words (64 512-bit blocks) to be read from Flash or RAM. The Hash peripheral will control the DMA to feed It data as fast as it can.
  - If the double-buffered instance, it will allow loading another 16 words while processing the previous. This pipelined method allows continuous processing if not slowed by waitstates or contention.

- An interrupt is used to notify the processor when the DMA completes. That ISR can enable the DIGEST interrupt (as well as ERROR) to process the results. Or, it can configure the DMA for more data if needed.
- If the last block is to be constructed separately, then either the DMA can move those 16 words or the processor can do so via interrupt.

- Using AHB Master
  - The Peripheral's master is configured for the location in RAM (e.g. 0x0400_0000), or Flash (e.g. 0x0000_0000) and the count of blocks.
  - The DIGEST interrupt is enabled (along with ERROR). It will not fire until the last block is complete and the digest computed.
  - If the last block is to be hand constructed, then the ISR may load the constructed last block (or use the DMA) and it will be interrupted when the DIGEST is ready.

33.4.1 General initialization

The steps to setup the Hash block are as follows.

1. Take the block out of reset and start its clocks using the SYSCON register. See Section 10.5 “Accessing peripherals through internal buses”.

   Note: The Hash peripheral only uses the main AHB clock, so no special clocking or scaling is needed
2. Select SHA1 or SHA2 using the control register. Also write NEW, which will self clear.
3. If using:
   - CPU: write to INTENSET regsiter to enable the WAITING and ERROR interrupts.
   - DMA

     i.Setup the DMA (channel and config block), including enabling it.

     ii.Enable the DMA interrupt so the application will know when DMA is done.

     iii.Then set the DMA bit in the CTRL register.
   - AHB Master:

     i.Enable the INTENSET[DIGEST] and INTENSET[ERROR] interrupt.

     ii.Write to the MEMADDR register with the offset in Flash, CodeRAM or RAM

     iii.In the MEMCTRL register, set the MASTER control bit and also write the number of blocks to process into the COUNT field.

33.4.2 ISR for CPU use

Algorithm for ISR for CPU looks like:

- If ERROR interrupt occurs, there has been an overrun. This type of error typically occurs during development time and indicates a coding error.
- If WAITING, write 16 words into Hash peripheral. Fastest method is structure copy as shown below. If have no more blocks after this, disable WAITING interrupt using INTENCLR[WAITING] and then enable DIGEST interrupt by setting INTENSET[DIGEST].

  The fastest copy is usually:

  struct HASH_W {unsigned v[8];} *src, *dst;

```
src = (struct HASH_W * )memory_to_read_from; // use location in Flash or RAM
dst = (struct HASH_W * )HASH0_INDATA; // indata and aliases
dst[0] = src[0]; // 1st 8, usually using LDM/STM for best performance
dst[0] = src[1]; // 2nd 8, usually using LDM/STM for best performance
```

- If DIGEST, then process Digest register (e.g. read out the Digest result) and clear the interrupt using INTENCLR.

33.4.3 ISR for DMA use

The ISR for the DMA done uses:

- If all DMA data transfer has completed, write 1 to INTENSET[DIGEST]. The DIGEST interrupt will occur when the DIGEST is ready.
- It may be necessary to transfer one last block of data after the configured DMA transfer has completed. If this is needed, then either can write the 16 words now, or use the CPU ISR to transfer this last block.

When the DIGEST Interrupt occurs then process the DIGEST result, e.g. by reading out the DIGEST result.

The ERROR interrupt should not occur, but if it does then an overrun has occurred and the software must be checked.

33.4.4 ISR for AHB Master

The ISR for AHB Master is only for DIGEST or ERROR. An ERROR would be a bus error, so the algothism is:

- If ERROR, the master bus faulted. The MEMCTRL count indicates which block it was on, and the MEMADDR register indicates which location it was on when it failed.
- If DIGEST, then process DIGEST register (e.g. copy) and clear the interrupt using INTENCLR register.

33.5 Software control

To use the functionality of the Hash-Crypt peripheral, it is recommended to use software functions from within fsl_sha.c.

34.1 How to read this chapter

The SPI flash interface is available on all JN5189(T)/JN5188(T) parts.

34.2 Features

- Quad SPI Flash Interface (SPIFI) interface to external flash.
- Transfer rates of up to SPIFI_CLK/2 bytes per second.
- Supports 1-, 2-, and 4-bit bidirectional serial protocols.
- Half-duplex protocol compatible with various vendors and devices (see Table 66 “Supported QSPI devices”).
- A driver library available from NXP Semiconductors to assist in using the SPIFI.
- Four line cache to improve efficiency, when directly accessing data (memory mode) from external flash memory. Each cache line is 8 bytes.

34.3 Basic configuration

Initial configuration of the SPIFI peripheral is accomplished as follows:

- Power: Set SYSCON_AHBCLKCTRL0[SPIFI].

  Remark: On reset, the SPIFI is disabled.
- SPIFI clock: set up the SPIFI clock using the SYSCON_SPIFICLKSEL and SYSCON_SPIFICLKDIV registers
- Pins: Select SPIFI pins and pin modes through the relevant IOCON registers. IOs need to be configured for fast slew rate if operating above 10 MHz.

34.4 General description

The SPI Flash Interface (SPIFI) allows low-cost serial flash memories to be connected to the CPU with little performance penalty compared to parallel flash devices with higher pin count.

A driver API is available to manage the setup, programming and erasing of the flash. After an initialize call to the SPIFI driver, the flash content is accessible as normal memory using byte, halfword, and word accesses by the processor and/or DMA.

Many serial flash devices use a half-duplex command-driven SPI protocol for device setup and initialization. Quad devices then use a half-duplex, command-driven 4-bit protocol for normal operation. Different serial flash vendors and devices accept or require different commands and command formats. SPIFI provides sufficient flexibility to be compatible with common flash devices, and includes extensions to help insure compatibility with future devices.

Serial flash devices respond to commands sent by software or automatically sent by the SPIFI when software reads either of the two read-only serial flash regions in the memory map (see Table 66 “Supported QSPI devices”).

Table 64. Available pins and configuration registers

Memory	Address
SPIFI data	0x1000 0000 to 0x103F FFFF **Remark:** This is the address space allocated to the SPIFI for direct access. The area allocated allows a maximum of 4 MB of SPI flash to be mapped into the CPU memory space. In practice, the usable space is limited to the size of the connected device.

34.5 Pin description

Table 65: SPIFI Pin Description

Pin	Type	Description
SPIFI_CLK	O	Serial clock for the flash memory, switched only during active bits on the IO0/MOSI, IO1/MISO, and IO3:2 lines.
SPIFI_CSN	O	Chip select for the flash memory, driven low while a command is in progress, and high between commands. In the typical case of one serial slave, this signal can be connected directly to the device. If more than one serial slave is connected, software and off-chip hardware should use general-purpose I/O signals in combination with this signal to generate the chip selects for the various slaves.
SPIFI_IO0 or SPIFI_MOSI	I/O	This is an output except in quad/dual input data fields. After a quad/dual input data field, it becomes an output again one serial clock period after CSN goes high.
SPIFI_IO1 or SPIFI_MISO	I/O	This is an output in quad/dual opcode, address, intermediate, and output data fields, and an input in SPI mode and in quad/dual input data fields. After an input data field in quad/dual mode, it becomes an output again one serial clock period after CSN goes high.
SPIFI_IO[3:2]	I/O	These are outputs in quad opcode, address, intermediate, and output data fields, and inputs in quad input data fields. If the flash memory does not have quad capability, these pins can be assigned to GPIO or other functions.

34.6 Supported devices

Serial flash devices with the following features are supported:

- Read JEDEC ID
- Page programming
- at least one command with uniform erase size throughout the device

Table 66 shows a list of vendor QSPI devices which are supported. Other devices can be used and will run in basic single SPI mode at lower speed.

Remark: All QSPI devices have been tested at an operating voltage of 3.3 V.

Table 66: Supported QSPI devices

Manufacturer	Device name
AMIC	A25L512, A25L010, A25L020, A25L040, A25L080, A25L016, A25L032, A25LQ032
Atmel	AT25F512B, AT25DF021, AT25DF041A, AT25DF081A, AT25DF161, AT25DQ161, AT25DF321A, AT25DF641
Chingis	Pm25LD256, Pm25LD512, Pm25LD010, Pm25LD020, Pm25LD040, Pm25LQ032

Table 66: Supported QSPI devices

Manufacturer	Device name
Elite (ESMT)	F25L08P, F25L16P, F25L32P, F25L32Q
Eon	EN25F10, EN25F20, EN25F40, EN25Q40, EN25F80, EN25Q80, EN25QH16, EN25Q32, EN25Q64, EN25Q128
Gigadevice	GD25Q512, GD25Q10, GD25Q20, GD25Q40, GD25Q80, GD25Q16, GD25Q32, GD25Q64
Macronix	MX25L8006, MX25L8035, MX25L8036, MX25U8035[1], MX25L1606, MX25L1633, MX25L1635, MX25L1636, MX25U1635[1], MX25L3206, MX25L3235, MX25L3236, MX25U3235[1], MX25L6436, MX25L6445, MX25L6465, MX25L12836, MX25L12845, MX25L12865, MX25L25635, MX25L25735
Numonyx	M25P10, M25P20, M25P40, M25P80, M25PX80, M25P16, M25PX16, M25P32, M25PX32, M25P64, M25PX64, N25Q032, N25Q064, N25Q128
Spansion	S25FL004K, S25FL008K, S25FL016K, S25FL032K, S25FL032P, S25FL064K, S25FL064P, S25FL129P
SST	SST26VF016, SST26VF032, SST25VF064
Winbond	W25Q40, W25Q80, W25Q16, W25Q32, W25Q64

[1] Level translation circuitry, which might affect performance, is required for these parts.

The following devices lack one or more of these features and are not supported:

- Elite: F25L004, F25L008, F25L016.
- Eon: 25B64.
- SST: 25VF512, 25WF512, 25VF010, 25WF010, 25LF020, 25VF020, 25WF020, 25VF040, 25WF040, 25VF080, 25WF080, 25VF016, 25VF032.

34.7 SPIFI hardware

The SPIFI has a base address for the registers and a base address for the memory area in which the serial Flash connected to the SPIFI can be read.

The first operation with the serial Flash is Read JEDEC ID, which is implemented by most serial Flash devices. Depending on the device identity code returned by the serial Flash in this operation, device-specific commands are used for further operation. Programming and other operations on the serial Flash can be performed using a software AHI functions or using the register interface.

34.8 Register description

The SPIFI function is controlled and monitored through the registers. An overview of the registers follows.

34.8.1 SPIFI control register

The SPIFI control register controls the overall operation of the SPIFI and should be written before any commands are initiated

34.8.2 SPIFI command register

Writing to the Command Register can initiate the transmission of a new command. If the command requires additional data such as an address or intermediate data then this must be provided before the command is issued. If the command contains input data, software can read it from the Data Register after writing to this register.

34.8.3 SPIFI address register

Before writing a command that includes an address field to the Command register, software should write the address to this register. The most significant byte of the address is sent first.

34.8.4 SPIFI intermediate data register

Before writing a command to the Command register that requires specific intermediate byte values, software should write the value of the bytes to this register. The least significant byte of this register is sent first. If more than four intermediate bytes are specified in the Command register, 0s are sent after the 4th byte.

The main use of this register with current serial flash devices is to select the no-opcode mode (continuous read) using the byte value 0xA5, and canceling this mode using 0xFF.

Many devices that require dummy (delay) bytes don't care about their contents, in which case this register need not be written

34.8.5 SPIFI cache limit register

The SPIFI hardware includes caching of previously-accessed data to improve performance. Software can write an address within the device to this register, to prevent such caching at and above that address. After Reset this register contains the allocated size of the SPIFI memory area, so that all possible accesses are below that value and are thus cacheable

34.8.6 SPIFI data register

After initiating a command that includes a data output field by writing to the Command Register, software should write output data to this register. Store Byte instructions provide one data byte, Store Halfword instructions provide two bytes, and Store Word instructions provide 4 bytes of output data. Store commands are waited if the FIFO is too full to accept the number of bytes being stored. For Store Halfword and Store Word, the least significant byte is sent first.

After initiating a command that includes a data input field by writing to the Command Register, software should read input data from this register. Load Byte instructions deliver one data byte to software, Load Halfword instructions deliver two bytes, and Load Word instructions deliver 4 bytes of input data. Load commands are waited if a command is in progress and the FIFO does not contain the number of bytes being loaded. For Load Halfword and Load Word commands, the least significant byte is received first

CMD[DATALEN] bytes should be read from or written to this register. If such a (read or write) command needs to be terminated before that time, software should write a 1 to the STAT[RESET] bit to accomplish this. If software attempts to read or write more data than was specified in CMD[DATALEN], a Data Abort exception will occur.

In polling mode (see the CMD[POLL]), one byte must be read from this register because the poll mechanism writes the matching byte.

This register is not used for commands initiated by reading the flash address range in the memory map. In DMA transfers in peripheral to-or-from-memory mode, the address of this register should be used as the peripheral address.

34.8.7 SPIFI memory command register

Before accessing the flash area of the memory map, software should set up the device. After optionally writing to the Intermediate Data register, software should write a word to this register to define the command that is used to read data. Thereafter data can be read from the flash memory area, either directly or by means of a DMA channel.

Writing to this register will be ignored when a command is in progress or while data has yet to be written or read from the FIFO for a command issued. Use the STAT[MCINIT] bit to verify that the hardware is in Memory mode. A successful write to this register sets the SPIFI into Memory mode. The content of this register is identical to that of the Command Register, except for the DATALEN (not used), POLL, DOUT, and FRAMEFORM bits.

34.8.8 SPIFI status register

This register indicates the state of the SPIFI.

34.9 Functional description

34.9.1 Data transfer

Serial SPI uses the signals SPIFI_CLK, SPIFI_CSN,SPIFI_MISO, and SPIFI_MOSI, while quad mode adds the two IO signals SPIFI_SIO[3:2].

The SPIFI implements basic, dual, and quad SPI in half-duplex mode, in which the SPIFI always sends a command to a serial flash memory at the start of each frame. (A frame is the sequence of bytes transmitted during one period with CS LOW.) In general, commands start with an opcode byte although some serial flashes allow a no-opcode mode in which commands start with the address to be read. In write commands, the SPIFI sends all of the data in the frame, while in read commands, the SPIFI sends the command, and then the serial flash sends data to the SPIFI.

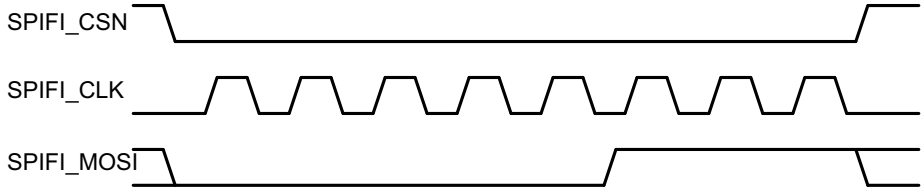
Classic SPI includes four modes (mode 0 to mode 3), of which the SPIFI and most serial flashes implement modes 0 and 3. In mode 0, the SCK line is LOW between frames while in mode 3 it is HIGH. In mode 0, the SPIFI drives the first data bits from the time that it drives CS LOW, and drives the rest of the data on falling edges of SCK. In mode 3, the SPIFI drives SCK LOW one-half clock period after it drives CS LOW, and drives data on the falling edge of SCK. In either mode the serial flash samples the data on the rising edges of SCK.

The same scheme (transmitter changes data on falling edges of SCK, receiver samples data on rising edges) is maintained for the entire frame, including read data sent by the serial flash to the SPIFI.

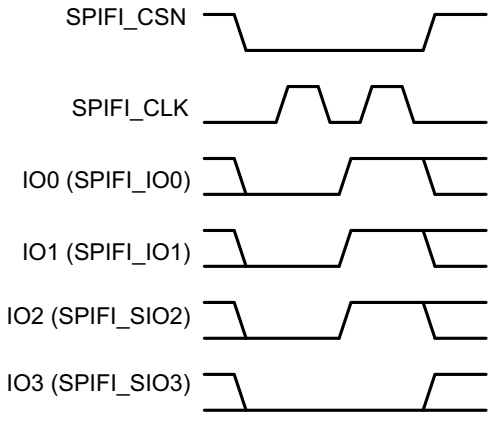
The SPI protocol avoids all issues of set-up and hold times between the clock and data lines by using half of the SCK period to transmit the data. For high clock speeds, it is necessary to sample read data using a feedback clock. The CTRL[FBCLK] bit enables the feedback clock from the SCK pad sampling method. This provides the best possible timing margin for both read and write data under the opposite-edge scheme.

But maximizing clock frequency is of such importance that further improvement is sometimes needed, by means of using the whole serial clock period to transmit data. This choice is enabled for read data by setting the CTRL[RFCLK] bit. When this bit is 1, the SPIFI samples data on the falling edge of the serial clock that follows the rising edge which is normally used. CTRL[RFCLK] and CTRL[FBCLK] and CTRL[MODE3] should not all be 1 because in this case there would be no falling edge of the feedback clock to capture the last bit of a frame.

Consult the data sheet of the serial flash device to be used for the formats of the commands that it supports. Figure 86 shows commands consisting of an opcode field only, sent in SPI and quad modes. All fields are multiples of 8 bits long. Bytes are sent with the most significant bit first in SPI mode, and the most significant 4 bits first in quad mode.



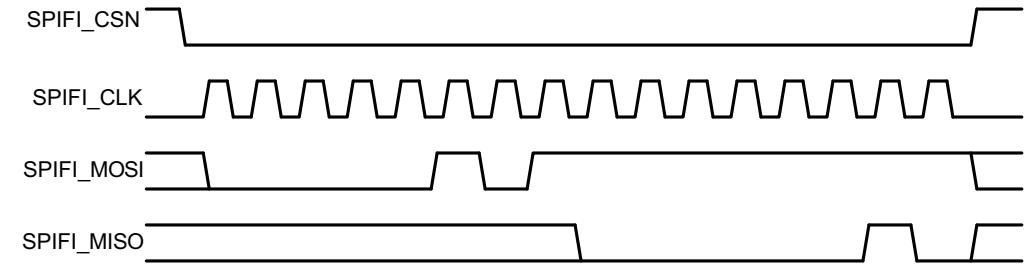
Opcode-only command in SPI mode



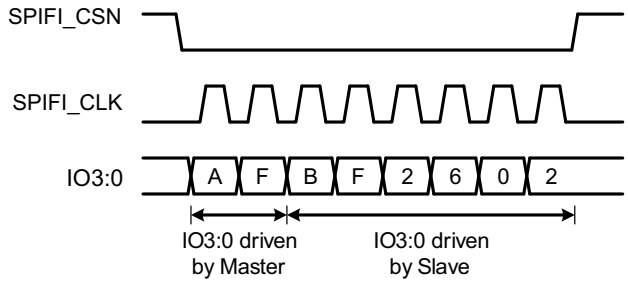
Opcode-only command in quad mode

Fig 86. Opcode only commands

Figure 87 shows a command that reads 1 byte from the slave in SPI mode and a command that reads 3 bytes from the slave with the opcode and input data fields both in quad mode.



Opcode 05, input data 02, SPI mode



Opcode AF, input data BF 26 02, both fields quad mode

Fig 87. Read commands

In quad mode, the IO3:0 lines are driven by the SPIFI in opcode, address, intermediate and output data fields, and driven by the flash memory in input data fields. In address fields the more significant bytes are sent first.

34.9.2 Software requirements and capabilities

During device set-up, software should initialize the external serial flash device using those commands that place it in its highest-performance mode. When this sequence is complete, software should write the command that will be issued in response to a read from the serial flash region of the memory map, to the Memory Command Register. If software attempts to read the flash region after Reset, power-up, or writing the Command Register without writing the Memory Command Register thereafter, the SPIFI responds with an Abort error.

After writing the Memory Command Register, the contents of the flash would appear to the system as memory mapped. This enables data access from the serial flash over AHB.

SPIFI has two operational modes:

- Memory Mode - whereby the contents of the FLASH are memory mapped in the chip.
- Command Mode - whereby the user can manually construct command sequences for the flash.

SPIFI cannot switch over from Memory Mode to Command mode and vice versa without writing 1 to the STAT[RESET] bit and polling until it is cleared by hardware to ensure that the current mode has been aborted.

The SPIFI has a small cache for accesses to the serial flash region of the memory map. The cache is only used in Memory mode and can be disabled.

Because the SPIFI is an AHB device, software or a DMA channel can read bytes, halfwords, or words from the flash region.

Reads from the flash region are delayed by deasserting the register interface HREADY signal when necessary, until the requested bytes are available to be read.

In Memory mode, SPIFI prefetches sequential addresses in order to improve performance.

If no AHB accesses have taken place for a period specified by the CTRL[TIMEOUT], the SPIFI will deassert CS. Once a new access occurs that requires a new fetch of data, the SPIFI will reassert CS and send a new command to fetch the required data. This is done in order to save power in the SPI flash device.

If software reads or writes more data from the Data Register than was configured in the CMD[DATALEN] field or reads or writes when no command was issued, the SPIFI hardware issues an abort exception.

When the serial flash needs to be programmed or erased, software should not write to the flash region of the address map. Instead, it should write the appropriate sequence of commands to the Command, Address, and Data registers. When an actual erase or program operation is under way in the serial flash device, software should write a Read Status command (with the CMD[POLL] bit set) to the Command register. Thereafter:

- If CTRL[INTEN] is 1, the SPIFI will interrupt the processor when the erase or write operation (and thus the Read Status command) completes.
- If not, software can continually or periodically read the Status register until it indicates that the Read Status command is complete.

When erasing or programming completes, software can do further programming or erasing, or return to normal (memory mode) operation.

34.9.3 Peripheral mode DMA operation

The SPIFI inserts wait states when necessary during read and write operations by the core to maintain synchronization between core accesses and serial data transfer with the serial flash. This mechanism is all that is needed for load and store accesses and for memory-to-memory transfers by a DMA channel.

The peripheral mode is a mode that supports DMA transfers in which the SPIFI acts as a peripheral and drives a request signal to the DMA channel to control data transfer. This mode does not necessarily move data faster than memory-to-memory operation, but it may be advantageous in systems in which software controls dynamic transfer of code and/or data between the serial flash and RAM on an as-needed basis. The advantage is that clock cycles are not lost to wait states, and thus the overall operation of the AHB is more efficient.

The DMA controller should be programmed to present word operations at the fixed address of the Data Register to have a burst size of one transfer. The SPIFI drives the DMA request to the DMA controller.

To use this mode, software should write the Command register to start the command and program a DMA channel as described above to transfer data between the Data register and RAM. The SPIFI asserts the DMA request when:

- CTRL[DMAEN] is 1.
- STAT[MCINIT] is 0.
- There are at least 4 bytes in the FIFO for a read operation, or at least 4 empty byte locations in the FIFO for a write/program operation.

34.10 Software control

To use the functionality of the SPIFI module, it is recommended to use software functions from within fsl_spifi.c.

35.1 Introduction

This device embeds a hardware IP that - combined with appropriate software - can be used to generate true random numbers with the highest levels of quality (FIPS140-2, AIS31,P2/PTG.3, etc).

The RNG feature is thus a combination of a hardware IP with a dedicated application software.

Such complementary software is required, in particular for these aspects:

- cryptographic post-processing
- entropy generation and monitoring

Using less sophisticated software, target for the RNG feature should be lowered to AIS31,P2/PTG.2 or even AIS31,P1/PTG.1.

A basic software should not target more than AIS20,K3/DRG.2. The difference being the way entropy sources are activated and monitored.

Note that in all cases, generated numbers will pass testsuites such as DieHard.

35.2 Functionality and usage

The TRNG embedded module is a true random number generator based on at least 2 sources of entropy:

- phase noise of imprecise clocks coming from embedded oscillators
- start-up state of a large number of internal flip-flops after power-up

This TRNG uses as inputs 1 imprecise clock, FRO32M, plus 1 precise clock, XTAL32M, and a system clock.

No TRNG certification has been initiated nevertheless NXP internal test pass for several advanced checks such as classical test suites and statistics computation on entropy sources.

TRNG deliver random number by reading register RANDOM_NUMBER. Successive random numbers delivered by TRNG should pass most test suites including DieHard or NIST SP800-22 or FIPS_140-1. With regards to initial value after power-up, a concatenation of first random numbers being read will pass the test suite requirement as well.

The hardware monitors the generation of entropy within the block. This is reflected in the COUNTER_VAL[REFRESH_CNT] field. It is possible to wait until this register indicates that enough entropy has been created and then read the random number from the RANDOM_NUMBER register. However, the COUNTER_VAL[REFRESH_CNT] does not rise linearly with time; instead it rises logarithmically. Generally, it is not practical to use this method to create the random number. It will give True Random number performance,

but it will take too long. A software procedure using this hardware entropy accumulation register is presented below and gives a usable method for generating random numbers which have True Random number performance.

Alternatively, it is also possible to just read random numbers from the RANDOM_NUMBER register without any delays. This will generate pseudo random numbers which should still be able to pass classical random number test suites. This method is quicker but at the expense of some randomness.

Refer to your NXP local FAE support for advanced use of the TRNG.

The software procedure using the entropy creation is presented below. It has two steps:

1. Initialization Step. This must be performed on a device power-up and also wake-up from power-down or deep power-down.
2. Generate Number Step. This is performed each time a new random number is required.

Initialization Step:

1. Set the SW parameter REF_CHI_SQUARED equal 2.
2. activate all clocks. Keep registers set to default values.
3. activate CHI computing with ONLINE_TEST_CFG[ACTIVATE] = 1.
4. loop on polling ONLINE_TEST_VAL while MIN_CHI_SQUARED>MAX_CHI_SQUARED.
5. if MAX_CHI_SQUARED > REF_CHI_SQUARED, program ONLINE_TEST_CFG[ACTIVATE] = 0 (to reset), then increment COUNTER_CFG[SHIFT4X] then back to step 3

Generate Number Step:

This routine creates a 32-bit random number, referred to as NewRand.

1. activate all clocks and keep CHI computing active
2. Read RANDOM_NUMBER register and assign to NewRand
3. Loop 32 times
   a. Wait until COUNTER_VAL[REFRESH_CNT] field is non-zero; this indicates that at least one bit of entropy has been created.
   b. Update NewRand value with the result of [NewRand XORed with the current value read from the RANDOM_NUMBER register].
   c. Reading the RANDOM_NUMBER register causes the COUNTER_VAL[REFRESH_CNT] to reset back to 0.

After this sequence, the NewRand value will be a new true random number; the routine waits for a total of 32-bit fresh entropy and so 32 true random bits are created.

Initial entropy

Typically, initial entropy is estimated to 32 bits for this IC, knowing that internal states are defined on at least 196 bits.

Self-checking for entropy creation

Checking initial entropy, for total failure or low quality:

There is no hardware self-checking mechanism.

Some software procedure can be implemented:

- Software can use the same procedure described above to read initial number in order to ensure a minimum entropy accumulation after power-up
- Software may store initial values in non volatile memory and compute some statistics

Checking run-time entropy:

- total failure: if no clock, no random number will be generated and this will halt the system bus leading to an exception.
- low quality run-time entropy: use the embedded CHI computing to detect very poor quality of entropy source.

35.3 Software control

To use the functionality of the RNG module, it is recommended to use software functions from within fsl_rng.c.

36.1 General description

The Control Interface is a card interface for smart card reader. The Contact Interface supports both synchronous and asynchronous 5 V (class A), 3 V (class B) and 1.8 V (class C) smart cards.

This User Manual describes the digital part module of the Contact Interface. The digital part controls an external analogue part to manage the card activation and deactivation sequences, see Section 36.4 “System integration” for a system diagram. The digital part ensures smart card communication via an embedded USART (Universal Asynchronous Receiver Transmitter) taking care of ISO7816 and EMV (Europay, MasterCard and Visa) requirements. The Contact Interface is controlled by software via APB slave interface.

36.2 Features

Functional features:

- Support of Class A (5 V), Class B (3 V) and Class C (1.8 V) contact smart cards, with suitable external analog device
- Compliant with ISO7816 standard
- Specific ISO USART with APB access for automatic convention processing, variable baud rate through frequency or division ratio programming error management at character level for T=0 and extra guard time register
- FIFO for 1 to 32 characters in both reception and transmission mode
- Parity error counter in reception mode and in transmission mode with automatic re-transmission
- Card clock generation (up to frequency of its input clock clk_ip)
- Card clock stop (at HIGH or LOW level)
- Supports the asynchronous protocols T=0 and T=1 in accordance with ISO7816 and EMV
- Versatile 24-bit time-out counter for Answer To Reset (ATR) and waiting times processing
- Specific Elementary Time Unit (ETU) counter for Block Guard Time (BGT): 22 in T=1 and 16 in T=0
- Supports synchronous cards

36.3 Functional description

The digital part of the Contact Interface is composed of the following blocks:

- APB slave interface
- Registers
- Sequencer

- Timers
- ATR counter
- ISO USART

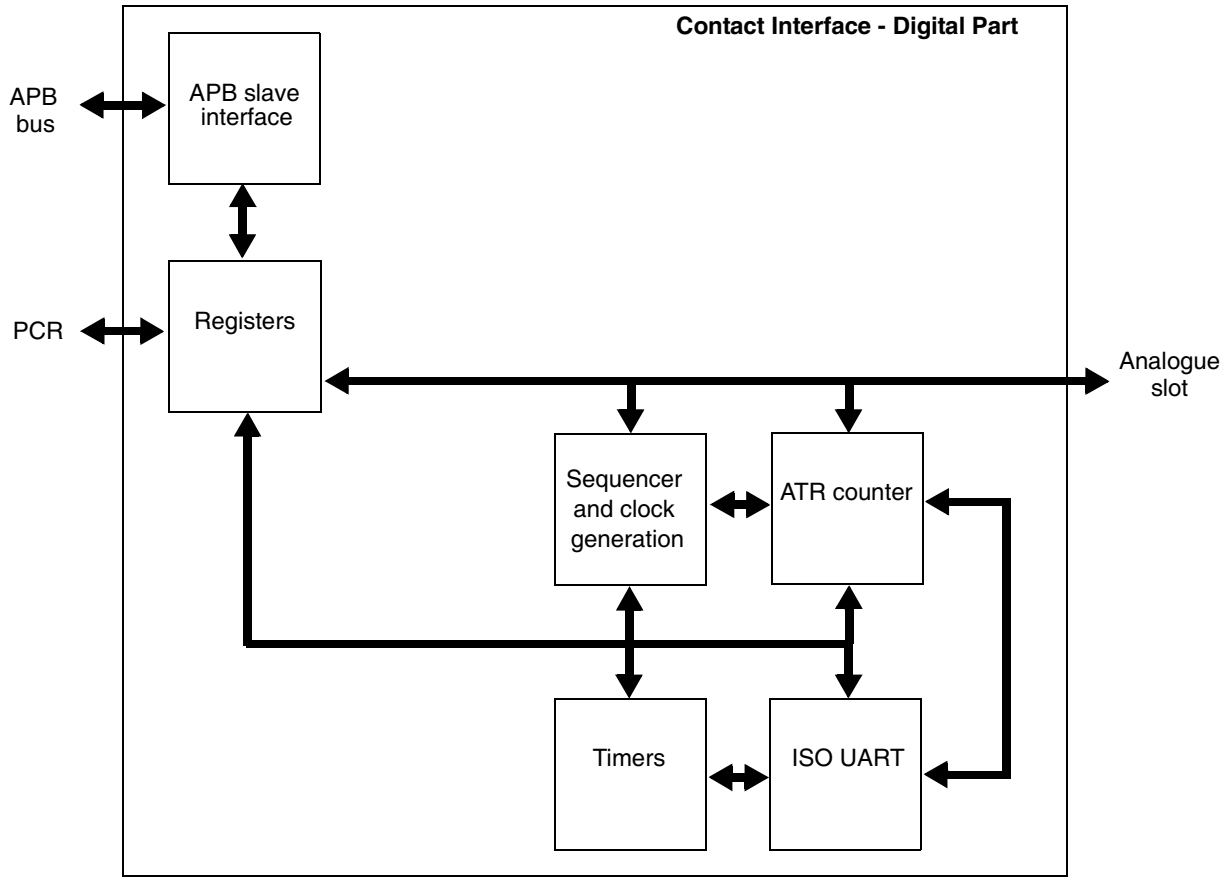


Fig 88. Block diagram of Contact Interface (Digital Part)

The Contact Interface is a card interface for smart card reader. The Contact Interface supports both synchronous and asynchronous 5V, 3V and 1,8V smart cards. The digital part embeds a sequencer and clock generator to control the use of the interface. The ATR counter manages RST and checks the ATR (Answer To Reset) of the asynchronous cards. The digital part ensures smart card communication via an embedded ISO USART (Universal Asynchronous Receiver Transmitter) taking care of ISO7816 and EMV requirements. The Timers count ETU with different available modes enabling to process some waiting times like BWT, WWT. The Contact USART is controlled via its registers, accessible by software through the APB slave interface.

36.3.1 Register interface

Reserved and Illegal access to this block will trigger a hardfault exception. The following table shows the response to illegal accesses and accesses to RFU (reserved for future) registers. All these conditions cause the hardfault.

Table 67. Register accesses causing a hardfault

Access Type	APB interface response
Write to a read-only register	No effect. Register not updated
Read from a write-only register	Read 00000000h
Write to an RFU address	No effect
Read from an RFU address	Read 00000000h

Remark: an access attempt (both read and write transfer) in user mode to a register only accessible when not in user mode (see registers descriptions) is seen as an access to an RFU address.

36.3.2 Registers

Registers block enables to control the Contact Interface, it is accessed via the APB slave interface.

Table 68. APB response to reserved and illegal accesses

Name	Width (bits)	Access	Reset value	Description
ct_ssr_reg	2	R/W	00000001h	Slot Select Register
ct_pdr1_lsb_reg	8	R/W	00000074h	Programmable Divider Register (LSB) slot 1
ct_pdr1_msb_reg	8	R/W	00000001h	Programmable Divider Register (MSB) slot 1
ct_fcr_reg	8	R/W	00000001h	FIFO Control Register
ct_gtr1_reg	8	R/W	00000000h	Guard Time Register slot 1
ct_ucr11_reg	6	R/W	00000000h	USART configuration Register 1 slot 1
ct_ucr21_reg	8	R/W	00000000h	USART configuration Register 2 slot 1
ct_ccr1_reg	6	R/W	00000000h	Clock Configuration Register slot 1
ct_pcr_reg	8	R/W	000000C0h	Power Control Register
ct_ecr_reg	8	R/W	000000AAh	Early Answer Counter Register
ct_mclr_lsb_reg	8	R/W	00000074h	Mute card Counter RST Low register (LSB)
ct_mclr_msb_reg	8	R/W	000000A4h	Mute card Counter RST Low register (MSB)
ct_mchr_lsb_reg	8	R/W	00000074h	Mute card Counter RST High register (LSB)
ct_mchr_msb_reg	8	R/W	000000A4h	Mute card Counter RST High register (MSB)
ct_urr_reg	32	R	00000000h	USART Receive Register
ct_utr_reg	32	W	00000000h	USART Transmit Register
ct_tor1_reg	8	W	00000000h	Time-Out Register 1
ct_tor2_reg	8	W	00000000h	Time-Out Register 2
ct_tor3_reg	8	W	00000000h	Time-Out Register 3
ct_toc_reg	8	R/W	00000000h	Time-Out Configuration Register
ct_fsr_reg	6	R	00000000h	FIFO Status Register
ct_msr_reg	3	R	00000002h	Mixed Status Register
ct_usr1_reg	6	R	00000000h	USART Status Register 1
ct_usr2_reg	8	R	00000000h	USART Status Register 2
RFU	0	R	0	RESERVED

36.3.3 Sequencer

The digital sequencer manages activation and deactivation sequences.

To perform the sequences, the SSR[SEQ_EN] bit must be set.

The sequencer is mainly composed of a FSM (finite state machine) four states: inactive_state, activation_state, active_state, deactivation_state.

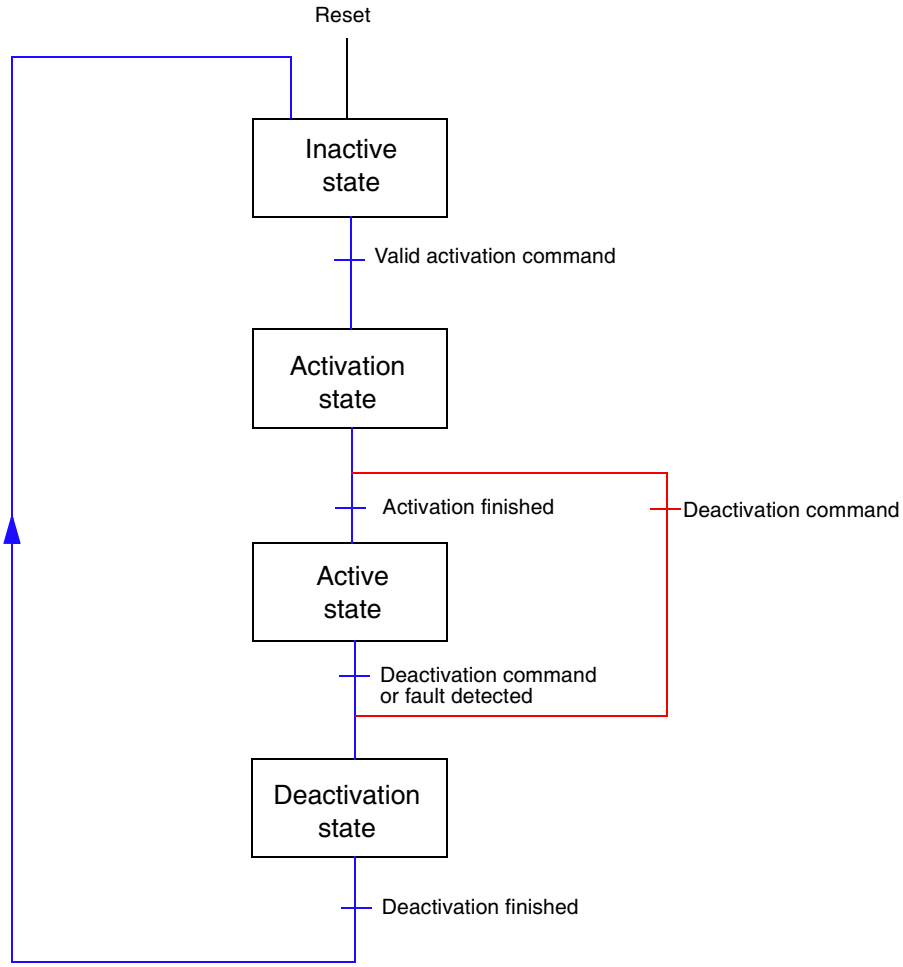


Fig 89. Sequencer FSM block diagram

The normal path is the blue one. There is an additional possible path that we call special case. Here is the description of the blue path.

- Inactive state

  This state is entered after power-on reset.

- Activation state

  Before the host starts a card session, it should set CRR1[SAN] bit to choose between asynchronous or synchronous card. Then, the host can set PCR[START] bit to logic level one. If there is no software reset, the activation sequence can occur.

The sequencer controls when the I/O is enabled (enable_io), when the CLK starts (enable_clk) and RST is enabled (enable_rst). The time T below is about 24 µs.

In synchronous mode, after 24T/2, since enable_rst is at logic level one, RST is the copy of PCR[RSTIN] bit. In asynchronous mode, PCR[RSTIN] is controlled by the ATR counter (see Section 36.3.4 “ATR Counter”).

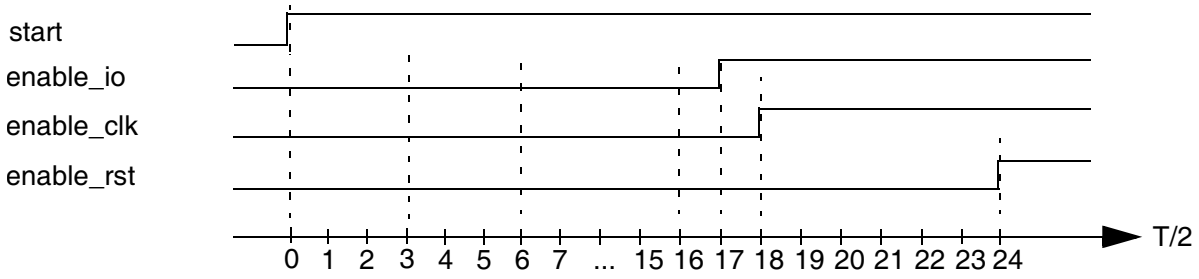


Fig 90. Activation Sequence

- Active state

  This state is entered after the activation sequence is completed.

- Deactivation state

  When a card session is completed, the host set PCR[START] bit to logic level zero. Then, a deactivation sequence is performed. ISO7816_RST is set to logic level zero (enable_rst), ISO7816_CLK is stopped (enable_clk), ISO7816_IO is disabled (enable_io). At t=0, the state machine goes back to the state inactive_state.

  Special cases:

  The activation sequence can be stopped by a deactivation command (PCR[START] bit set at logic level zero). The activation sequence is stopped whether or not it has completed and the deactivation sequence is started (it is the red path Figure 89). Three time windows have to be considered:

  - The activation sequence is stopped between 0 and the rising of enable_io on Figure 90. The deactivation sequence will start from 4T/2 on Figure 91 and go to 0.
  - The activation sequence is stopped between the rising of enable_io and the rising of enable_rst on Figure 90 The deactivation sequence will start from 6T/2 on Figure 91 and go to 0.
  - The activation sequence is stopped after the rising of enable_rst on Figure 90. It is the normal case: the deactivation sequence will start from 7T/2 on Figure 91 and go to 0.

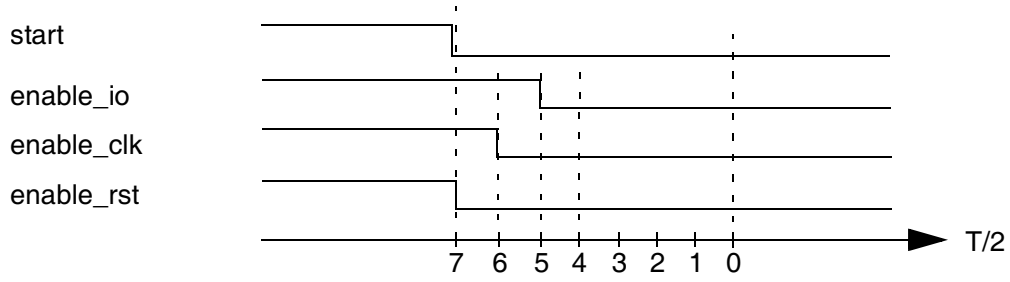


Fig 91. Deactivation Sequence

36.3.3.1 Card 1 clock circuitry

The digital card 1 clock circuitry generates the card 1 clock from the clk_ip clock. It manages card 1 clock dynamic frequency switches. The card 1 clock generation is composed of several stages. The first one generates the clk_ip frequency division. The second stage manages the clock stop. It consists in clock gating on both clock edges to stop the clock either at logic level one or zero. The last stage multiplexes the result of the second stage with the clock in synchronous mode (CCR1[SHL] bit value), which finally gives the card 1 clock. Because the card 1 clock is only needed during a card session, it is gated to logic level zero when the session is not active in order to improve consumption.

The card 1 clock frequency may be chosen at fclk_ip , fclk_ip /2, fclk_ip /3, fclk_ip /4, fclk_ip /5, fclk_ip/6, fclk_ip /8 or fclk_ip /16. In addition, the card 1 clock can be put at logic level zero and one (clock stop feature). The choice is done via ACC2, ACC1, ACC0, CST, SHL bits in ct_ccr1_reg register. See Table 69.

Table 69. Asynchronous card clock settings

ACC2 - ACC0	CST	SHL	SAN	Card clock
000	0	—	0	Clk_ip
001	0	—	0	Clk_ip/2
010	0	—	0	Clk_ip/3
011	0	—	0	Clk_ip/4
100	0	—	0	Clk_ip/5
101	0	—	0	Clk_ip/6
110	0	—	0	Clk_ip/8
111	0	—	0	Clk_ip/16
—	1	0	0	Logic 0
—	1	1	0	Logic 1

In case of synchronous card (bit CCR1[SAN]=1), the card clock is the copy of CCR1[SLH] bit (see the registers description). See Table 70.

Table 70. Synchronous card clock settings

ACC2 - ACC0	CST	SHL	SAN	Card clock
—	—	0	1	Logic 0
—	—	1	1	Logic 1

The frequency change is synchronous, which means that during transition, no pulse is shorter than 45% of the smallest period and that the first and last clock pulse around the change has the correct width.

When changing dynamically the card 1 clock frequency from a clk_ip frequency division to another one, the change occurs at the next card clock rising edge.

The card 1 clock circuitry generates card 1 clock and its buffer enable signal enable_clk. enable_clk is derived from the internal signal generated by the sequencer, it is synchronous to the frequency change.

The dynamic change (while activated) of CCR1[SAN] bit is not supported. The choice between asynchronous and synchronous card must be done before activating.

36.3.4 ATR Counter

The sequencer manages the activation and deactivation sequences. In addition to the sequencer, the ATR counter is used to manage RST and check the asynchronous cards ATR on the full slot. In case of synchronous cards, RST is controlled by the host via PCR[RSTIN] bit and the card ATR is not checked. The operating mode (asynchronous or synchronous) has to be selected by the host (see the registers description).

The ATR counter block is composed of two counters. One checks the early answer and the other checks if the card is mute.

The early answer counter is composed of a fixed part that counts up to 200d CLK cycles. An additional part counts up to ECR[7:0] value CLK cycles (see the registers description). The default value of ECR[ECR] field is 170d, which gives a total default count of 370d CLK cycles. The additional configurable count enables to follow a potential standard change.

The mute counter counts up to MCRL[15:0] (made up of the 16 bits in the MCRL_MSB and MCRL_LSB registers) value CLK cycles when RST is LOW and up to MCRH[15:0] (made up of the 16 bits in the MCRH_MSB and MCRH_LSB) value CLK cycles when RST is HIGH (see the registers description). The default value of MCRL[15:0] and MCRH[15:0] bits is 42100, which gives a default count of 42100 CLK cycles. The value chosen for MCRL[15:0] bits can be different from the one of MCRH[15:0] bits. This configurable count enables to support ISO7816 and EMV compliant cards and to follow a potential standard change.

To an asynchronous card activation and ATR, first, the host starts the activation (PCR[START]) after having configured the slot 1 (activation voltage). The sequencer performs the activation sequence. The DCDC converter is started, then VCC goes to logic level one, I/O is enabled and CLK starts. RST is at logic level zero.

Then the ATR counter checks the following steps:

1. If a start bit is detected on I/O during the first 200 CLK cycles, it is ignored and the count goes on.
2. If a start bit is detected while RST is at logic level zero between 200d and 42100 (or the value written in MCRL[15:0] bits) CLK cycles, the bits USR1[EARLY] and USR1[MUTE] are set to logic level one. RST will remain at logic level zero, it is up to the host to decide whether accepting the card or not.

3. If no start bit has been detected until 42100 (or the value written in MCRL[15:0] bits) CLK cycles, RST is set to logic level one.
4. If a start bit is detected within the first 370 (or 200 + the value written in ECR[ECR] bits) CLK cycles with RST at logic level one, the bit USR1[EARLY] is set to logic level one.
5. If the card does not answer before 42100 (or the value written in MCRH[15:0] bits) CLK cycles with RST at logic level one, the bit USR1[MUTE] is set to logic level one.
6. If the card answers within the correct time window, the CLK cycles count is stopped and the host may send commands to the card

The picture below shows the timings checked by the ATR counter:

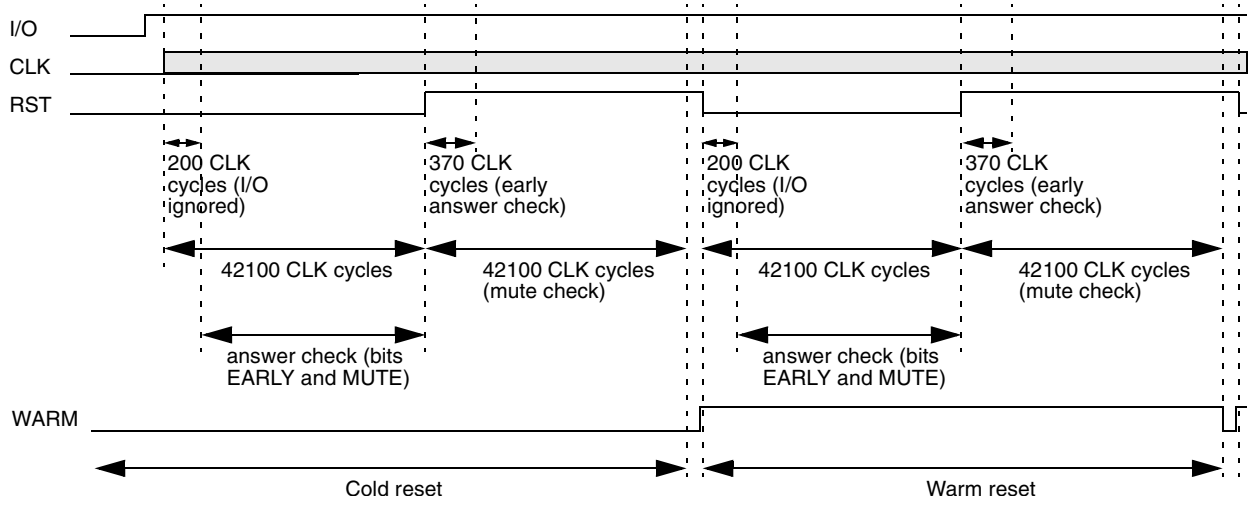


Fig 92. ATR counter timings checked (with default values)

The bits USR1[EARLY] and USR1[MUTE] signal an interrupt when set to logic level one (see the registers description).

The sequence mentioned above relates to a cold reset (left part of the Figure 92). If the card is mute (has not answered), the host may start a warm reset by setting PCR[WARM] bit to logic level one (see the registers description). Then, the ATR counter set RST to logic level zero and performs the same timing checks (right part of the Figure 92).

36.3.5 Timers

This block counts with different available modes a number of ETU.

It consists of two parts:

- tor_latch
- mode

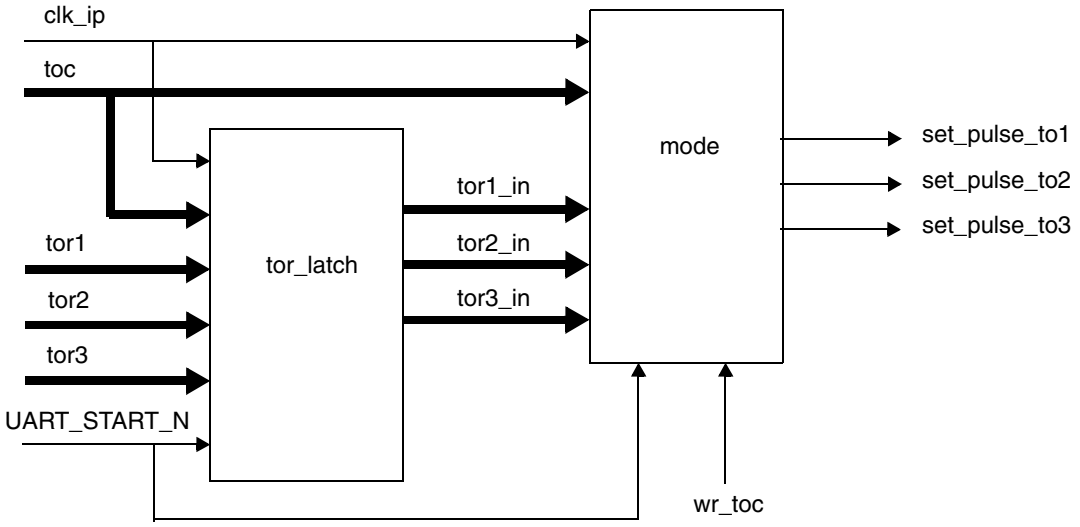


Fig 93. Timer block diagram

Remark: USART_START_N indicates a start bit has occurred.

Remark: wr_toc indicates a write in TOC register has occurred. The three registers TOR1, TOR2 and TOR3 form a programmable 24-bit ETU counter, or two independent counters (16-bit and 8-bit), or three independent 8-bit counters. The value to load in TOR1, TOR2 and TOR3 registers is the number of ETUs to count.

It exists three different modes to count the number of ETUs loaded in TOR registers (Autoreload, Software-triggered and Triggered on start bit on I/O). This is configured in TOC register.

36.3.5.1 tor_latch

If the timers are configured in START_TRIG mode, then this block latches the value written in TOR1[TOR1], TOR2[TOR2], TOR3[TOR3] on the next start bit detected. In other modes (Autoreload, Soft_trig), TOR1, TOR2 and TOR3 registers have the same value as TOR registers. Indeed, in START_TRIG mode, the next count can be configured whereas the running count is not finished.

36.3.5.2 mode

Most of the inputs and outputs of this block are used to test this block. The bits of the register TOC are shown by the following table.

Table 71. TOC register

TOC7	TOC6	TOC5	TOC4	TOC3	TOC2	TOC1	TOC0
EATR	MODE3	MODE2[1]	MODE2[0]	MODE1[1]	MODE1[0]	8/16	16/24

The three registers of TOR1, TOR2 and TOR3 form a programmable 24-bit ETU counter, or two independent counters (16 and 8-bit), or three independent 8-bit counters (depending on TOC0 and TOC1).

The value to load in TOR1, TOR2 and TOR3 is the number of ETUs to count.

The TOC register is used for setting different configurations of the time-out counter:

- If 16/24 and 8/16 bits are logic 0, then the counter is wired as a single 24-bit counter loaded with registers TOR3 (MSB byte), TOR2 and TOR1 (LSB byte). The end of count is signaled by USR2[TO3], USR2[TO1] and USR2[TO2] bits are logic zero.
- If 16/24 bit is logic 1 and 8/16 bit is logic 0, then the counter is wired as 2 independent counters. One is 16-bit loaded with registers TOR3 (MSB byte) and TOR2 (LSB byte), the end of count is signaled by USR2[TO3] and USR2[TO2] is logic zero. The other one is 8-bit loaded with register TOR1.
- If 8/16 bit is logic 1 whatever the value of 16/24 bit is, then there are 3 independent 8-bit counters, loaded respectively with registers TOR3, TOR2 and TOR1.

Table 72. Register CT_TOC_REG

Toc(1:0)	Counter3	Counter2	Counter1
00	tor3→	tor2→	tor1
01	tor3→	tor2	tor1
1X	tor3	tor2	tor1

Both counters 1 and 2 have four operation modes defined by bits TOC[MODE1] for counter 1, by bits TOC[MODE2] for counter 2; for starting the action in the different modes, the software must write the specific bits for all 3 counters in the configuration register. See the following table.

Table 73. Timers operating modes

(MODE1[1:0]	Operation mode
00	**Stop**: The counter stops counting, whatever the previous mode was.
01	**Autoreload**: The counter starts counting the value stored in the associated register on the first start bit after the mode has been programmed, and automatically restarts counting this value when it has reached the terminal count. An interrupt is generated at every terminal count. Changing the value of the associated TOR register during a count is not allowed.
10	**Software triggered:** The counter starts counting the value stored in the associated registers when this configuration has been written, and stops and generates an interrupt when it has reached its terminal count. The counter shall be stopped before reloading new value in the associated TOR register.
11	**Triggered on start bit on I/O:** In this mode, the counter will automatically start counting the value stored in the associated registers when a start bit occurs on I/O, and then on each subsequent start bit. It is possible to change the content of the associated TOR register during a count; the current count will not be affected and the new count value will be taken into account at the next start bit. An interrupt is only generated if the counter reached the terminal count.

Counter 3 has only two operation modes defined by bit MODE3[0]:

- When set to logic 0, counter 3 is stopped.
- When set to logic 1, counter 3 starts counting the value stored in TOR3 register (software triggered).

When counters 3 and 2 form a 16-bit counter, if counter 2 mode is triggered on start bit on I/O and counter 3 mode is software triggered, then the mode of this 16-bit counter is triggered on start bit on I/O. In the same way, when counters 3, 2 and 1 form a 24-bit counter, if counters 1 and 2 mode is triggered on start bit on I/O and counter 3 mode is software triggered, then the mode of this 24-bit counter is triggered on start bit on I/O.

When the bit TOC[EATR] is set to logic 1, the counters will be stopped automatically on the 12th ETU of the next character received after TOC[EATR] has been set to logic 1; if the counters had not reached their terminal counts at this moment, then no interrupt will be generated. This feature may be useful for some ATR configurations. This bit must be reset (set to logic 0) by software.

The design is composed of different blocks:

- Management of mode Autoreload: This part detects if some of the timers are configured in Autoreload mode.
- Management of mode Soft_Triggered: This part detects if some of the timers are configured in Soft_Triggered mode.
- Management of tor_counter: This defines 3 counters counting the value stored in tor1, tor2, tor3, depending on the configuration of timers.
- Management of interrupt: This part generates a pulse when the counter reaches the terminal count.

36.3.6 ISO USART

The block ISO USART manages reception and transmission of characters. This transceiver is done with a state machine. It also manages the control signals for registers. These signals induce some interrupts monitored by the micro-controller.

It consists of five parts:

- Clock circuitry
- FIFO
- Reception
- Transmission
- CSFR

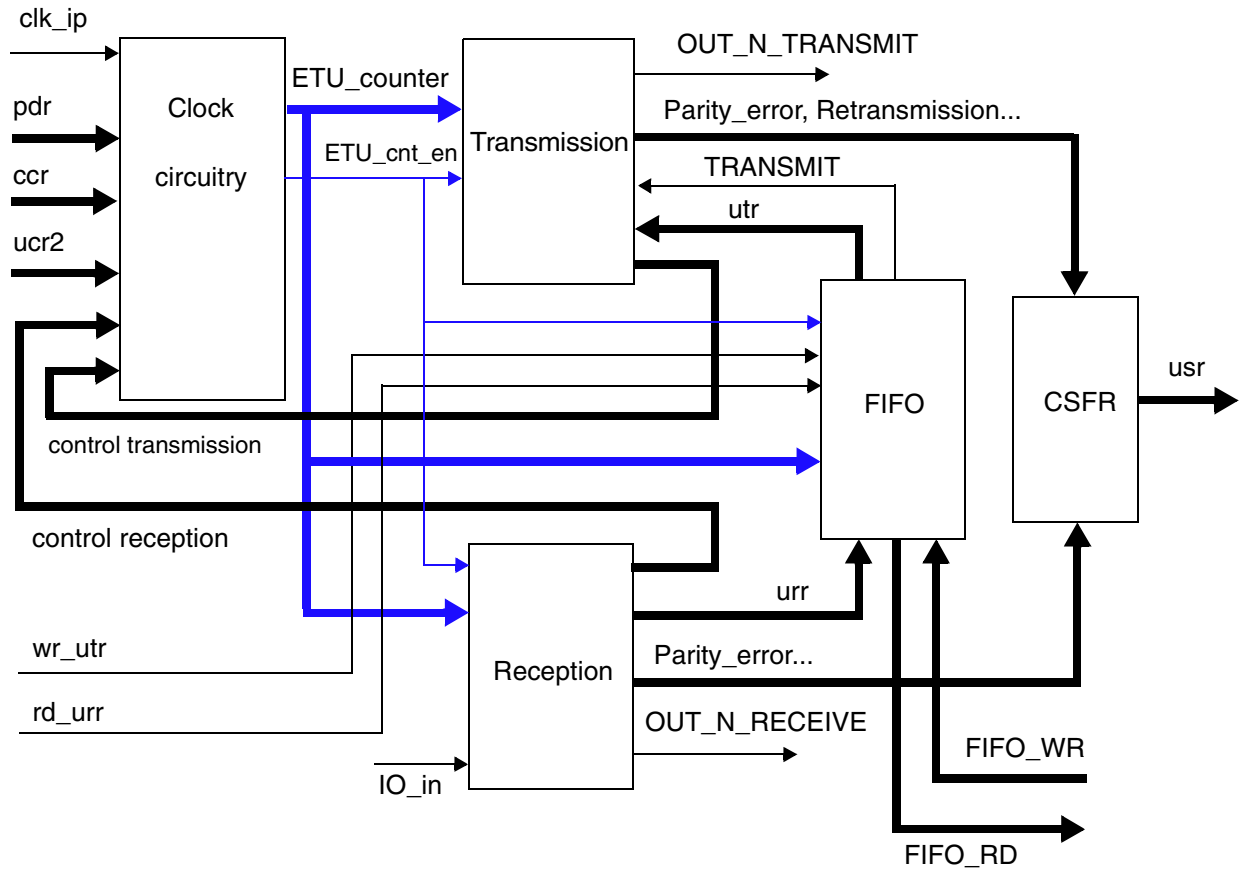


Fig 94. ISO USART block diagram

The block Clock circuitry provides a signal which enables to count (ETU_cnt_en). It also embeds a counter (ETU_counter) used for reception and transmission according to the frequency of the card clock.

The blocks Reception and Transmission respectively manage the reception and transmission mode with the card.

The FIFO receives the characters that the card emits via the Reception block and provides characters given by the micro-controller to the Transmission block.

The block CSFR (Control Signals For Registers) provides signals that will interrupt the micro-controller via registers.

36.3.6.1 FIFO

The FIFO is used for both Reception and Transmission modes. This block receives characters from the card via the Reception block and provides characters written by the micro-controller to the Transmission block. Its depth is 32 bytes.

- Reception:
  - In reception mode, when a received character is correct (no parity error) at 10.5 ETUs (T=0), it is loaded into the FIFO which size pointer is incremented.

- If the FIFO size pointer equals 32, no more character can be loaded into the FIFO. An Overrun interrupt will be generated by the ct_usr1_reg register to mean that at least one character will be lost.
- When the micro-controller reads a character, the FIFO size pointer is decremented.
- A read operation when the FIFO is empty will not cause any action for the FIFO (size pointer unchanged). In this case, 0 is read.

- Transmission:
  - The micro-controller can write a character into the FIFO while the USART is in transmission mode and if the FIFO is not full. If the FIFO contains already 32 characters, the write operation is not taken into account.
  - The FIFO commands the transmission. If its size pointer is one or more, the FIFO starts the transmission by loading the first character to transmit in the Transmission block. Then, the Transmission block will manage the transmission to the card. The FIFO size pointer is decremented at 9.5 ETUs.
  - If a parity error interrupt occurs after one or more retransmission(s) (T=0), there are two cases:

    PEC=0: no action, the transmission doesn't stop.

    PEC>0 (automatic mode): the transmission stops. The software will deactivate the card: the parity error counter will be reset by hardware when activating. If necessary, the firmware has the possibility to pursue the transmission. By reading the number of bytes present into the FIFO (FSR[FSR] bits), it can determine which character has been naked PEC +1 times by the card. It will then flush the FIFO (UCR21[FIFIOFLUSH]). The next step consists in unlocking the transmission using UCR21[DISPE] bit. By writing this bit at logic level one (and then at logic level zero if the firmware still wants to check parity errors), the transmission is unlocked. The firmware can now write bytes into the FIFO.
- Turnaround Reception -> Transmission:
  - There is a hardware protection when switching from reception to transmission mode. If the micro-controller sets to logic 1 the bit T/R for example between 10.5 (ft occurs) and 11.8 ETUs (reception finished), only the FIFO switches in transmission mode and not the rest of the USART which remains in reception mode. This allows the micro-controller to write characters into the FIFO. At 11.8 ETUs, the whole USART switches in transmission mode.
  - The FIFO is in transmission mode when the bit UCR11[T_R] is set to logic 1, else in reception mode. The transmission starts when the whole USART is in transmission mode, that is to say when the internal bit UCR11[T_R] is set to logic 1.

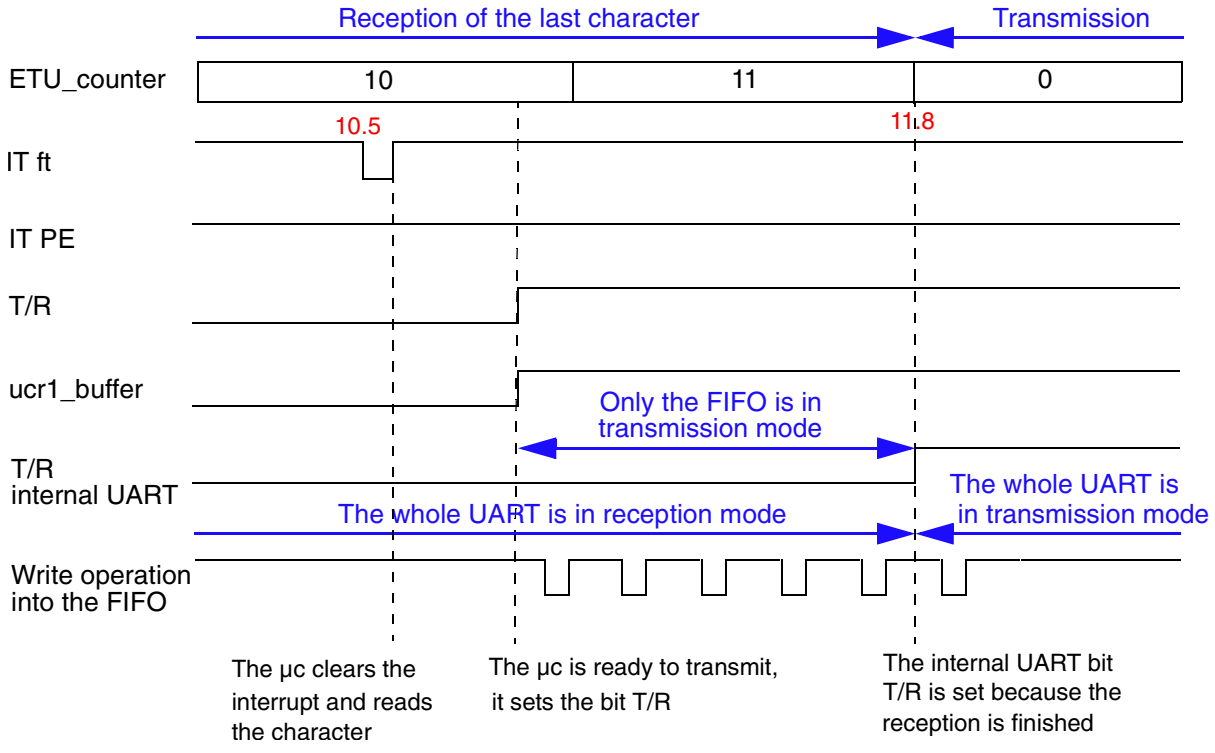


Fig 95. FIFO turnaround reception -> transmission

Remark: When switching from/to reception to/from transmission mode, the FIFO is flushed (and the size pointer is reset). Any remaining bytes are lost.

36.4 System integration

The Contact Interface supports the smart card communication. Software controls the Contact Interface via its APB slave interface. The Contact USART is connected to an external analog device capable to communicating with the Smart Card. This external analogue interface could be a TDA8020, TDA8023, TDA8026 or TDA8034 controlled via I^2C Master interface.

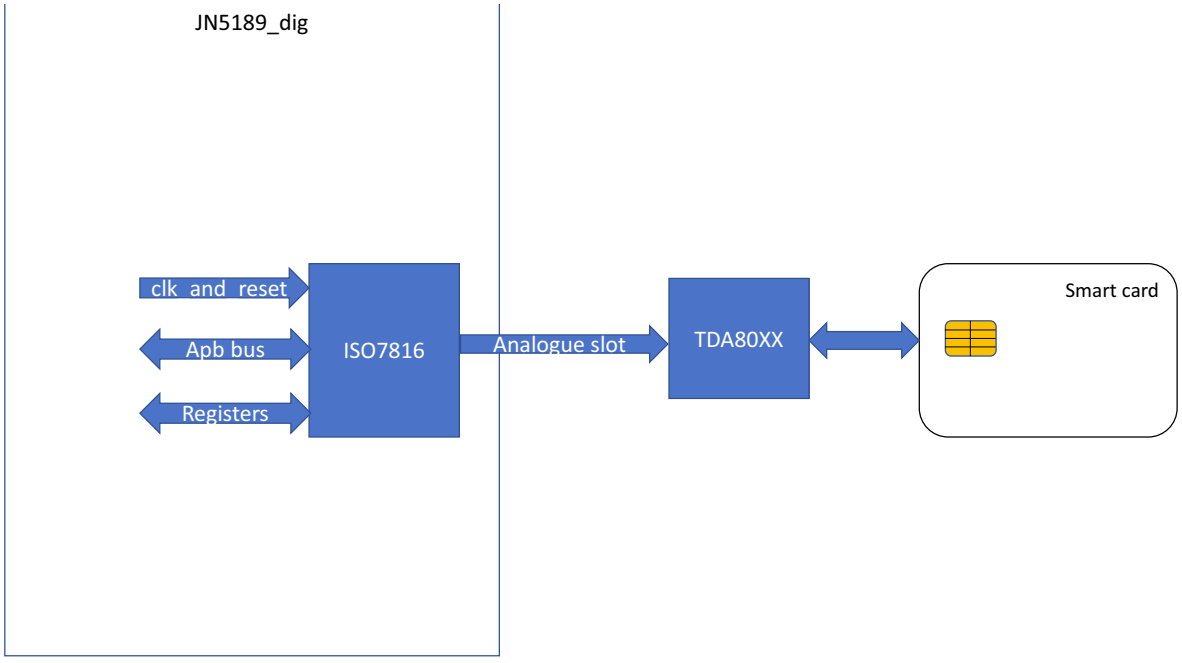


Fig 96. ISO7816 interface

36.4.1 Clock and Reset

The functional clock of the ISO7816 block (clk_ip) is connected to system_ahb_clk[21] thus it is generated from the MAINCLK as all the AHB clocks.

- The system_ahb_clock must be enabled by writing a '1' to SYSCON_AHBCLKCTRL0[ISO7816]. Use the CLOCK_AttachCLK() function to configure this.
- The ISO7816 block generates an output clock out of the peripheral clock, with a division factor that can be configured through the CCR1 register
- The iso7816 clock out can be directed to
  - PIO0_3 when IOCON_PIOx[FUNC] is 110b
  - PIO0_17 when IOCON_PIOx[FUNC] is 010b. See Table 19 for further details.

No other functional clock is connected to this IP, and therefore used to generate the output clock.

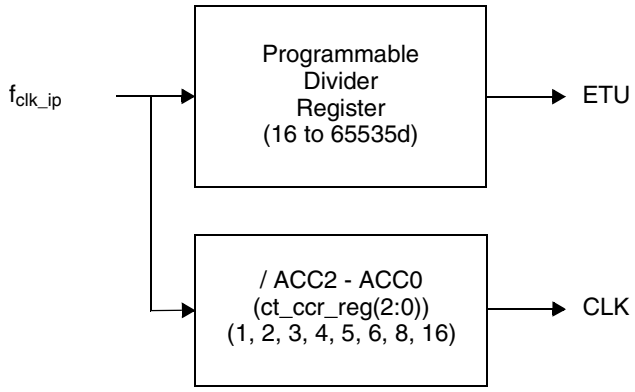


Fig 97. Clock generation

The reset of the ISO7816 block is connected to reset_peripheral_n_comb[21]. (see SYSCON_PRESETCTRL0 register).

The ISO7816 reset out can be directed to

- PIO0_2 in the fmux when mode is "110"
- PIO0_16 in the fmux when mode is "010"

36.4.2 GPIO configuration

2 GPIOs configurations are available on JN5189(T)/JN5188(T) to use ISO7816 interface:

- For IOCON_PIO[FUNC] = 010:
  - PIO0_16 = ISO7816_RST
  - PIO0_17 = ISO7816_CLK
  - PIO0_18 = ISO7816_IO
- For IOCON_PIO[FUNC] = 110:
  - PIO0_2 = ISO7816_RST
  - PIO0_3 = ISO7816_CLK
  - PIO0_4 = ISO7816_IO

To configure the IOs use the software API function IOCON_PinMuxSet with the required Func value for relevant PIO.

37.1 Introduction

The async system configuration module is located beyond a bridge, along with Timers 0/1 modules. It controls thus these timers, and some other analog-oriented modules: temperature control, external NFC tag, XTAL32, frequency measurement. It handles software reset as well. It is necessary to enable the bridge before accessing to ASYNC_SYSCON (SYSCON_ASYNCAPBCTRL[ENABLE]).

The ASYNC_SYSCON registers are reset by pin reset, watchdog reset, brownout reset, power-on reset, Arm System reset, SW reset and during power-down and deep power-down.

37.2 Counter/timers 0/1

Control of reset of timers, CTIMER0/1

- ASYNCPRESETCTRL to set or clear the resets.
- ASYNCPRESETCTRLSET just set the resets
- ASYNCPRESETCTRLCLR just clear resets

.

37.3 Clock control

Selection of the clock used for the async modules: ASYNC_SYSCON itself, timers CTIMER0/1: see definition of registers ASYNCAPBCLKSELA.

37.4 Temperature sensor control

The temperature sensor is connected to one of the ADC inputs; full information on using it is described in the Chapter 28 “Temperature Sensor”.

The enable for the temperature sensor cell is controlled by ASYNC_SYSCON. Also there are configuration which should be set to the predefined values. See register TEMPSENSORCTRL.

37.5 External NFC tag

For the JN5189T and JN5188T devices there is an NFC tag within the package. The I^2C2 block is used to communicate with the tag. The power for the tag is controlled from an IO cell. Before the tag can be used, the IO cells and tag power need to be configured correctly.

See registers NFCTAGPADSCTRL, NFCTAG_VDD.

It is recommended to use software APIs.

37.6 Frequency measurement

The frequency measurement block can be used to measure the frequency of one clock using another clock of known frequency.

The block has two clock inputs: reference clock and target clock. A scaler, SCALE, is programmed. When the block then starts it counts (2^{SCALE}-1) reference clock edges. During this time the target counter runs, incrementing every target clock edge.

Hence from the ratio of the count values, then the ratio of the frequencies is known.

See register FREQMECTRL for details. The higher the SCALE, the more accurate, but the slower.

The clock selection muxes are shown in Figure 98. Configure the clock selection by programming INPUTMUX_FREQMEAS_REF and INPUTMUX_FREQMEAS_TARGET, then start the measurement like this:

- Write the SCALE value, to the FREQMECTRL[CAPVAL] field
- set the FREQMECTRL[PROG] bit
- Poll the FREQMECTRL[PROG] bit until it is clear
- Read back FREQMECTRL[CAPVAL] , the number of target clock edges counted, from the FREQMECTRL[CAPVAL]

The result can be calculated from the equation:

Freq of Target clock = (Freq of Ref clock)* (CAPVAL + 1) / (2^{SCALE}-1)

Of course, both clocks (reference and target) must be enabled prior to this. If there is a large difference in frequency between the two clocks, configure the clocks so that the slowest clock is input as reference clock. This gives the highest accuracy. Software support for this operation can be found in fsl_meas.c

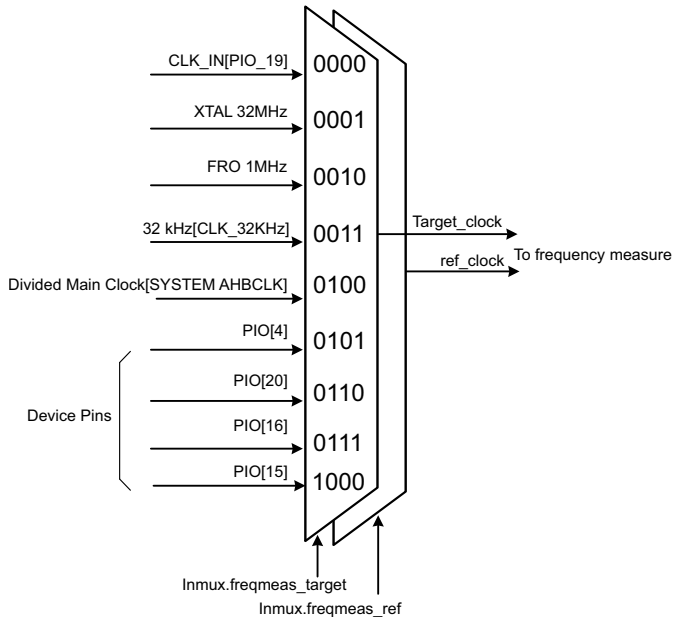


Fig 98. Clock sources to frequency measure block diagram

37.7 SW reset

It is possible for software to generate a reset from the SWRESETCTRL register and also a control bit in the PMC_CTRL register. See Section 7.2 “Reset”.

38.1 How to read this chapter

ISP functionality is supported by the boot code on all JN5189(T)/JN5188(T) devices.

38.2 Features

All JN5189(T)/JN5188(T) devices include ROM-based services for programming and reading the flash memory in addition to other functions. In-System Programming works on an unprogrammed or previously programmed device using a USART interface.

38.3 General description

At boot time, the device can be placed in a mode where a USART is enabled and software commands can be sent into the device. These are serviced by the boot code and allow many operations to be performed, such as reading or writing memory, erasing the flash.

There is also functionality to protect the contents of devices from being accessed in certain configurations.

38.4 Physical interface

The ISP protocol is implemented on USART 0, using DIO9 for RX and DIO8 for TX. The baud rate is 115200 with formatting of 8 bits per character, no parity bit and 1 stop bit.

In the default configuration, ISP mode is entered during a cold start if DIO5 is pulled low and DIO4 is pulled high. The normal approach to enter ISP mode is therefore:

1. Pull the reset pin low
2. Pull DIO5 low
3. Pull DIO4 [ISP_SEL] high. Note that If there is no external driver into this pin then the internal pull-up will keep this pin high.
4. After a short delay (1 ms), release the reset pin
5. Do not release DIO5 for at least another 10 ms

In practice, DIO5 can continue to be held low in ISP mode, and only be released before the device is reset again. Note that, in a system design with high capacitance on the circuit around DIO5, there must leave enough time after DIO5 is released for it to return to a high state before reset is asserted.

38.5 General message format

Standard packets have the following format:

0	1	2	3	4	…	n-4	n-3	n-2	n-1	n
Flags	Length (n+1)		Type	Payload			Checksum			

Fig 99. Standard packets format

Each field is described individually in the following sections.

Any value that spans more than one byte is usually sent big-endian: the most significant byte is transmitted first. Exceptions to this rule are noted in the descriptions.

38.5.1 Flags field

This is an 8-bit field. Bits are defined as follows:

Table 74: Flags field

Bit	Meaning
0	Undefined: leave as 0
1	If 1, packet has a SHA-256 signature (see section 3.4)
2	If 1, packet has a “next hash” (see section 3.4)
3-7	Undefined: leave as 0

38.5.2 Length field

The length is the total number of bytes. Using the terminology from the diagram above, the value will be n+1 (the Flags field was byte 0 and the final byte of the Checksum is n, so the total length is n+1).

38.5.3 Type field

The following commands are supported by the ISP:

Table 75: Type field

Command	Value	
	Request	Response
Reset	0x14	0x15
Execute (Run)	0x21	0x22
Set Baud Rate	0x27	0x28
Get Device Info	0x32	0x33
Open Memory For Access	0x40	0x41
Erase Memory	0x42	0x43
Blank Check Memory	0x44	0x45
Read Memory	0x46	0x47
Write Memory	0x48	0x49
Close Memory (prevent access)	0x4A	0x4B
Get Memory Info	0x4C	0x4D

Table 75: Type field

Command	Value	
	Request	Response
Unlock ISP	0x4E	0x4F
Use Certificate	0x50	0x51
Start Encrypted Transfer	0x52	0x53

In each case, the Response value is the same as the Request value but with bit 0 inverted. If a Request is not recognized, a response is send with bit 7 inverted instead.

38.5.4 Payload field

38.5.4.1 Request packets

The Payload of a Request packet has the following format:

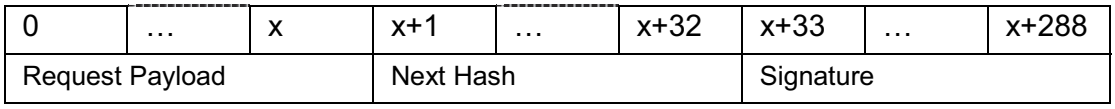


Fig 100. Payload request packet format

The Request Payload is dependent upon the value of the Type field; see Section 38.5.3 “Type field”.

The Next Hash and Signature fields are optional, with their presence controlled by bits in the Flags field.

The Next Hash is a SHA-256 hash calculated across all fields of the next packet, except the checksum, as part of an authenticated sequence.

The Signature is a SHA-256 hash calculated across all proceeding fields in the packet and encrypted with a 2048-bit RSA private key.

38.5.4.2 Response packets

The Payload of a Response packet has the following format:

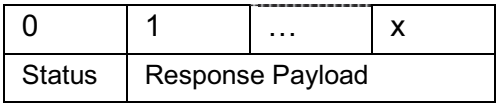


Fig 101. Payload of response packet format

The Response Payload is dependent upon the value of the Type field; see Section 38.5.3 “Type field”.

The following status codes are defined:

Table 76: Status code

Status	Descriptions
0x00	Success
0xEF	Memory invalid mode
0xF0	Memory bad state
0xF1	Memory too long
0xF2	Memory out of range
0xF3	Memory access invalid
0xF4	Memory not supported
0xF5	Memory invalid
0xF6	No response
0xF7	Not authorized
0xF8	Test error
0xF9	Read fail
0xFA	User interrupt
0xFB	Assert fail
0xFC	CRC error
0xFD	Invalid response
0xFE	Write fail
0xFF	Not supported

38.5.5 Checksum field

The checksum is a 32-bit CRC calculated across all proceeding fields. Functions to generate this are provided in the Production Flash Programmer source code in files:

- Library/Source/crc.c
- Library/Source/crc.h

Pseudo-code to generate the CRC for a packet using these functions is as follows:

```
crc_t crc = crc_init();

crc = crc_update(crc, packet_data, packet_length);

crc = crc_finalize(crc);
```

38.6 Commands and specific message formats

38.6.1 Standard commands

In each case, the Response message will always contain a status code, as listed in Section 38.5.4.2. Not all status codes are applicable to all message types.

38.6.1.1 Reset

This command resets the device. The response is sent before the device resets.

38.6.1.2 Execute

This command executes (runs) code in flash or RAM. The response is sent before execution jumps to the provided address.

The Request Payload contains the following fields:

Table 77: Execute field descriptions

Field	Offset	Size	Description
Address	0	4	Memory address to start execution from NOTE: Value is sent in little-endian

The Response Payload is empty.

38.6.1.3 Set baud rate

This command sets the ISP data rate. Each interface may support a different range of rates.

The Request Payload contains the following fields:

Table 78: Set baud rate fields descriptions

Field	Offset	Size	Description
Reserved	0	1	Reserved: value unimportant
Baud Rate	1	4	Baud rate, in bits per second NOTE: Value is sent in little-endian

The Response Payload is empty.

38.6.1.4 Get device information

This command returns device specific information and can be used to identify the connected device.

The Request Payload is empty.

The Response Payload contains the following fields:

Table 79: Get device info fields descriptions

Field	Offset	Size	Description
Chip ID	0	4	Chip identification number
Version	4	4	Chip version number

38.6.1.5 Open memory for access

This command selects and initializes a memory for programming.

The Request Payload contains the following fields:

Table 80: Open memory for access fields descriptions

Field	Offset	Size	Description
Memory ID	0	1	The ID of the memory block to be accessed. See table in section 4.1.11
Access Mode	1	1	Required access mode. See table in Section 38.6.1.11 "Get Memory Info"

The Response Payload contains the following fields:

Table 81: Get device info fields descriptions

Field	Offset	Size	Description
Handle	0	1	Handle to be used with subsequent access commands NOTE: Current implementation always returns 0, and only one memory type can be accessed at a time

38.6.1.6 Read memory

This command reads data from the selected memory.

The Request Payload contains the following fields:

Table 82: Read memory request fields descriptions

Field	Offset	Size	Description
Handle	0	1	Handle returned by open memory command (see Section 38.6.1.5)
Mode	1	1	Read mode: always use 0
Address	2	4	Address within memory to start reading from
Length	6	4	Number of bytes to read

The Response Payload contains the following fields:

Table 83: Read memory response fields descriptions

Field	Offset	Size	Description
Data	0	n	Data that was read from the memory

38.6.1.7 Write memory

This command writes data to the selected memory.

The Request Payload contains the following fields:

Table 84: Write memory request fields descriptions

Field	Offset	Size	Description
Handle	0	1	Handle returned by open memory command (see Section 38.6.1.5)
Mode	1	1	Write mode: always use 0
Address	2	4	Address within memory to start writing to NOTE: Value is sent in little-endian
Length	6	4	Number of bytes to write NOTE: Value is sent in little-endian
Data	8	n	Data to write

The response has no payload.

38.6.1.8 Erase memory

This command erases a region of the selected memory.

The Request Payload contains the following fields:

Table 85: Erase memory request fields descriptions

Field	Offset	Size	Description
Handle	0	1	Handle returned by open memory command (see Section 38.6.1.5)
Mode	1	1	Erase mode: always use 0
Address	2	4	Address within memory to start erasing from NOTE: Value is sent in little-endian
Length	6	4	Number of bytes to erase NOTE: Value is sent in little-endian

The response has no payload.

38.6.1.9 Blank Check Memory

This command checks if a region of the selected memory has been erased.

The Request Payload contains the following fields:

Table 86: Blank Check Memory request fields descriptions

Field	Offset	Size	Description
Handle	0	1	Handle returned by open memory command (see Section 38.6.1.5)
Mode	1	1	Blank check mode: always use 0
Address	2	4	Address within memory to start blank check from NOTE: Value is sent in little-endian
Length	6	4	Number of bytes to blank check NOTE: Value is sent in little-endian

The response has no payload. The Response status is Read Fail (0xF9) if the memory region is not blank.

38.6.1.10 Close Memory

This command de-selects and finalizes the programming of a memory. Any writes of buffered data should be completed before a response is sent. Once this command has been successfully completed, the device may be reset without loss of data written to non-volatile memory.

The Request Payload contains the following fields:

Table 87: Close Memory request fields descriptions

Field	Offset	Size	Description
Handle	0	1	Handle returned by open memory command (see Section 38.6.1.5)

The response has no payload.

38.6.1.11 Get Memory Info

This command returns information about the available memory blocks on a device. Each request results in information about 1 memory block. To retrieve information about all memory blocks, multiple requests should be made, each with an increasing value in the Memory ID field, until the response contains an error status to indicate that all memory blocks have been reported.

The Request Payload contains the following fields:

Table 88: Get Memory Info request fields descriptions

Field	Offset	Size	Description
Memory ID	0	1	First memory ID to search for (see list below)

The Response Payload contains the following fields:

Table 89: Get Memory Info response fields descriptions

Field	Offset	Size	Description
Memory ID	0	1	Memory ID found
Base Address	1	4	Base address of memory block. NOTE: Base addresses may be 'virtual' values rather than physical addresses
Length	5	4	Size of memory block, in bytes
Sector Size	9	4	Size of each sector within the memory block, and hence the minimum size for each read, write or erase operation
Type	13	1	Memory type (see table of basic memory types, below)
Access	14	1	Access rights, as a bitmap of multiple options (see table of access bit values, below)
Name	15	n	ASCII name of memory block

The following memory IDs are supported on the JN5189(T)/JN5188(T):

Table 90: JN5189(T)/JN5188(T) supported memory ID

Memory ID	Name
0	FLASH
1	PSECT
2	pFlash
3	Config
4	EFUSE
5	ROM
6	RAM0
7	RAM1

The following basic memory types are available:

Table 91: JN5189(T)/JN5188(T) available basic memory types

Type ID	Name
0	ROM

Table 91: JN5189(T)/JN5188(T) available basic memory types

Type ID	Name
1	FLASH
2	RAM
5	EFUSE (OTP)

The following access bit values are available:

Table 92: JN5189(T)/JN5188(T) available access bit values

Memory ID	Name
0	Read enabled
1	Write enabled
2	Erase enabled
3	Erase all enabled
4	Blank check enabled

38.6.1.12 Unlock ISP

This command initiates ISP functionality on the interface. Until this command has been issued, other commands will not work.

The Request Payload contains the following fields:

Table 93: Request payload fields descriptions

Field	Offset	Size	Description
Mode	0	1	ISP Mode (see table below)
Key	1	n	Key (see table below)

Table 94: Request payload mode descriptions

Mode	Descriptions
0x00	Default state: only Get Device Info command works in this mode
0x01	Start ISP functionality: all commands work in this mode if device is not locked Key (n = 16): 0x11223344556677881122334455667788
0x02-0x7E	Reserved
0x7F	Unlock device Key (n = 288): Signed Unlock Key
0x80-0xFF	Extended ISP unlock If an authenticated image is present, then the unlock command is passed to the ISP extension interface.

The response may contain the following status codes:

Table 95: Response possible status code

Status	Descriptions
Success	Success
Invalid Parameter	An unsupported mode or incorrect key was provided

For security, the OTP and protected sector are locked during ISP. To erase these areas, a mass erase of all other data will be performed and the unlock key written to the protected sector update page. During the next boot, the unlock key will be detected and the remaining sectors erased.

For all modes in the extension range (0x80 - 0xFF), the command is passed to the ISP Extension Handler. If the handler returns a success response, then subsequent commands will also be passed on via the extension interface.

38.6.1.13 Use certificate

This command is used to provide a public key certificate to the device. The certificate must be signed with the root certificate present in the device. If no root certificate is present, an error is generated.

The Request Payload contains the following fields:

Table 96: Request payload fields descriptions

Field	Offset	Size	Description
Certificate	0	296	Certificate

The response has no payload.

38.6.1.14 Start Encrypted Transfer

This command configures encrypted data transfer.

If this command completes successfully, the data payload of all subsequent memory read and write commands with the encrypted flag set will be decrypted / encrypted using the supplied settings.

The Request Payload contains the following fields:

Table 97: Request payload fields descriptions

Field	Offset	Size	Description
Mode	0	1	Encryption mode (see table below)
Base	1	4	Start address for encryption
End	5	4	End address for encryption
Key	9	24	Encrypted key + integrity code
Initialization Vector	33	4	Initialization vector

Table 98: Encryption Mode

Encryption Mode	Descriptions
0x00	None
0x01	AES CTR
0xFF-0x02	Reserved

The Base sets the encryption start address such that the offset into the encrypted data may be calculated from the memory access address.

The encryption Key is encrypted with the device firmware update key using AESKW. This allows for a firmware image to be encrypted with a unique key which is then securely provided to the device. Each device may then have a unique update key without requiring individual device images.

The response has no payload.

38.6.2 Authenticated commands

A sequence of one or more authenticated commands is started by a signed command. The authenticity of the signed command may be verified by calculating the SHA-256 hash of the command and comparing this to the decrypted signature.

Each command in the authenticated sequence must have the Authenticated bit set and contain a SHA-256 hash calculated across the Type, Length, Payload and Next Hash fields of the next packet in the sequence.

The sequence ends when a packet is received without the authenticated flag or if a command does not match the expected hash.

38.6.3 Extension interface

Reserved ISP unlock modes are passed to the extension interface. If the extension accepts the unlock command and returns a success status, then all following commands will be passed to the extension before further processing is performed by the ROM code. The extension may request the standard ROM processing by returning a continue status.

38.7 Essential message sequences

38.7.1 Initialize communications

It is important to use the Unlock ISP command before other commands are attempted. This sequence shows basic initialization and changing of the baud rate:

Unlock ISP to default state

```
TX 00 00 09 4E 00 A7 09 AE 19

RX 00 00 09 4F 00 BE 12 9F 58
```

Get Device Info (optional)

```
TX 00 00 08 32 21 4A 04 94

RX 00 00 11 33 00 88 88 88 88 00 00 00 00 AB 9A 33 AE
```

Unlock ISP to start ISP functionality

```
TX 00 00 19 4E 01 11 22 33 44 55 66 77 88 11 22 33 44 55 66 77 88 4B 17 03 DE

RX 00 00 09 4F 00 BE 12 9F 58
```

38.7.2 Read default MAC address from memory

Assumes communications have been initialized.

Open Memory ID 3 (Configuration) for Access

```
TX 00 00 0A 40 03 01 F2 CF AF 52

RX 00 00 0A 41 00 00 AF 27 A6 30
```

Read Memory (8 bytes from offset 0x9FC70)

```
TX 00 00 12 46 00 00 70 FC 09 00 08 00 00 00 E2 50 61 12

RX 00 00 11 47 00 8D 4C F4 01 00 8D 15 00 3A C4 D2 2E
```

Close Memory

```
TX 00 00 09 4A 00 C3 65 6B 1D

RX 00 00 09 4B 00 DA 7E 5A 5C
```

38.7.3 Erase Flash

Assumes communications have been initialized.

Open Memory ID 0 (Flash) for Access

```
TX 00 00 0A 40 00 0F 3E 5A D1 96

RX 00 00 0A 41 00 00 AF 27 A6 30
```

Erase Memory (0x9DE00 bytes from offset 0)

```
TX 00 00 12 42 00 00 00 00 00 00 00 DE 09 00 E9 69 09 F9

RX 00 00 09 43 00 12 A7 D0 54
```

Blank check (optional)

```
TX 00 00 12 44 00 00 00 00 00 00 00 DE 09 00 01 DC C3 BA

RX 00 00 09 45 00 44 FD 77 D2
```

Close Memory

```
TX 00 00 09 4A 00 C3 65 6B 1D

RX 00 00 09 4B 00 DA 7E 5A 5C
```

38.7.4 Program image to Flash

Assumes communications have been initialized and flash has been erased.

Open Memory ID 0 (Flash) for Access

```
TX 00 00 0A 40 00 0F 3E 5A D1 96

RX 00 00 0A 41 00 00 AF 27 A6 30
```

Program Memory (0x00000200 bytes from offset 0x00000000)

```
TX 00 02 12 48 00 00 00 00 00 00 00 02 00 00 E0 5F
```

```
   01 04 81 01 00 00 A3 01 00 00 A5 01 00 00 A7 01

   ....

   25 4B 98 47 81 1E 48 42 04 F0 7F 04 48 41 96 19

   A5 B7

RX 00 00 09 49 00 E8 48 38 DE
```

Program Memory (0x00000200 bytes from offset 0x00000200)

```
TX 00 02 12 48 00 00 00 02 00 00 00 02 00 00 40 2C

   05 D1 21 4B 2B 40 1A 1F 53 42 53 41 00 E0 00 23

   ....

   78 28 28 D1 00 21 00 E0 01 21 0C AC D8 F8 DE 8C

   F1 08

RX 00 00 09 49 00 E8 48 38 DE
```

(Additional Program Memory commands follow until complete image has been transferred)

Close Memory

```
TX 00 00 09 4A 00 C3 65 6B 1D

RX 00 00 09 4B 00 DA 7E 5A 5C
```

38.7.5 Read contents from Flash

Assumes communications have been initialized and flash has been erased.

Open Memory ID 0 (Flash) for Access

```
TX 00 00 0A 40 00 0F 3E 5A D1 96

RX 00 00 0A 41 00 00 AF 27 A6 30
```

Read Memory (0x200 bytes from offset 0)

```
TX 00 00 12 46 00 00 00 00 00 00 00 02 00 00 0C E1

   0D 66

RX 00 02 09 47 00 E0 5F 01 04 81 01 00 00 A3 01 00

   00 A5 01 00 00 A7 01 00 00 A9 01 00 00 AB 01 00

   ....

   4B 1C 6F D3 F8 B0 50 25 4B 98 47 81 1E 48 42 04

   F0 7F 04 48 41 CF 21 30 7F
```

Read Memory (0x200 bytes from offset 0x200)

```
TX 00 00 12 46 00 00 00 02 00 00 00 02 00 00 9B 7E
   1C 4F
RX 00 02 09 47 00 40 2C 05 D1 21 4B 2B 40 1A 1F 53
   42 53 41 00 E0 00 23 03 42 E6 D0 4F F0 80 43 D3
   ....
   44 88 46 58 28 03 D0 78 28 28 D1 00 21 00 E0 01
   21 0C AC D8 F8 9D D6 AA FD
```

(Additional Read Memory commands follow until complete image has been transferred)

Close Memory

```
TX 00 00 09 4A 00 C3 65 6B 1D
RX 00 00 09 4B 00 DA 7E 5A 5C
```

38.7.6 Set ISP access level to write-only

Remark: ISP access level of write-only means that all memory accesses to all memory regions are disabled except write to flash and erase all flash. Change is applied after next device reset

Assumes communications have been initialized.

Open Memory ID 2 (pFlash) for Access

```
TX 00 00 0A 40 02 0F 0C 6C B3 14
RX 00 00 0A 41 00 00 AF 27 A6 30
```

Program memory (4 bytes from offset 0x14, data 0x02020202)

```
TX 00 00 16 48 00 00 14 00 00 00 04 00 00 00 02 02 02 02 29 3E E8
   FF
RX 00 00 09 49 00 E8 48 38 DE
```

Close memory

```
TX 00 00 09 4A 00 C3 65 6B 1D
RX 00 00 09 4B 00 DA 7E 5A 5C
```

38.7.7 Reset device

Assumes communications have been initialized.

Reset

```
TX 00 00 08 14 F3 47 81 69
```

```
RX 00 00 09 15 00 FE 46 2A 86
```

38.8 Usage restrictions

It is possible to restrict access to the ISP functionality, for instance in order to protect code in the flash.

The ISP can be used at several levels of access:

- Unrestricted: normal developer mode
- Write-only: allows updates but existing contents cannot be read
- Locked: only allows limited access, with a secure handshake, for returns testing

It is also possible to disable ISP completely; in that case the boot loader never enters the ISP state. The access level is determined by the ISP access level field in the pFlash.

The mapping of access level to functionality is as follows:

Table 99: Access level functionality

Feature	Access level		
	Unrestricted	Write-only	Locked
Erase selected sectors (in application, pFlash, reserved)	x		
Erase all application flash at once	x	x	
Read flash (application, pFlash, reserved)	x		
Write application flash	x	x	
Write pFlash, reserved flash	x		
Return device ID	x	x	x
Secure handshake using device-specific key	x	x	x
Erase all flash at once, only after secure handshake	x	x	x
Unlock SWD, only after secure handshake	x	x	x

Note that erasing all flash will return the N-2 page, pFlash and page0 to the initial state, where all settings are unlocked. The user should not perform this operation because vital settings and trim values will be lost.

The secure handshake procedure relies upon public-key cryptography, where a public key is stored on the device. A private key, known to an authorized user, is used to send a message to the device and the device can then use the public key to determine if the message used the correct private key.

The public key is stored in pFlash, AES encrypted using the device's 128-bit random key in eFuse. The encryption ensures that a malicious user with write access would be unable to replace the key with a usable alternative.

39.1 Introduction

The AES provides an on-chip hardware AES encryption and decryption engine to protect the image content and to accelerate processing for data encryption or decryption, data integrity, and proof of origin. Data can be encrypted or decrypted by the AES engine using the secret encrypted key in the OTP or a software supplied key.

39.2 Features

- DMA support
- AHB Slave Interface
- 32 bit counter increment
- Integrated GF128 hash engine for Galois Counter Mode (GCM)
- Supported modes: ECB, CBC, CFB, OFB, CTR, GCM

39.3 Function description

The AES engine uses 128-bit, 192-bit or 256-bit key and processes blocks of 128 bits. Encrypting and decrypting data, storing and retrieving the key.

An AES encryption/decryption engine. Encryption/decryption is selected through the CFG register.

The AES encryption and decryption engine optionally supports DMA for transferring data in and out the AES engine.

The AES engine can be loaded with two different keys:

- Secret key stored in the efuse OTP, can be used to encrypt or decrypt data. This key is the most secure key, and cannot be read by software. It is used by the stack software.
- A software defined key, stored in the KEY0-KEY7 write-only registers. As it is stored in registers, the software defined key must be reloaded after reset, and will not be retained in deep power-down mode.

The AES dma request signals (for intext and exttext transfer) are connected to the DMA Trigger inputs number 10 and 11 and they can be connected to the trigger inputs of any DMA channel.

The DMA-based transfer is optional, but when enabled it allows to perform an AES encryption without using the CPU.

Next picture shows the AES data encryption/decryption principle.

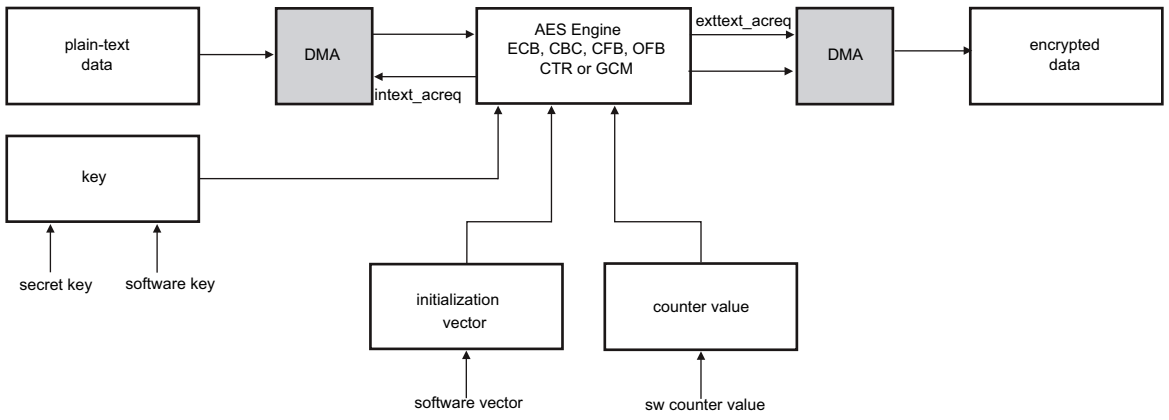


Fig 102. AES encryption/decryption

39.4 Software interface

The AES registers are accessible through an AHB slave port. This section gives a brief description of the AES registers, please refer to section of the JN5189(T)/JN5188(T) Registers document for more details.

39.4.1 AES Configuration register

The Configuration register allows to select the features of the AES like:

- 128/192/256 bits key
- Encryption/Decryption only, GF128 Hash only, Encrypt/Decrypt and Hash
- Input and Output blocks selection
- Input/ Output Text Word Swap

39.4.2 AES Command register

The command register is used to send commands to the AES core, like:

- Copy the secret key and enable cypher
- Switch from Forward to Reverse mode
- Abort
- Wipe (aborts encryption/decryption, clears the key, disables cypher, clears GF128_Y)

39.4.3 AES Status register

The Status Register contains information about when input text can be written (input buffer empty) and when output text can be read (encryption/decryption). Those bits are polled by the CPU when the DMA is not used.

39.4.4 AES Other Registers

The remaining registers provide support for the software key, input and output text. Please refer to the JN5189(T)/JN5188(T) Registers files for more details.

39.5 Example of Operations

39.5.1 Encryption without DMA

This paragraph describes an example of using the AES block for encryption without DMA. The correct sequence or operations is listed here below:

- AES Initialization (reset, enable clock)
- Write to the CFG register, to select the features to be enabled
- Write to KEY0-KEY7 registers to program the software defined key
- Write to the CMD register
- Read the STAT Register (to check if the INTEXT buffer is ready to be written)
- Write the plain text data to be encrypted into INTEXT0-INTEXT3 registers
- Poll the STAT Register (end of encryption is flagged in the OUT_READY bitfield)

- Read the encrypted text from OUTTEXT0-OUTTEXT3 registers

39.5.2 Encryption with DMA

This section describes an example of using the AES block with DMA; this mode allows to execute data encryption without using the CPU for better efficiency.

- Configure the trigger mux (to connect the triggers from AES to a DMA channel)
- DMA Initialization (reset, clock enable, enable DMA master by setting the DMA_CTRL[ENABLE] bit)
- AES Initialization (take the IP out of reset, enable clock)
- Write to the CFG register, to select the features to be enabled
- Define DMA Descriptor for the aes_rx transfer (source = OUTTEXTx register, destination = memory buffer)
- Configure DMA Channel to use hardware AES_RX trigger for rx
- Enable DMA Rx Channel
- Define DMA Descriptor for the aes_tx transfer (source = memory buffer, destination = INTEXTx register)
- Configure DMA Channel to use hardware AES_TX trigger for tx
- Enable DMA Tx Channel
- Trigger the first Tacreqx transfer by software (setting to 1 the related bit of the DMA_SETTRIG0 register), because the intext_acreq signal (indicating that the input buffer is ready to be filled) is by default =1, and generates an edge only after the first time that the buffer is filled and emptied.

39.5.3 Key management

To use the known key, it must be loaded in the AES registers KEY0 to KEY7.

To use the secret key, it is necessary to set the bit CMD[COPY_SKEY]. Then the secret key will be used by the AES engine for performing the encryption.

39.6 Software control

To use the functionality of the AES module, it is recommended to use software functions from within fsl_aes.c.

40.1 How to read this chapter

One Infra-Red Modulator (IRB) is available on all JN5189(T)/JN5188(T) devices.

40.2 Features

- Transmission of data over an InfraRed link to another system or device
- IR blaster support RC5, RC6, RCMM and SIRCS protocols and all existing protocols for common TV or STB appliances
- IR Blaster does NOT support the IRDA protocols

40.3 Basic configuration

Initial configuration of IR blaster peripheral is accomplished as follows:

- If needed, use the SYSCON_PRESETCTRL1 register to reset the IR interface.
- Configure the FIFOs for operation
- Configure IR blaster:
  - In the SYSCON_AHBCLKCTRL1 register, set the appropriate bit to enable the clock to the register interface.
  - Enable or disable the related IR Blaster Interface interrupt in the NVIC (see Table 17 “Connection of interrupt sources to the NVIC”).
  - Configure the related IR Blaster Interface pin functions via IOCON, see Chapter 12 “I/O Pin Configuration (IOCON)”.

40.4 Pin description

PIO0_12, PIO0_20, PIO0_21 can be used for IR function. Recommended IOCON settings are shown in Table 100. See Chapter 12 “I/O Pin Configuration (IOCON)” for definitions of pin types.

Table 100. Suggested IR Blaster pin setting for standard GPIO IO

IOCON bit(s)	Field	Setting	Note
2:0	Func	Set for IR_BLASTER function	
4:3	Mode	2	No pullup or pull-down
5	Slew0	0	See slew1
6	Invert	0	No need to invert
7	Digimode	1	Digital mode
8	FilterOff	1	Generally disable filter

Table 100. Suggested IR Blaster pin setting for standard GPIO IO

IOCON bit(s)	Field	Setting	Note
9	Slew1	0	With slew0: generally set to 0. Settings to 1,2 or 3 at high usary rates may improve performance
10	OD	0	Normal GPIO mode
11	IO_clamp	0	Clamp Off

See Figure 2 “Main memory map” for Main memory map details.

40.5 General Description

The InfraRed Blaster provides timers and counters that generate specific pulse waveforms that are applied to an InfraRed diode. This allows transmission of data over an InfraRed link to another system or device. InfraRed (IR) technology employs the use of a carrier waveform that is composed of short pulses, usually with a duty cycle of less than 50%. This reduces the "ON" time of the InfraRed diode and, thus, lowers the overall system battery drain for hand-held remote controllers, joysticks, etc.

The IR Blaster is used within the Set-Top Box (STB), such that under software control the STB will use wide angle IR diode to communicate with other appliances and control them. This offers the advantage that the end user may then control all of his appliances with a single remote-controller that comes with the STB.

The IR Blaster Module provides a variety of programming options that support all existing protocols for common TV or STB appliances. Note, however, the IR Blaster does NOT support the IRDA protocols.

40.6 Functional description

The InfraRed Blaster (IR Blaster) generates specific waveforms that are applied to an IR diode. It supports RC5, RC6, RCMM and SIRCS protocols. Generated waveforms are the result of an amplitude modulation of two signals:

- the carrier: this is a square signal with programmable frequency and duty cycle. It doesn't carry any information. It reduces power consumption by reducing "ON" time of the InfraRed diode. This signal is filtered out by receiver.
- the envelope: this signal carries information. It consists of a succession of high and low level of different duration.

The IR Blaster produces repetitive waveform with predefined frequency and duty cycle with little involvement from software. A commonly used IR signal can be described as an AM (amplitude modulated) signal, where an envelope timing is defined as a multiple of a base time unit, which is generally different for every brand of video and audio equipments.

A carrier frequency is defined by settings in the IR Blaster carrier configuration register. Two fields are used to determine the relative "high" and the "low" time of the carrier. All timings are provided in Carrier Timebase Units (CTU), as determined by the CARRIER[CTU] register field. Because all timings are in CTUs, there is a guarantee of no glitches or unwanted pulses or the disappearance of any parts of the output signal.

The block diagram below shows the functional units of the IR Blaster module.

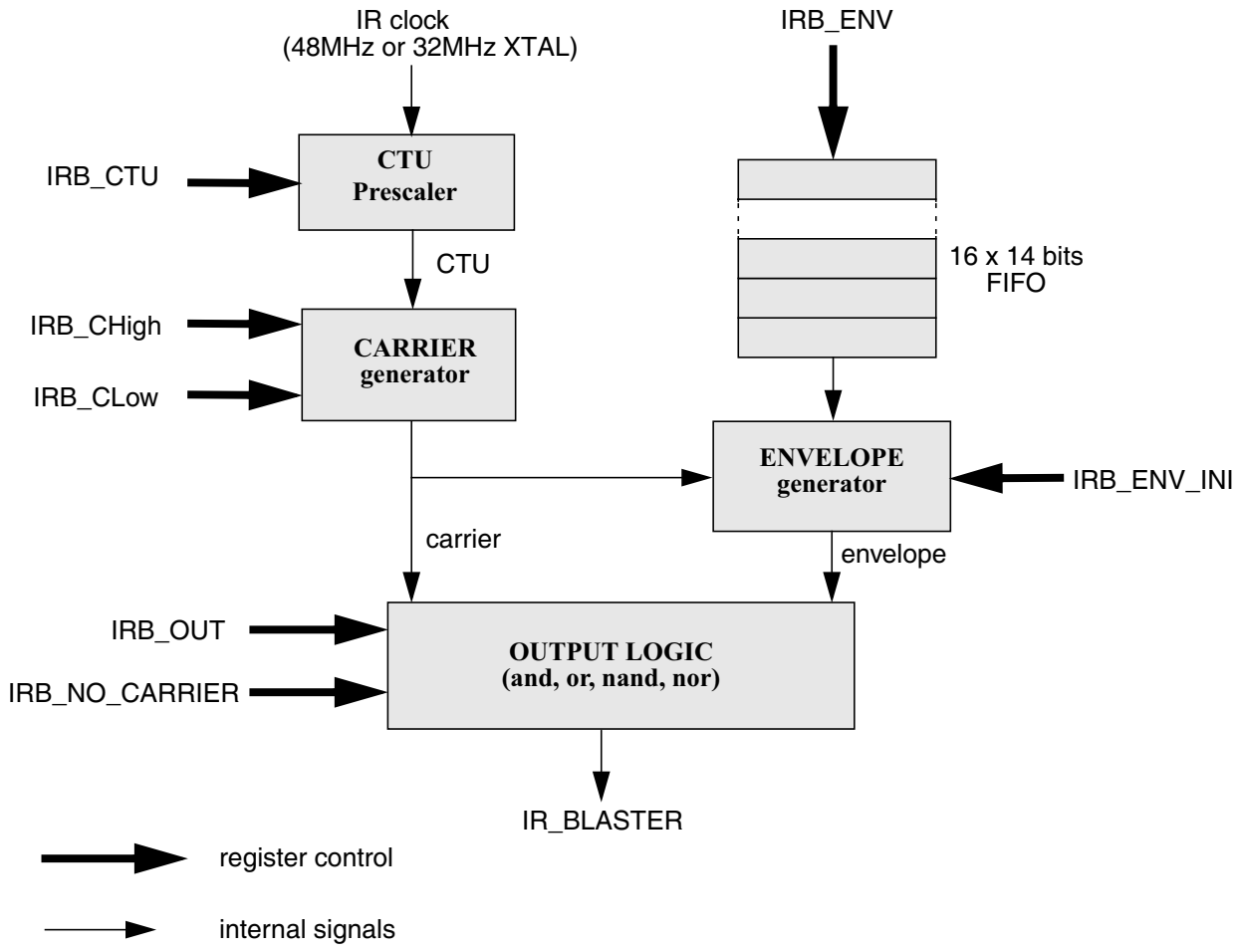


Fig 103. IR blaster block diagram

IR Blaster can be used in 2 modes according to the CONF[MODE]:

- Normal mode:

  In this mode, if IR Blaster encounters a data, FIFO_IN[ENV], with FIFO_IN[ENV_LAST] = 1, then it stops right after transmitting the corresponding envelope even if the FIFO is not empty. The software can restart IR Blaster to make it transmit the following data present in FIFO, by writing 1 to the CMD[START] field.

  Before restarting, it is possible to add data in FIFO. When IR Blaster is stopped, envelope signal is always 0.

  If there is not data in FIFO with FIFO_IN[ENV_LAST] = 1, IR Blaster transmit all data present in FIFO and stop when the FIFO becomes empty (this event generates FIFO_UFL_INT interrupt).

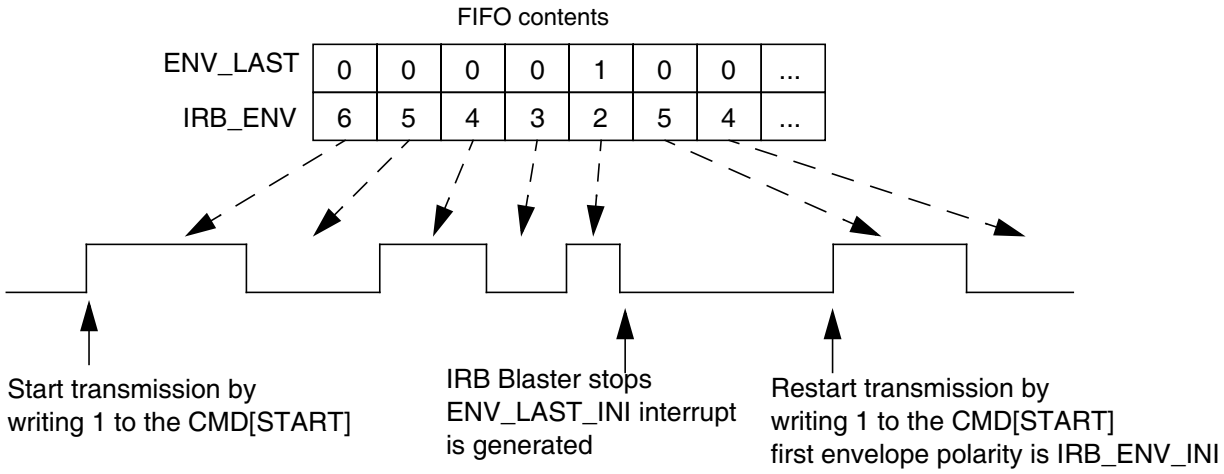


Fig 104. Normal mode

- Automatic restart

  In this mode, IR Blaster transmit all the data present in FIFO regardless of the FIFO_IN[ENV_LAST] field. It will stop only when FIFO becomes empty (this event generates FIFO_UFL_INT interrupt).

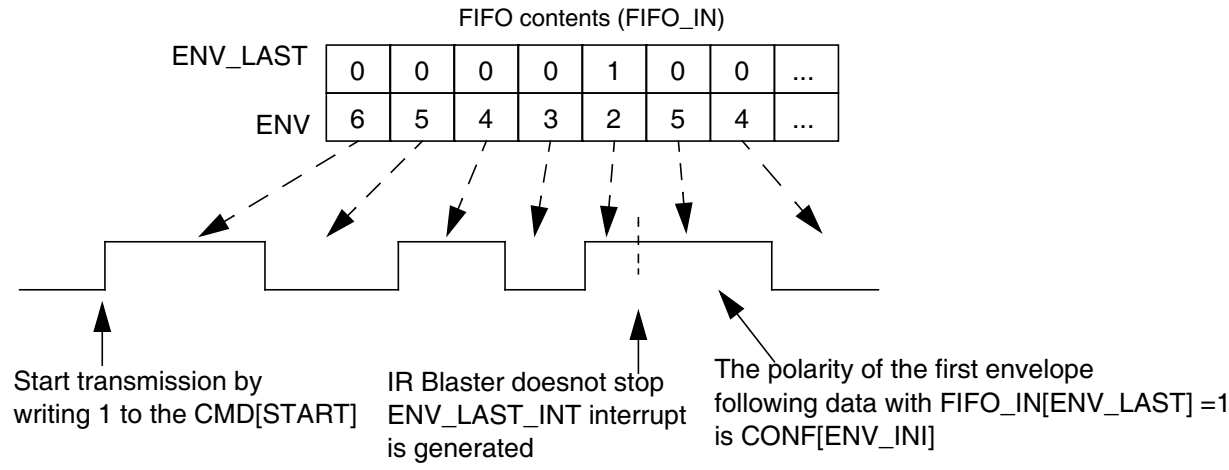


Fig 105. Automatic mode

Remark: In both modes, it is possible to have consecutive envelopes with the same polarity:

- Normal mode: This situation occurs if first or last envelope (envelope with FIFO_IN[ENV_LAST] = 1) is for a low level.
- Automatic restart: This situation occurs if last envelope polarity = IRB_ENV_INI.

40.7 Programming examples

Some programming example for RC5, RC6, RCMM and SIRCS are provided in this section of the document. In addition there is an example for RC5 within the SDK. The configuration assumes that the clock for the Infra-Red Modulator is 48 MHz.

40.7.1 RC5

The discussion below is based on the following data table giving the tolerances for RC5 protocol from a decoder point of view

Remark: 1T = 888,88 μs

Table 101. RC5 timings

Carrier definition	Tmin (μs)	Tmax (μs)
Period	27.233	28.345
"ON" time with 33% duty cycle	9.078	9.448
Pulse width during "AGC time"		
Receiver "1T" RC5 start pulse	675.556	1306.667
Receiver "2T" RC5 start pulse	1351.111	2613.333
Receiver "1T" RC5 first low pulse	506.667	1120.000
Pulse width after "AGC time"		
Receiver "1T"	675.556	1120.00
Receiver "2T"	1520.000	2053.333

Enable Blaster unit:

- Set CMD[ENA] field

Enable Blaster interrupts:

- Set INT_ENA[ENV_START_ENA] and INT_ENA[ENV_LAST_ENA] bits

Configure Blaster:

- CONF[ENV_INI] = 0b
- In RC5, the first data bit is always '1', so the first envelope is a low level (first half of the biphase encoded bit).
- CONF[MODE] = 0b
- The Blaster unit should stop after completion of transmission
- CONF[OUT] = 00b
- Uses "AND" output logic function if the LED does not invert the signal (LED is on when signal is at high level) otherwise uses "NAND" function.
- CONF[NO_CAR] = 0b
- Normal mode for carrier + envelope.
- CONF[CAR_INI] = 1b
- Keep default definition of carrier.

Configure Carrier:

- CARRIER[CTU] = 444 = 0x1BC
- CARRIER[CLOW] = 01b
- CARRIER[CHIGH] = 00b
- With this configuration, carrier period is (CARRIER[CTU] * (CARRIER[CLOW] + CARRIER[CHIGH] + 2)) / 48MHz = 27.75 μs, and "ON" time is CARRIER[CTU] * (CARRIER[CHIGH] + 1) / 48MHz = 9.25 μs.

Fill FIFO:

- Assuming a system latency of 1 ms, and considering that the shortest period between 2 'normal' edges is 888 μs, a room of 2 envelopes in the FIFO should be enough. To avoid any problems, we can choose a room of 4 envelopes.
- Fill the FIFO with encoded values (0x20 for 1T and 0x40 for 2T).
- For the 12th data, set the FIFO_IN[ENV_INT] bit.

Start transmission:

- Set CMD[START] field.

When ENV_START_INT interrupt occurs:

- Fill the FIFO with remaining envelopes.
- Set the FIFO_IN[ENV_INT] bit for the data placed in FIFO when FIFO level reaches 12 and set FIFO_IN[ENV_LAST] bit for the last data of the RC5 frame (Signal Free Time).

When ENV_LAST_INT interrupt occurs:

- The Blaster unit is ready for next RC5 frame. Just refill the FIFO with data of new RC5 frame and re-start the blaster unit.

40.7.2 RC6

The discussion below is based on the following data table giving the tolerances for RC6 protocol from a decoder point of view.

Remark: 1T = 444,44 μs

Table 102. RC6 timings

Carrier definition	Tmin (μs)	Tmax (μs)
Period	27.233	28.345
"ON" time with 33% duty cycle	9.078	9.448
Pulse width during "AGC time"		
Receiver "6T" start pulse	2178.667	3360.000
Pulse width after "AGC time"		
Receiver “1T”	249.111	658.000
Receiver “2T”	671.333	1124.667
Receiver “3T”	1125.6667	1573.111

The blaster unit programming is similar to that for the RC5 protocol. The differences are the envelope values (1T = 0x10, 2T = 0x20...) and CONF[ENV_INI] = 1b.

40.7.3 RCMM

The discussion below is based on the following data table giving the tolerances for RCMM protocol from a decoder point of view.

Remark: 1T = 27,778 μs

Table 103. RCMM timings

Carrier definition	Tmin (μs)	Tmax (μs)
Period	27.233	28.345
“ON” time with 33% duty cycle	9.078	9.448
Pulse width during “AGC time”		
Receiver “15T” RCMM start pulse	330	583
Receiver “10T” RCMM first low pulse	169	306
Receiver “25T” RCMM header ON + OFF	639	752
Pulse width after “AGC time”		
Receiver “6T” RCMM	80	275
RCMM DATATIME 00	386	502
RCMM DATATIME 01	553	669
RCMM DATATIME 10	720	836
RCMM DATATIME 11	886	1002

The blaster unit programming is similar to that for the RC5 protocol. The differences are the envelope values (15T = 0xF, 10T = 0xA...) and CONF[ENV_INI] = 1b.

40.8 Software control

To use the functionality of the IRB module, it is recommended to use software functions from within fsl_cic_irb.c. In the SDK the Infra Red Blaster module is referenced as Infra Red Modulator (IRM) block.

41.1 Introduction

The Radio has the following features:

- Supported Protocols
  - 2.4 GHz IEEE 802.15.4 2011 compliant
- High receiver performance
  - IEEE 802.15.4 Receiver sensitivity -100 dBm
  - Receive current 4.3 mA
  - 50 Ω single ended input (no balun required)
  - Improved co-existence with WiFi
  - Antenna Diversity control
- High transmitter performance
  - Flexible output power up to +11 dBm, with 46 dB range
  - Low Energy Consumption
  - Transmit power +10 dBm current 20.28 mA
  - Transmit power +3 dBm current 9.44 mA
  - Transmit power 0 dBm current 7.36 mA
  - Integrated ultra Low-power sleep oscillator
- Deep Power-down current 300 nA (with wake-up from IO)
- Wide operating range
  - 1.9 V to 3.6 V supply voltage
- Reference clock
  - 32 MHz XTAL cell with internal trimming capacitors, able with suitable external XTAL, to meet the required accuracy for radio operation over the operating conditions
  - 128-bit or 256-bit AES security processor (within digital infrastructure)
  - MAC accelerator with packet formatting, CRCs, address check, auto-acks, timers

The 2.4GHz Wireless Radio consists of the following main blocks:

- Analog Radio Transceiver featuring the analog transmitter and receiver sub-system as well as the LO frequency synthesizer functionality. The 32 MHz system clock, used within the analogue and digital is generated in this block.
- Common Interface Blocks, this includes receive signal path processing such as DC offset cancellation, Automatic Gain Control (AGC), down-mixing/sampling and transmit signal path functions such as generating modulation signals.
- Radio Controller, this sequences the digital control interface to the analogue transceiver (radio warm-up/warm-down, accurate time control of enabling and disabling of the receiver and transmitter circuitry), controls calibration functions (of

receiver, transmitter and frequency synthesizer) which may have digital and analog functionality and also provides some overall control of the radio (power Amplifier output power ramping-up/down) and data path interaction,

- Zigbee/Thread Modem, this is responsible for Zigbee/Thread symbol recognition functions in receive mode such as detecting the start of a packet and determining the receive symbol data for the packets. In transmit mode it translates symbols into a modulating bit stream,
- Zigbee/Thread MAC/ Baseband Controller this is responsible for managing Zigbee/Thread packet level operations such as CRC checking, header filtering, auto acknowledgments, packet retries and CCA management. The packet data is also fetched or written into the system SRAM.

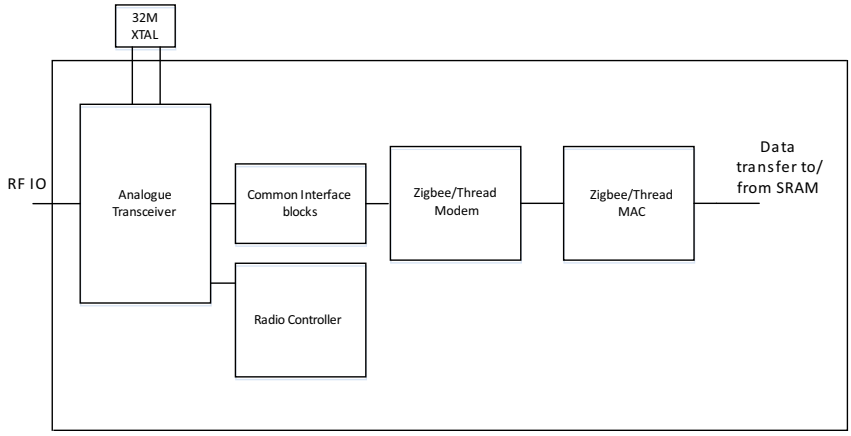


Fig 106. Radio system block diagram

41.2 Radio analog transceiver

The JN5189(T)/JN5188(T) features a highly configurable radio transceiver which supports Zigbee/Thread.

41.2.1 Radio architecture

The main features of the JN5189(T)/JN5188(T) analog radio are:

- Single ended shared RF pin for receive and transmit operations
- Each power domain has its own independent LDO (supplied by the top VDD_radio pin), as shown by the different colors.
- 1 low noise PLL serving either the receiver or the transmitter. In TX, the modulation is applied both at the feedback divider and on the VCO through the transmit DAC (2-point modulation PLL)
- An auxiliary PLL (noted Pilot Tone) is available for PA calibration and RX testing

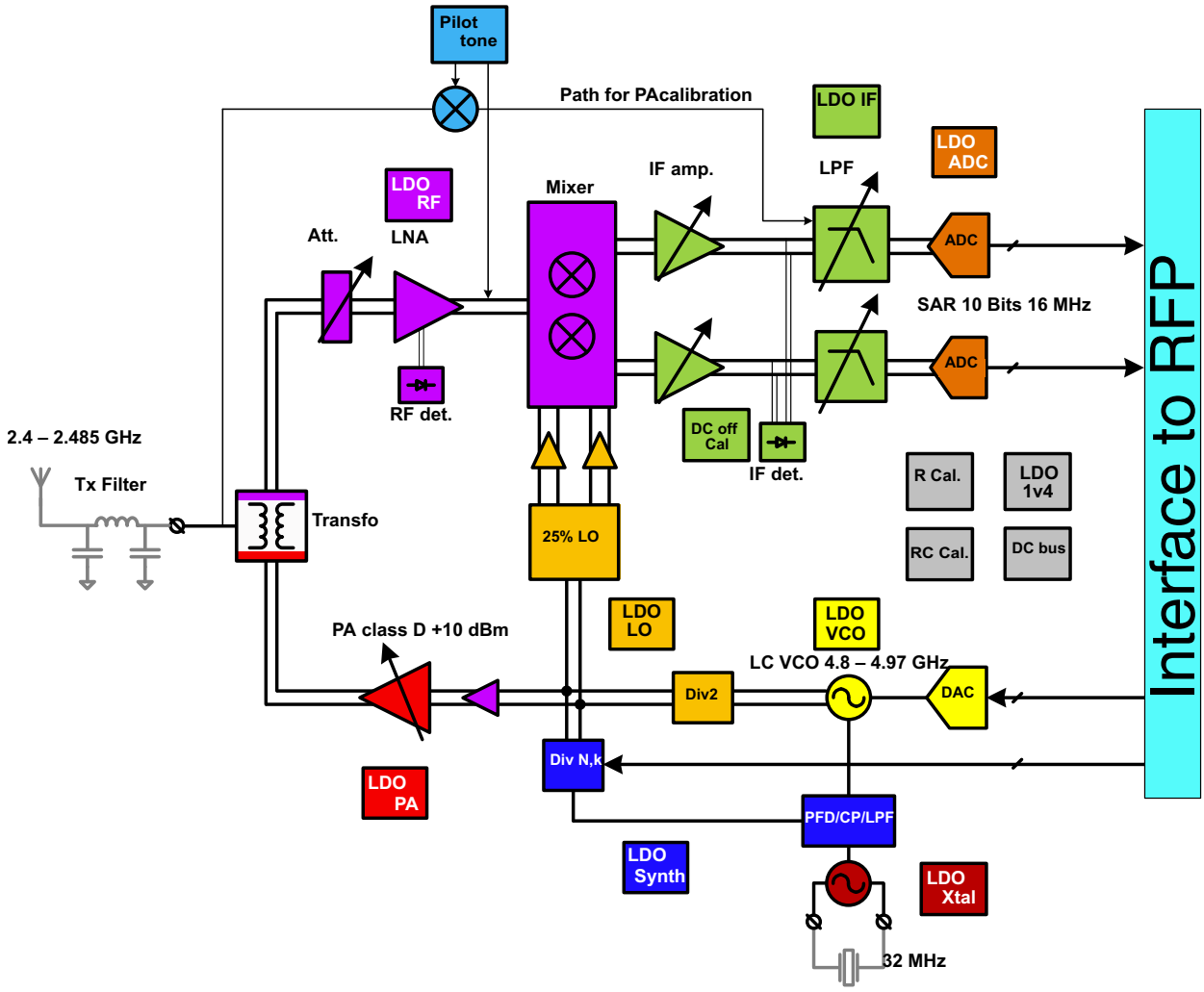


Fig 107. Radio block diagram

41.2.2 Antenna interface

The RF port is single ended and requires the following external matching network (see Figure 108) on the PCB. It includes harmonic filtering for the transmit function. Careful layout of this and power supplies is necessary to achieve the optimum performance.

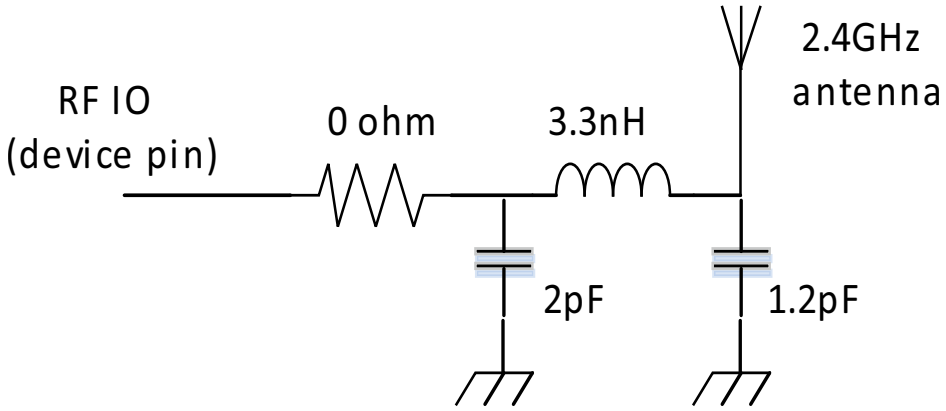


Fig 108. RF matching circuit

41.2.3 Fractional-N frequency synthesizer

A low phase noise, fully integrated fractional-N frequency synthesizer is used in receive mode to provide the LO frequency used by the down-conversion mixer. It is also used in transmit mode to generate the modulated RF carrier.

The synthesizer has a fast frequency settling time which allows very short receiver and transmitter wake-up time for optimized system power consumption.

Synthesizer's VCO calibrated is calibrated at power-up.

A 32 MHz frequency crystal oscillator provides the reference signal to the synthesizer input.

41.2.4 Receiver architecture

Receiver is based on a low-IF receiver architecture, consisting of a Low-Noise Amplifier (LNA) followed by an I/Q down-conversion mixer. Resulting I/Q signals are then amplified and channel filtered before being sampled by analog-to-digital converter (ADC) at 16 Mbps sampling rate. The samples from ADC feeds the digital RX datapath, described in next sections.

The IF frequency is adjusted versus protocol operation (1.3125 MHz in Zigbee/Thread). The IF can be configured for high-side or low-side injection operation, providing flexibility for improved robustness to identified interferer at the image frequency.

Automatic Gain Control (AGC) adjusts the receiver gain to avoid signal chain saturation while selecting the best noise linearity trade-off to guarantee optimum range operation / blocking performance.

The receiver is calibrated at production to improve image rejection performance.

Demodulation is performed in the digital domain.

A Received Signal Strength Indicator (RSSI) is available for signal level metrics. A RSSI value is associated with each received frame and the dynamic RSSI measurement can be monitored throughout reception.

RX antenna diversity is supportted in order to mitigate frequency-selective fading effect due to multi-path propagation and improve link budget. Antenna diversity is supported for specific PHY configurations in 2.4 GHz.

41.2.5 Transmitter architecture

Transmitter is based on two-point modulated fractional-N frequency synthesizer architecture.

TX digital data path modulator controls phase and frequency modulation in the frequency synthesizer. Transmit symbols or chips are pulse shaped by a digital shaping filter (Gaussian or Half Sine shaping).

A proprietary Zigbee/Thread mode is introduced (side lobes filtering by reinforcing the regular Half-sine pulse shaping filtering with a Gaussian based filtering (BT=0.95 or 0.5)) in order to improve coexistence (linked to adjacent/alternate power rejection performances).

Resulting signal is then processed to feed the synthesizer's two modulation points.

41.2.6 External 32 MHz crystal oscillator

A 32 MHz frequency crystal oscillator with integrated load capacitors, tunable in small steps, provides an accurate timing reference for the MCU as well as a clean reference to guarantee radio performances.

41.3 General operation

41.3.1 General transceiver operation

The general operation of the transceiver is as follows:

- Initialize radio for operation
  - Enable clocks and biasing
  - Apply default settings
  - Run calibration routines or recover calibration setting from previous activity time frame
- Apply protocol specific settings
- Configure the MAC for operation

When everything is configured, the packet transmit or receive is initiated at the link layer/MAC level.

The general mechanism for a transmit is as follow:

- MAC has a configuration to indicate how long the radio will take to be ready to transmit and therefore it will request a TX operation, to the radio controller, in advance of the intended transmit time
- Zigbee/Thread Modem indicates required channel and transmit power to radio controller

- Radio controller switches on the radio using the necessary sequences and timing control
- Radio controller indicates to data path that transmit is ready
- MAC starts sending data at the required point
- After packet data has completed then MAC indicates radio can switch off
- Radio controller will switch off the radio using the necessary sequences and timing control

Similarly, the general mechanism for a receive is as follows:

- MAC has a configuration to indicate how long the radio takes to be ready to receive and therefore it will request a RX operation, to the radio controller, in advance of the intended receive time
- Zigbee/Thread Modem indicates required channel
- Radio controller switches on the radio using the necessary sequences and timing control
- Radio controller indicates to data path that radio is ready in receive state
- Modem (Zigbee/Thread) starts searching for a start of packet/access address and in parallel the AGC controls radio gain to give a good signal level to the modem
- If a packet start is detected, the AGC will freeze and the modem will process the packet and output the demodulated data to the MAC where it will be processed
- Following the end of the received packet, the MAC will indicate to the radio controller to turn off the radio and the radio will be switched off using the necessary sequences and timing control
- If no packet is received, then the MAC may automatically turn off the radio when a certain time has elapsed, or software may need to terminate the receive operation

41.3.2 Low power operation

The device offers low-power wireless operation when it is required to transmit and receive. However, for applications that are battery powered, even lower average power consumption is required during sleep period of the Zigbee/Thread protocol. The use of Power-down mode and low-power timers support these applications.

For Zigbee/Thread applications, the low power wake-up timers can be used in power-down modes so that the device can wake from power-down state and return to active mode at the correct time to perform the action required. This action is application specific but could be to check a sensor or to send a packet to indicate that the device is active.

While the device is in the power down state, it is possible to maintain radio calibration data in retention flops, at the cost of a small increase in current consumption in the low-power state. The benefit is that after device wake-up, the radio re-initialization can be quicker because the calibrations do not need to be repeated.

41.3.3 Calibration overview

Several features within the radio are sensitive to one or more of the process, temperature and voltage (i.e. VCO center frequency / gain, receiver DC offset, etc). Embedded calibration schemes allow to compensate for block parameters spread/drift or unwanted block impairment by trimming concern blocks so that they can operate to the required accuracy. These calibrations may be performed during ATE device test or in active mode when the radio functionality is not currently being used.

The calibrations associated with radio operation include:

- Power Amplifier (PA) calibration
- Resistance (R) calibration
- Resistor / Capacitor product (RC) calibration
- Synthesizer calibrations
  - VCO center frequency trimming
  - VCO gain estimation
- Receiver I/Q signal paths mismatch calibration
- Receiver DC offset calibration

These are discussed further in Section 41.8 “Calibrations”.

41.4 Radio controller / RFP

The radio controller consists of several blocks needed to control the radio sequencing, functional blocks tightly linked to the radio such as AGC, TX modulation, RX data path functions (DC offset and IQ imbalance removal). These are described in further detail here.

To expand on the overall functionality of the radio controller, an introduction to the management of transmit and reception is given.

To manage transmit and receive, there is a state machine to control the sequencing of the enables to the radio so that the correct sequence occurs and the radio is ready at the time required by the baseband controller. More specifically, the sequence for a transmit and receive is described.

During a transmit:

- The PLL is enabled and the required carrier frequency is selected
- The transmit blocks are enabled
- The PA is controlled to ramp the output power-up just before packet start, to avoid noise generation at the antenna.
- Data from the baseband/modem is transmitted by modulating the PLL frequency
- After the packet has been transmitted, the PA power is ramped down, to avoid noise generation at the antenna.
- The radio blocks are switched off

During a receive:

- The PLL is enabled and the required carrier frequency is selected
- The receive blocks are enabled
- The AGC is active to control the radio gain until a packet is detected
- The demodulator processes the received data and determines the end of the packet
- The radio blocks are switched off

The calibration mechanisms are controlled by state machines which manage the radio configuration specifically for the calibrations. The results are stored by the PHY controller and then applied to the standard transmit and receive operations.

The results of the calibration are used when a transmit or receive operation occurs. For instance, in receive, the DC offset module uses the calibration results to cancel the expected offsets. Whenever a radio enters transmit or receive, the operating point for the PLL is managed by the hardware by interpolating between the calibration results for the PLL

Calibrations are performed during the radio initialization at application start-up. The calibration repeated if necessary by using the radio API. This may be needed due to large changes in temperature.

During power down mode, it is possible to maintain the results of the radio calibration within the PHY controller so that after wake from the power mode state it is not necessary to repeat the calibrations.

The PHY controller and the associated calibrations are managed by the Radio software drivers.

41.4.1 Radio settings

The analogue blocks within the radio have many settings to configure them correctly. The settings can be grouped in a few categories.

- Static settings; these are driven directly from control registers and are protocol agnotics. Where possible, the reset value of the register is the required setting to reduce SW overhead.
- Radio groups; Operating the radio requires many block enable signals to be sequenced in certain orders and with certain timings when switching on and off the radio for transmit and receive operations. This is achieved by grouping the signals into 9 groups and controlling these to meet the requirements of the radio. This is described further in the TMU (timing management unit) section.
- Calibration control; some radio blocks need to be configured in certain modes to support the calibration mechanisms. Therefore, these radio control settings need to be linked to calibration state machine settings. This is also covered in the TMU section.
- Finally, some radio blocks are required to take different settings for radio operation and calibration and so are controlled from both sets of controllers.

The following diagram shows these control structures and demonstrates how there is always direct software control of the signals on the radio interface.

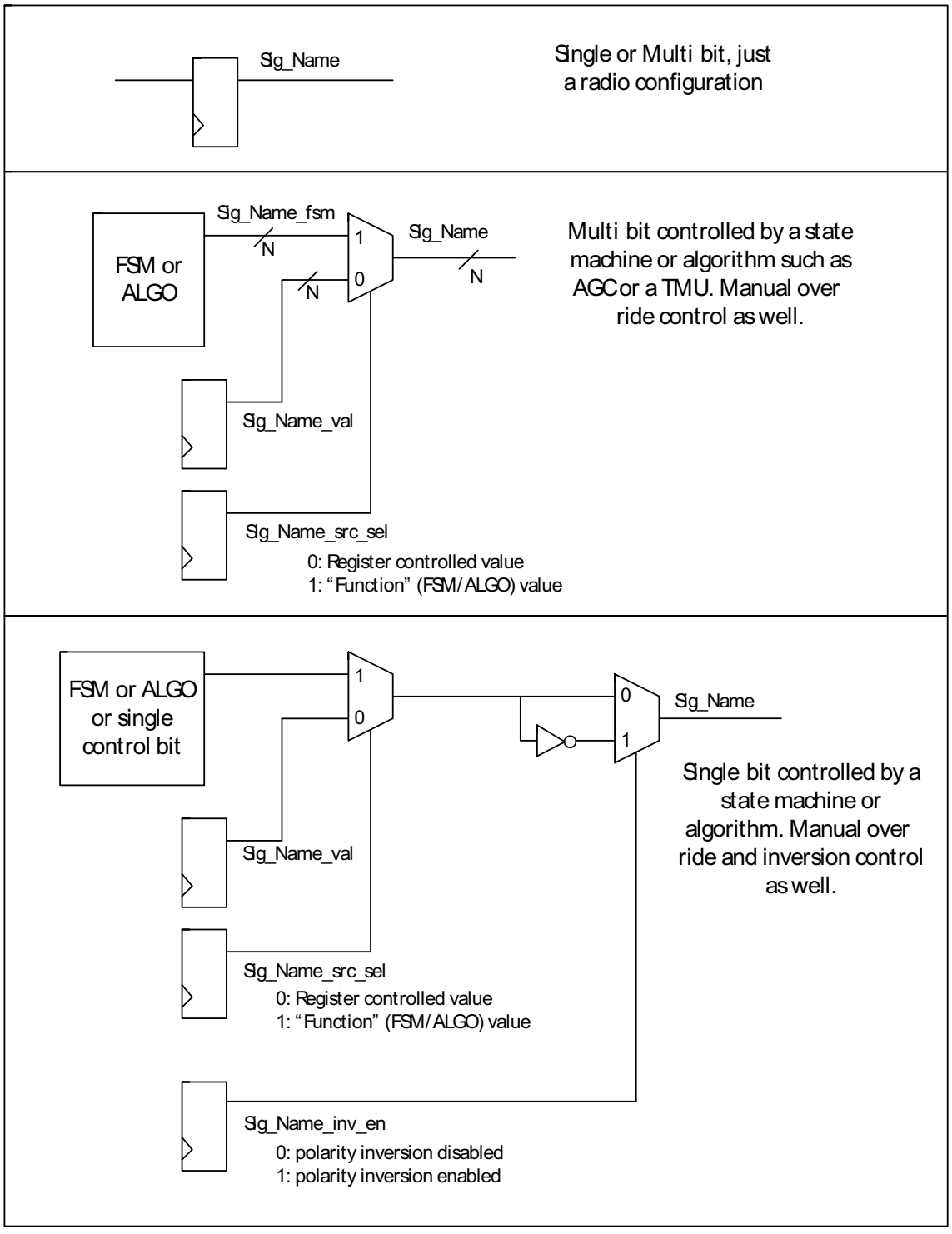


Fig 109. Radio interface control configuration

41.4.2 Timing Management Unit (TMU)

The timing management unit combines the global TMU and some block level TMUs. The global TMU controls the radio groups, see Section 41.4.3 “Radio group controls”, and also triggers the other TMUs. The other TMUs are triggered when required and then interact with the specific functionality that they are linked to.

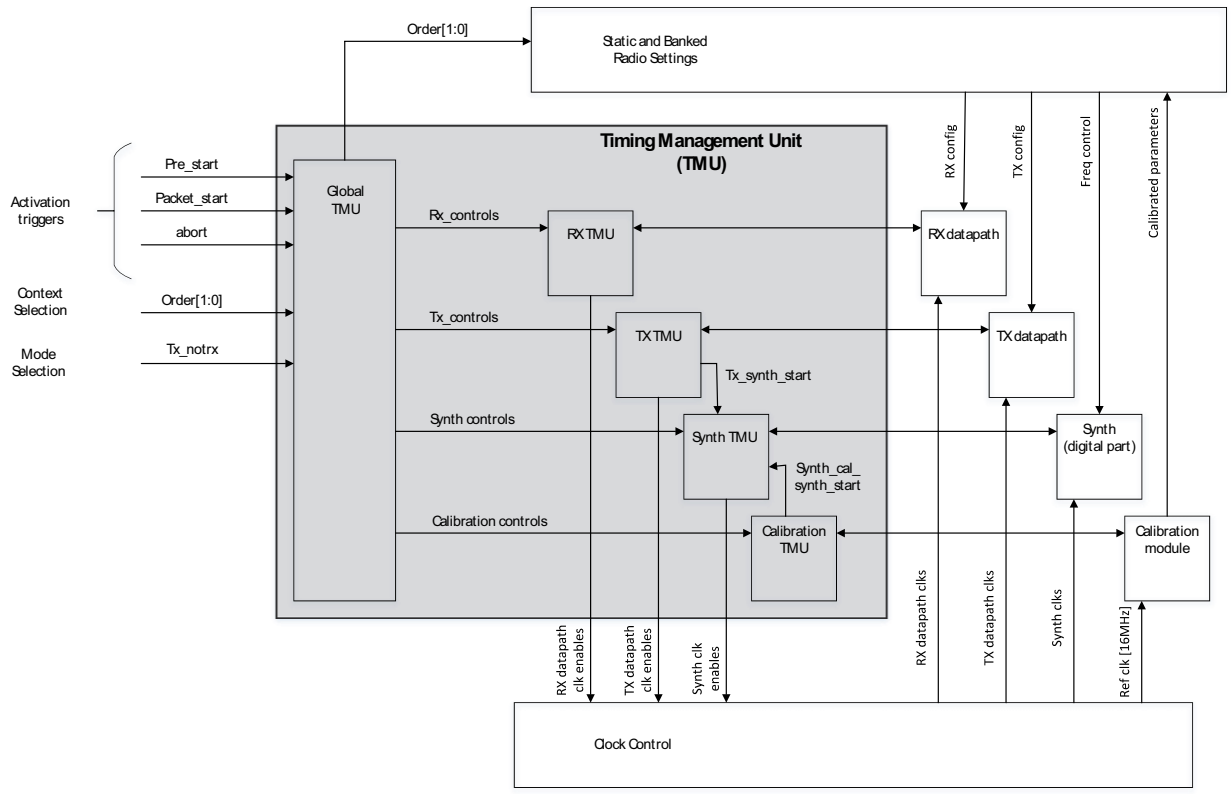


Fig 110. TMU architecture

41.4.3 Radio group controls

For the power sequencing towards the analog, the global TMU controls the radio by 9 groups. Three groups are linked to common blocks independent from Tx/Rx operation: G1, G2 (biasing/reference blocks) and PLL (synthesizer). Three groups are linked to the transmit: TX1 (LDOs), TX2 (RF front-end blocks) and TX3 (output power stage). Three groups are linked to the receive: RX1 (LDOs / ADC), RX2 (IF domain blocks) and RX3 (RF front-end blocks). Each group has a setting to control when it is enabled and then another setting for when the group is disabled. This is shown below:

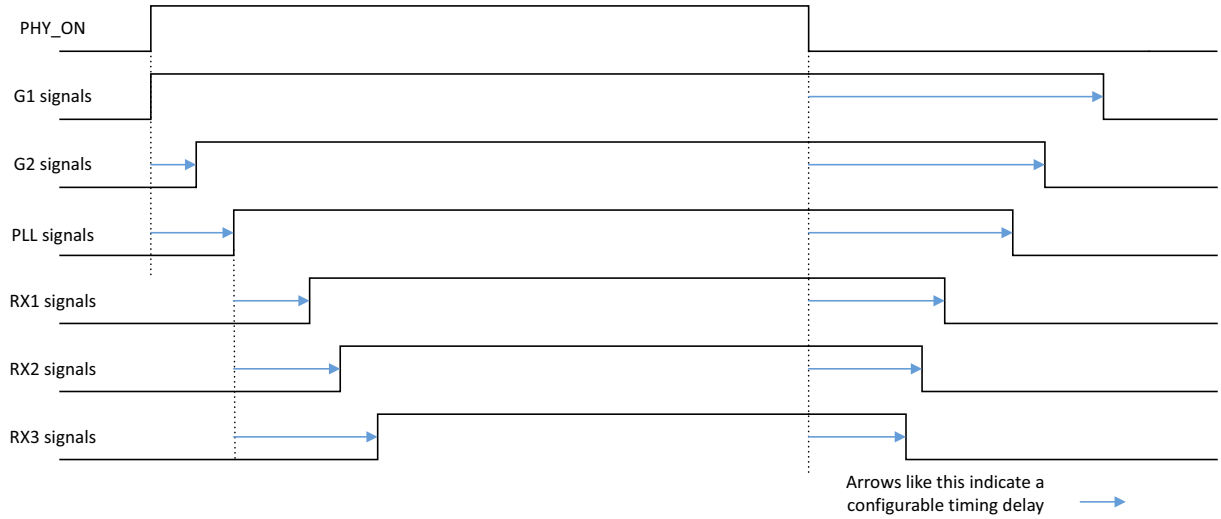


Fig 111. Receive sequence of group enables

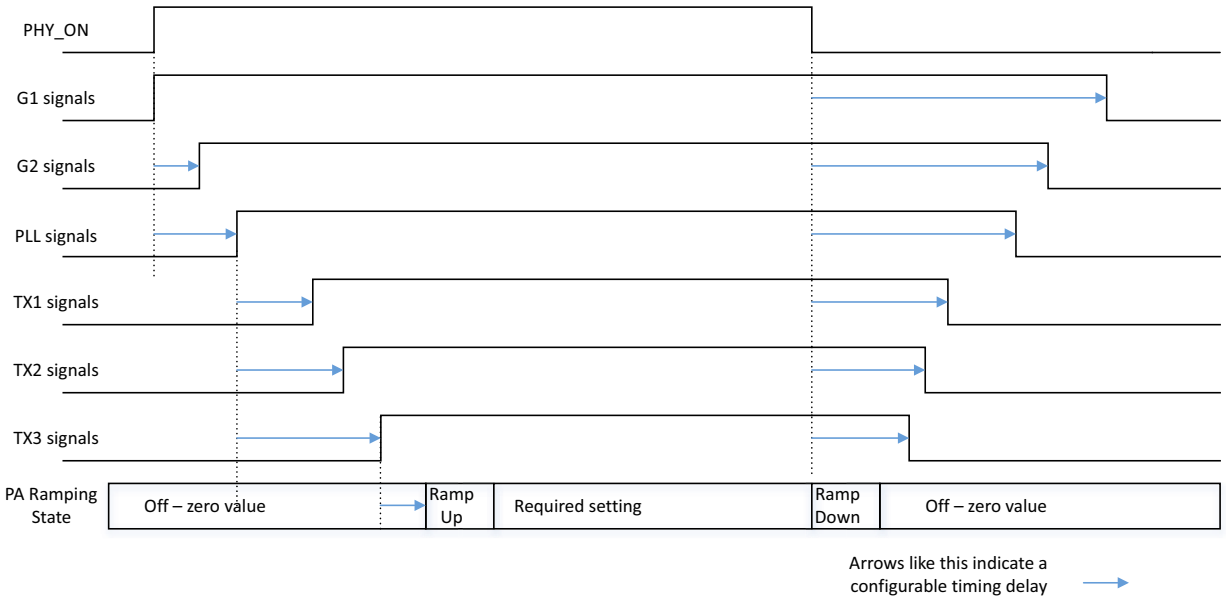


Fig 112. Transmit sequence of group enables

The central FSM of the TMU (Global TMU) decodes and executes the activation triggers received through LL/MAC control output or through register access.

Those activation triggers are:

- rising edge pre_start
- rising edge packet_start
- rising edge abort

Upon reception of one of the listed triggers, the central TMU activates/deactivates one of the other TMU cores depending on the applied operation mode together with this trigger:

- pre_start: This activation trigger is used to start/configure the analog front-end, run calibration loops, etc
- packet_start: This activation trigger is used to start the Tx or Rx data path depending on the selected mode (tx_notrx):
  - tx_notrx = '1': Tx TMU is activated, which will activate and control the Tx Data path
  - tx_notrx = '0': Rx TMU is activated, which will activate and control the Rx Data path
  - Abort: This activation triggers terminates all the current TMU activities..

When the Tx or Rx data path is activated, the global TMU passes also the selected configuration context/bank (tx_order[1:0] / rx_order[1:0]) towards the radio parameters module. For both Rx and Tx datapath up to 3 configuration contexts/banks are supported.

The global TMU also enables the modem/MAC functionality based on a 'packet start' setting from when the PLL group is enabled.

41.4.4 TX datapath

The transmit functions are shown in the next diagram. Two proprietary modes are supported for Zigbee/Thread where the data can also be gaussian filtered (BT=0.5 or 0.95) or for standard Zigbee/Thread with no filtering (equivalent to Half Sine filtered OQPSK modulation after FM modulation).

Then pulse shaped bit-stream feeds an FM modulator based on synthesizer two-point modulation tectonics. Synthesizer calibration data guarantee carrier frequency selection for the optimum phase noise performance while compensating for process spread and temperature drift. VCO gain (Kmod) calibration data guarantee quality of the transmit signal (EVM) by balancing the gain of the two modulation paths. To finish, digital TX datapath balances the delay in between the two modulation paths, too.

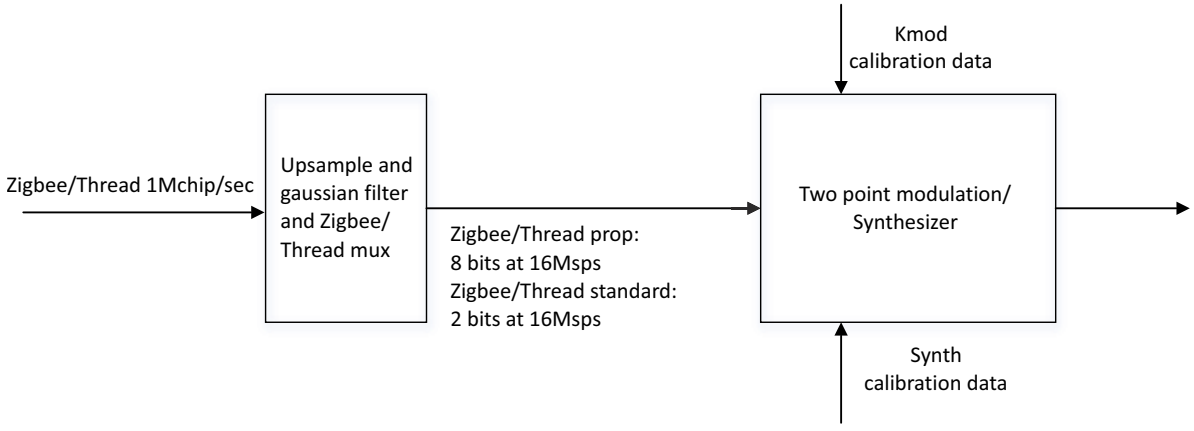


Fig 113. Transmit datapath architecture

41.4.5 RX Datapath

The RX data-path is shown in the next diagram.

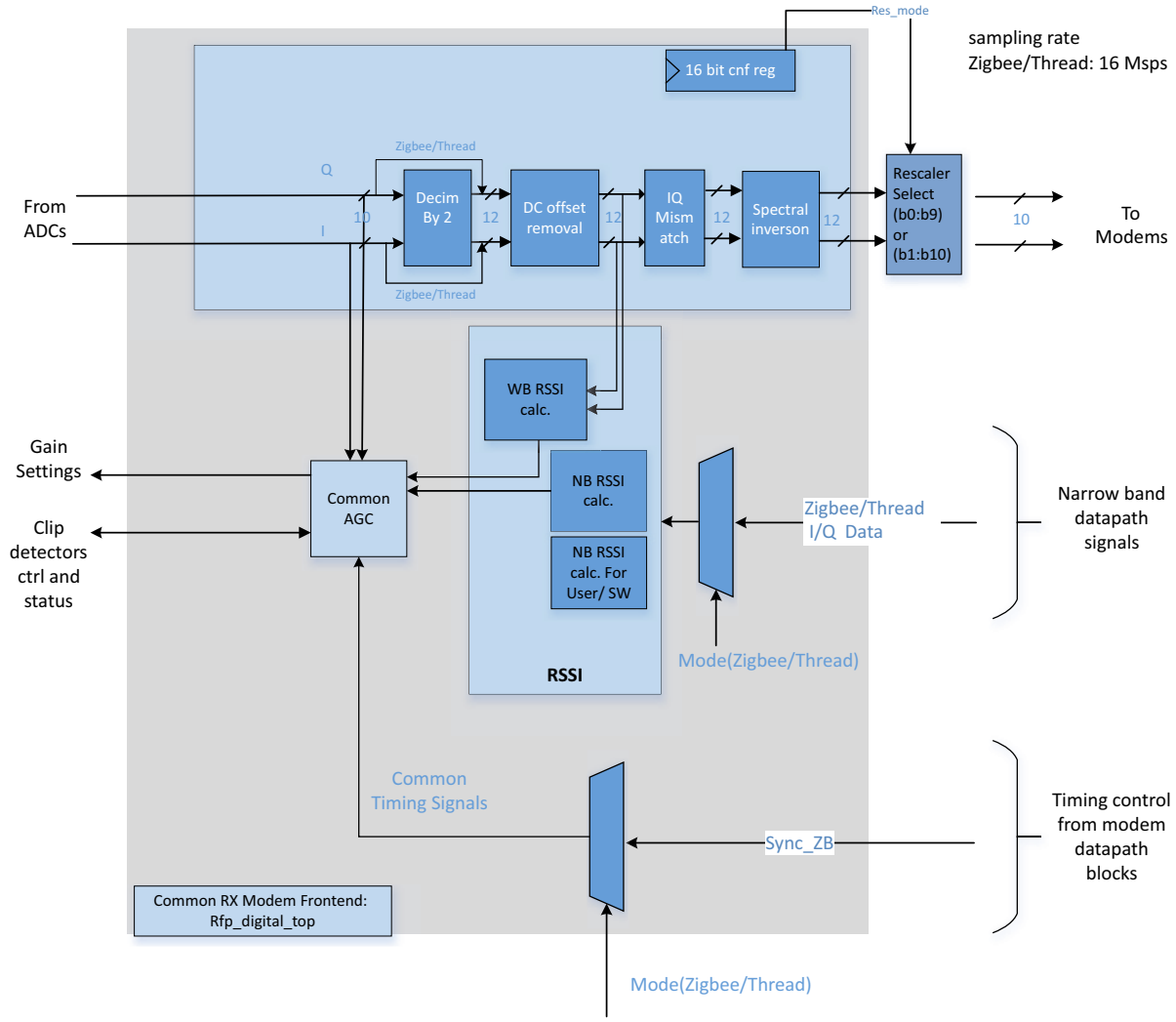


Fig 114. RFP RX Datapath architecture

The main functionality is as follows:

- DC Offset removal: DC-offsets created within the receiver analog part is removed into roughly compensated into analog domain and completely canceled into digital domain. This ensures usage of the full input range of the ADC while offering optimum demodulation performance in presence of unwanted interfere at the antenna.
- I/Q mismatch: if phase errors or gain errors exist within the radio I/Q /signals, then this will affect demodulation in presence of unwanted interfere at the image frequency location.
- Spectral inversion: if required, I and Q channels can be swapped

- AGC: receive chain gain must be controlled, especially at the start of reception to set a good signal to noise level, with headroom for interferer so that channel saturation does not occur. The AGC is fed by clip detector information from radio receive path, RSSI information from the RX digital data-path and modems.
- Received Signal Strength Indication (RSSI) calculation: wide-band (WB) and narrow-band (NB) RSSI is required to guide the AGC, for status reporting after packet reception and clear channel assessment functions.

41.5 AGC

41.5.1 Features

- Peak detector usage to control analog gains
- LUT mode capability based on narrow-band RSSI (low SNR mode)
- Versatile AGC
- Multi stages gain adjustment in analog: LNA gain, IF gain and AAF gain
- Fully programmable to manage analog parameters and convergence time
- ADC output muting when the AAF/IF gain changes (programmable)
- LNA gains range: -36 dB, -24 dB, -18 dB, -12 dB, 0 dB
- IF amplifier gain range: 26 dB, 15 dB, 0 dB
- Anti-aliasing filter (fine) gain range: 18 dB, 15 dB, 12 dB, 9 dB, 6 dB, 3 dB, 0 dB

41.5.2 General description

The AGC is used to control, especially at the start of reception to set a good signal to noise level, with headroom for interferes so that channel saturation does not occur. The AGC is fed by clip detector information from radio receive path, RSSI information from the RX data path and modems.

If any part of the receive chain saturates, then receive signal will be distorted. In these situations, the gain should be reduced. Therefore, it is necessary to be able to quickly detect this situation and three clip detectors have been implemented for this purpose. These are two analog detectors (clip detector 1 and clip detector 2) and also a digital detector post ADC (clip detector 3).

We can see below the general diagram with analog and digital AGC interaction:

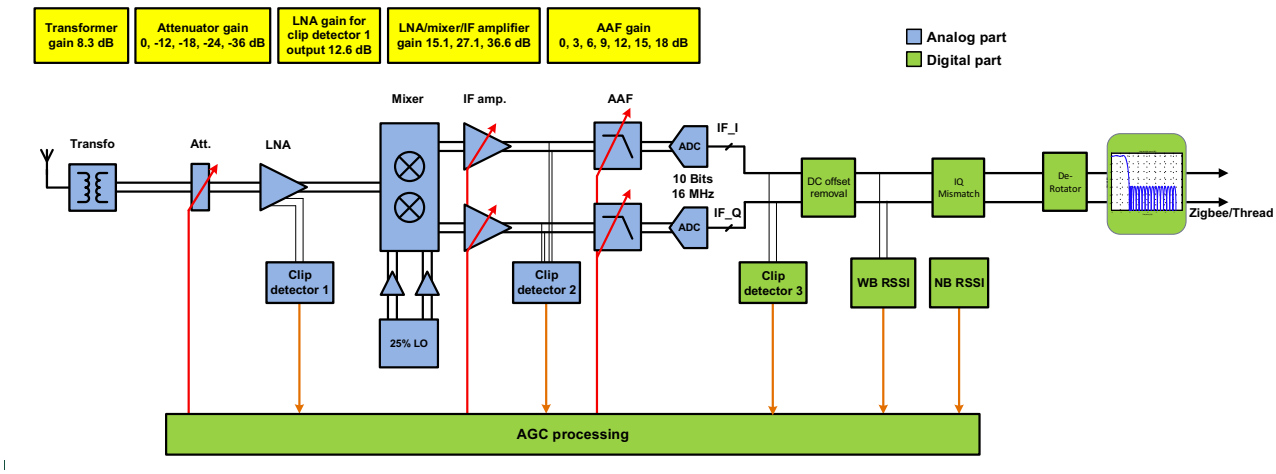


Fig 115. Receiver simplified block diagram for AGC

41.5.3 AGC clip detectors

Two analog clip detectors and a digital detector (post ADC) are used. They have several parameters which can be tuned as described below. They are respectively at output LNA, output IF and output ADC to optimize the gain adjustment depending on at which stage the saturation can occur.

Max. input signal is +10 dBm: the LNA is protected by the clipping in the attenuator in case of +10 dBm input signal for every gain setting of the attenuator.

41.5.3.1 Clip detector 1

The clip detector 1 senses the peak voltage at one of the two outputs of the LNA. This output has 12.6 dB voltage gain.

The clip detector 1 controls the gain of the attenuator.

The time needed to give the clipping information at the detector output when a level step event is present at the detector input is very dependent on the distance between the threshold of the detector and the level of the incoming signal. It is about 50 ns for an incoming signal level 1 dB above the threshold. It can be more than 100 ns if the incoming signal level is very close to the threshold.

By programming rx_datapath.clip_det_1_lev_sel we can adjust the level detection. The clipdet_rst in the figure below comes from digital and is driven when a saturation is on clipdet_out.

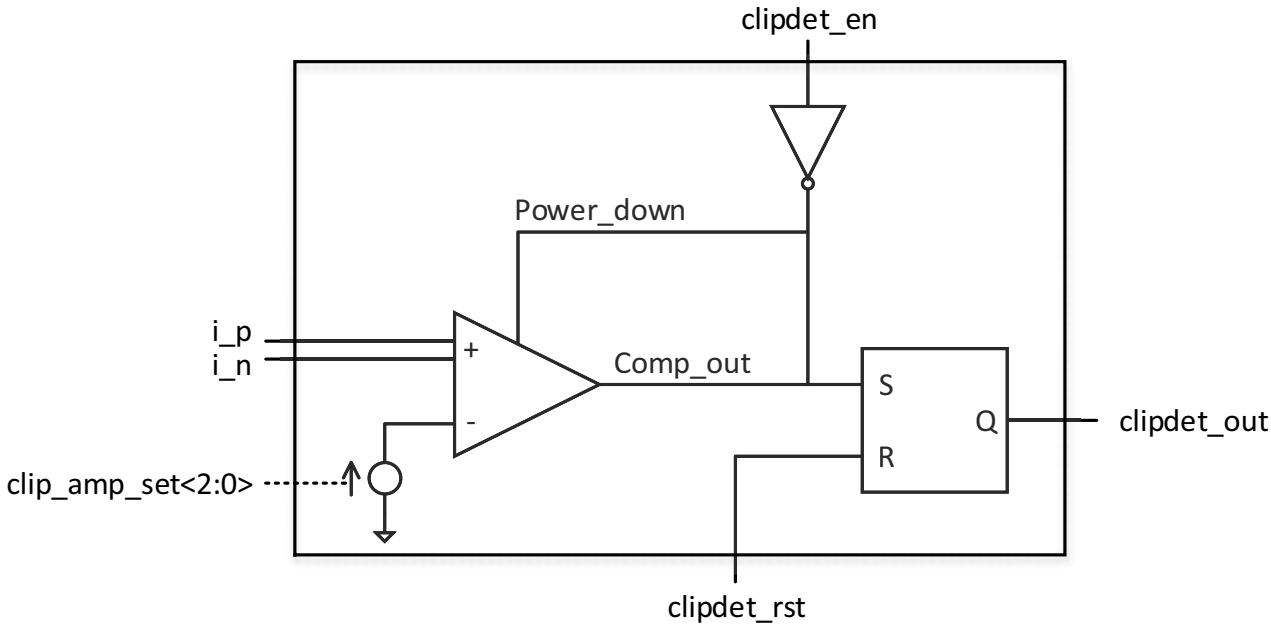


Fig 116. Clip detector 1

41.5.3.2 Clip detector 2

The clip detector 2 senses the signal power at the output of the IF amplifier.

The time needed to give the clipping information at the detector output when a level step event is present at the detector input is very dependent on the distance between the threshold of the detector and the level of the incoming signal. It is about 20 ns for an incoming signal level 3 dB above the threshold. It can be more than 100 ns if the incoming signal level is very close to the threshold.

By programming rx_datapath.clip_det_2_lev_sel we can adjust the level detection.

This detector can be modeled as follows:

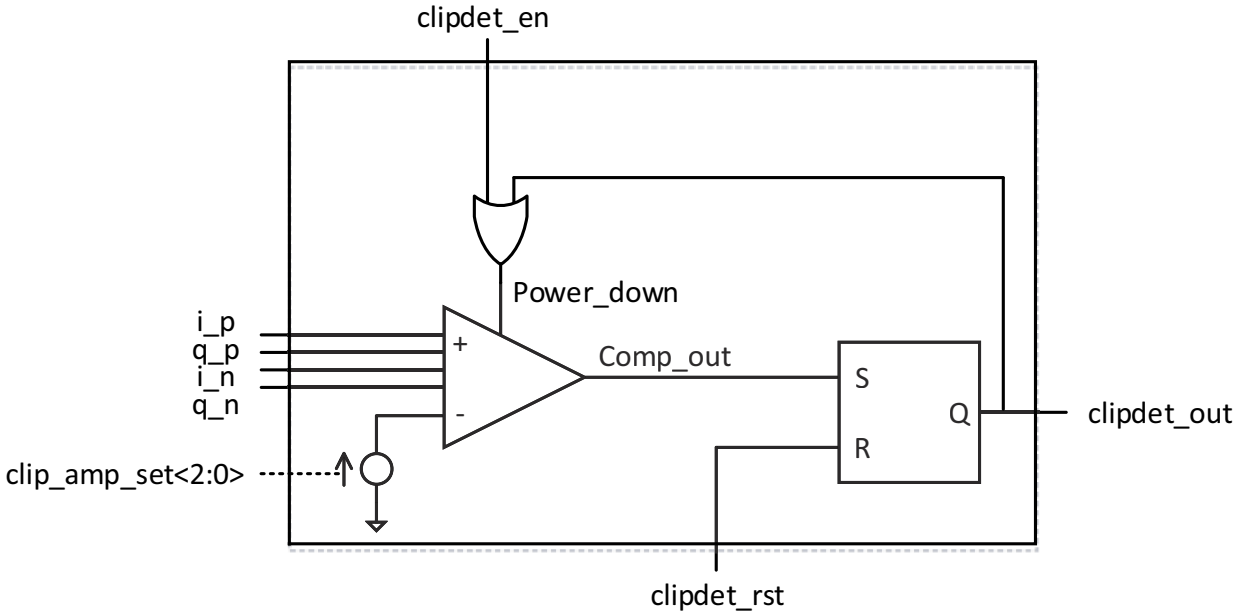


Fig 117. Clip detector 2

41.5.3.3 Clip detector 3 (post ADC)

This detector indicates that clipping occurs if the signal at the ADC output is reaching the ADC full scale e.g. 511.

This limit can be changed with the register rx_datapath.reg_offset_max_adc from 0 to 15. In this case, the new threshold is 511 - (rx_datapath.reg_offset_max_adc) * 16.

The third clip detector monitoring the AAF output level is located in the digital domain, immediately following the ADC, as this gives more information unlike the binary information provided by the analog clip detectors:

1. If saturation is observed (positive or negative clipping), a counter is enabled, which sums up the number of saturated samples during a programmable window
2. Based on the number of saturated samples, the gain is reduced with:
   a. 3dB in case 0 < saturated samples < threshold_1
   b. 6 dB in case threshold_1 ≤ saturated samples < threshold_2
   c. else 9 dB

   Obviously, in case no saturation is observed, no gain reduction 1 in the AAF takes place.
3. This process may be repeated. Note that reducing the gain is conservative in the sense we don't want the gains to be unnecessarily reduced since the clip detector information will be used as upper limit for the fine gain setting based on the wanted signal RSSI. In case of exaggerated gain reduction, no recovery will happen and the signal may get drowned in the noise floor.

The programmable window during which overload events after the ADC are monitored (values equal to either 2^9-1 or -2^9): rx_datapath.clip_det_3_window_size

Thresholds used for calculating the gain reduction are:

rx_datapath.clip_det_3_thr_1 and rx_datapath.clip_det_3_thr_2

Following a gain adjustment of the AAF, no detection is performed in clip detector 3 during: rx_datapath.clip_det_3_timeout

41.6 IEEE 802.15.4 Modem

The modem performs all the necessary modulation and spreading functions required for digital transmission and reception of data at 250 kbits/s in the 2.4 GHz radio frequency band in compliance with the IEEE 802.15.4 standard.

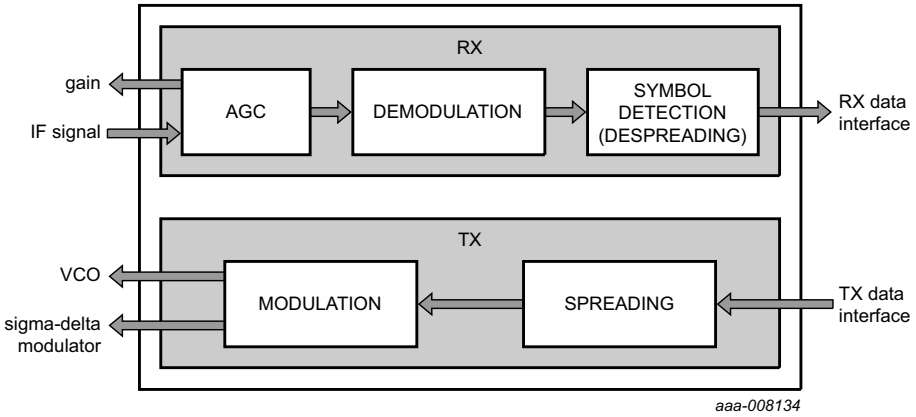


Fig 118. Modem architecture

Features provided to support network channel selection algorithms include Energy Detection (ED), Link Quality Indication (LQI) and fully programmable Clear Channel Assessment (CCA).

The modem provides a digital Receive Signal Strength Indication (RSSI) that facilitates the implementation of the IEEE 802.15.4 ED function and LQI function.

The ED and LQI are both related to receiver power in the same way, as shown in Figure 119, LQI is associated with a received packet, whereas ED is an indication of signal power-on air at a particular moment.

The CCA capability of the modem supports all modes of operation defined in the IEEE 802.15.4 standard, namely Energy above ED threshold, Carrier Sense and Carrier Sense and/or energy above ED threshold.

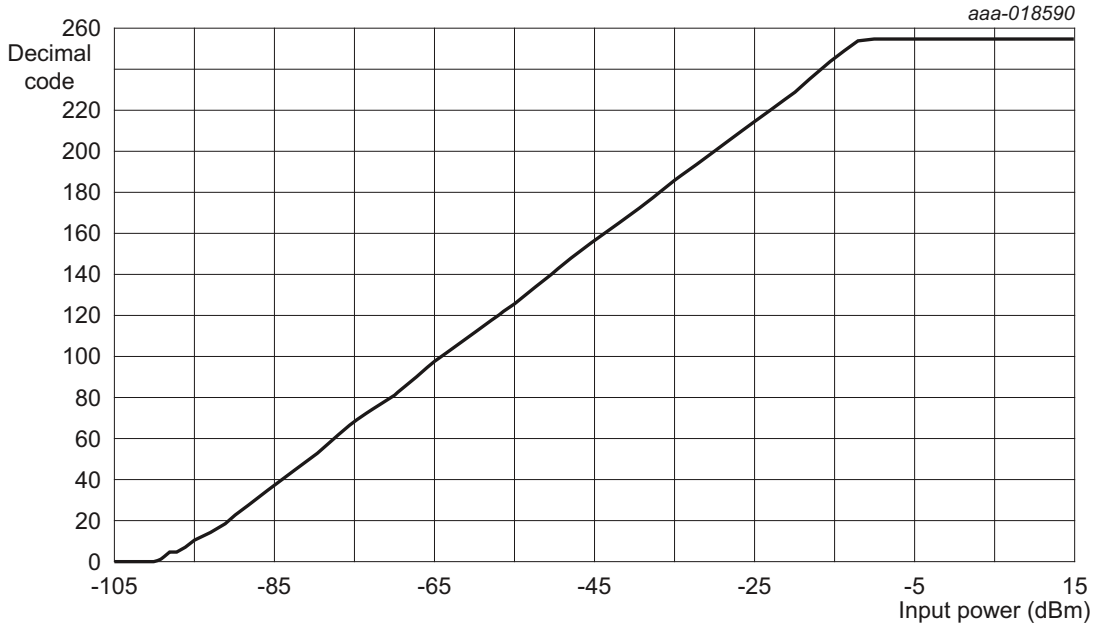


Fig 119. Energy detect (ED) value versus input power level

Other features supported by the modem are:

- Generation of continuous wave (CW) and modulated carried test patterns for radio testing
- Selection of channel
- Selection of transmit power
- Receive packet strength
- Status bits

41.6.1 Modem receiver

The following diagram shows the receive Zigbee/Thread modem data-path.

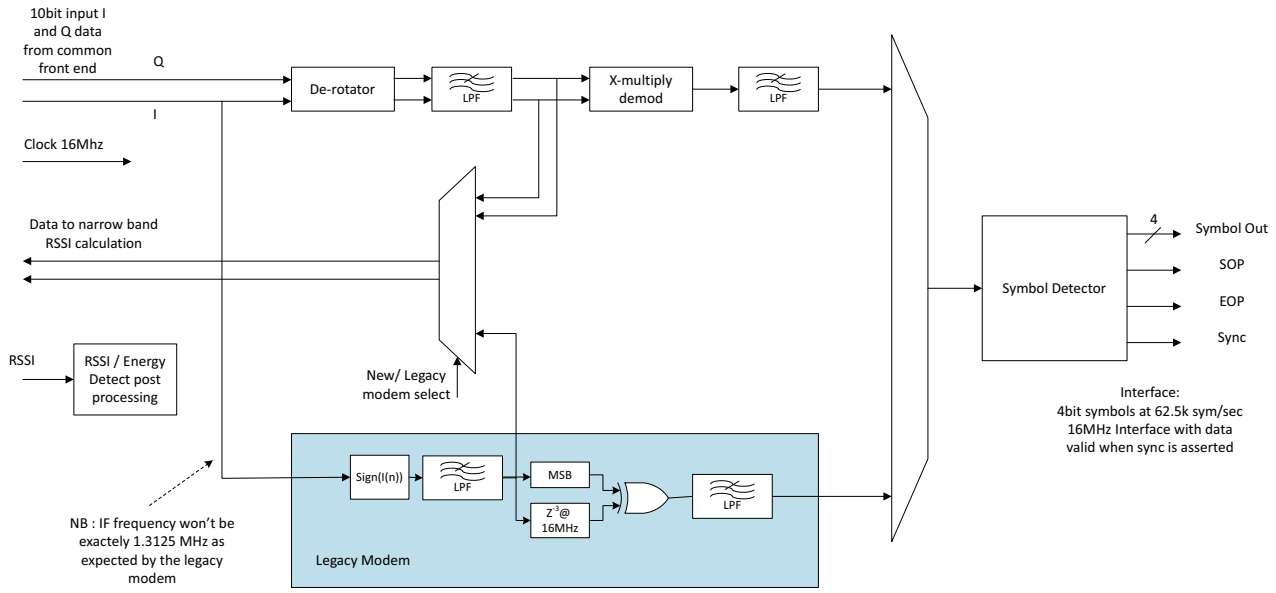


Fig 120. Modem data path

This RX part embeds 2 distinct demodulators: the upper part is the one recommended, the low part is the legacy one and has not been tested/characterized and hence is not supported by the radio driver. The legacy path is simpler, but less efficient and with some constraint on intermediate frequency.

Data entering the modem has come from the common receive path. The main demodulator path (the upper path) consists of a de-rotator that can handle several incoming intermediate frequencies to produce a signal with a fixed IF frequency. The configuration of the radio and this de-rotator must be aligned to match the incoming IF frequency. The demodulation phase is then fed into a symbol detector to perform the decoding.

The symbol detector has a state machine and a few states. Initially, it hunts for the preamble symbol, then assuming a correct symbol alignment it looks for further preamble symbols or the start of frame symbols. Then it is locked for packet reception. False locks, for instance, where preamble or start of frame symbols are not received cause the state machine to return to the hunt state.

The symbol detector interacts with the AGC block to prevent gain changes when packet detection has begun.

The output of the modem goes to the Zigbee/Thread MAC.

41.7 IEEE 802.15.4 MAC

The MAC or baseband processor provides all time-critical functions of the IEEE 802.15.4 MAC layer. Dedicated hardware guarantees that air interface timing is precise. The MAC layer hardware/software partitioning enables software to implement the sequencing of events required by the protocol and to schedule timed events with millisecond resolution and the hardware to implement specific events with microsecond timing resolution. The

protocol software layer performs the higher-layer aspects of the protocol, sending management and data messages between End Device and Coordinator nodes, using the services provided by the baseband processor. The MAC block diagram is shown below:

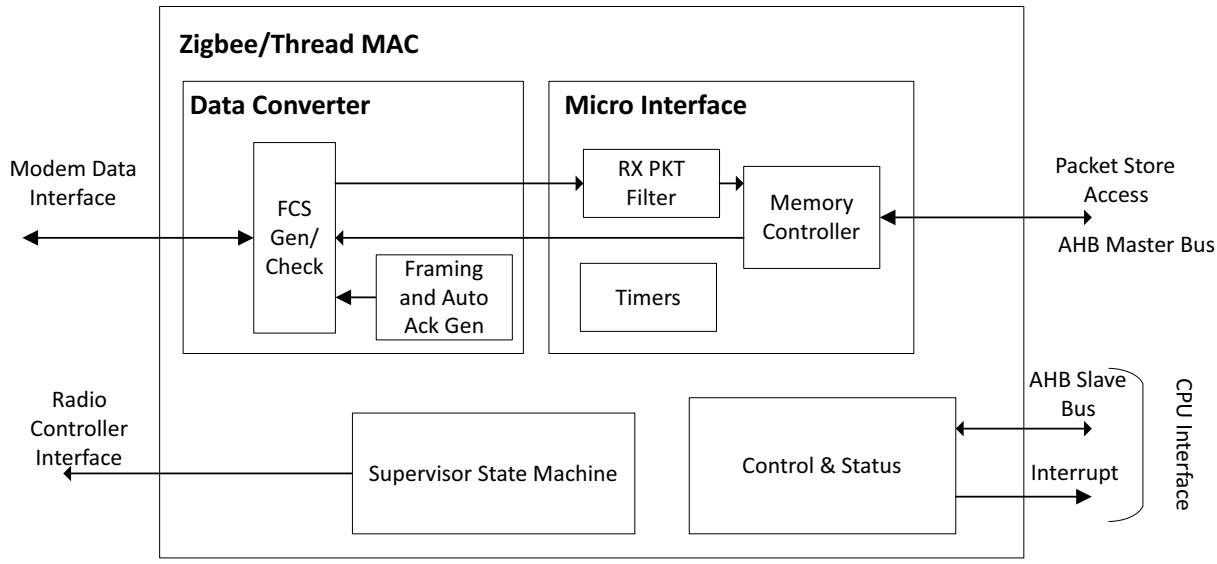


Fig 121. Zigbee/Thread MAC block diagram

41.7.1 Transmit

A transmission is performed by software writing the data to be transferred into the TX frame buffer in RAM, together with parameters such as the destination address and the number of retries allowed, as well as programming one of the protocol timers to indicate the time at which the frame is to be sent. This time will be determined by the software tracking the higher-layer aspects of the protocol such as superframe timing and slot boundaries. Once the packet is prepared and protocol timer is set, the supervisor block controls the transmission. When the scheduled time arrives, the supervisor controls the sequencing of the radio and modem to perform the type of transmission required, fetching the packet data directly from RAM. It can perform all the algorithms required by IEEE 802.15.4 such as CSMA/CA without processor intervention including retries and random back-offs.

When the transmission begins, the header of the frame is constructed from the parameters programmed by the software and sent with the frame data through the serializer to the modem. At the same time, the radio is prepared for transmission. During the passage of the bit-stream to the modem, it passes through a CRC checksum generator that calculates the checksum on-the-fly, and appends on it to the end of the frame.

41.7.2 Reception

During reception, the radio is set to receive on a particular channel. On receiving data from the modem, the frame is directed into the RX frame buffer in RAM where both header and frame data can be read by the protocol software. An interrupt may be provided on receiving the frame. An additional interrupt may be provided after the transmission of an acknowledgment frame in response to the received frame, if an acknowledgment frame

has been requested and the auto acknowledge mechanism is enabled, see Section 41.7.3. As the frame data is being received from the modem, it is passed through a checksum generator; at the end of the reception, the checksum result is compared with the checksum at the end of the message to ensure that the data has been received correctly. During reception, the modem determines the Link Quality, which is made available at the end of the reception as part of the requirements of IEEE 802.15.4.

41.7.3 Auto acknowledge

Part of the protocol allows for transmitted frames to be acknowledged by the destination sending an acknowledge packet within a very short window after the transmitted frame has been received. The JN5189 baseband processor can automatically construct and send the acknowledgment packet without processor intervention and hence avoid the protocol software being involved in time-critical processing within the acknowledge sequence. The JN5189, baseband processor can also request an acknowledge for packets being transmitted and handle the reception of acknowledged packets without processor intervention.

41.7.4 Security

The transmission and reception of secured frames using the Advanced Encryption Standard (AES) algorithm is handled by the stack software and the AES module.

The application software must provide the appropriate encrypt/decrypt keys for the transmission or reception. On transmission, the stack software will encrypt the packet and then use the baseband processor to transmit the encrypted packet. On reception, the baseband processor will store the received encrypted packet in the packet buffer in SRAM. Then stack software will then use the baseband processor to decrypt the packet.

41.7.5 Additional features

The MAC / Baseband processor supports various additional features or controls as follows:

- Configuration of transmit and receive buffer address
- CCA configuration, back-off control (CCA mode and threshold configured in modem)
- Number of transmit retires allowed
- Transmit timeout control, used to prevent deadlock
- Receive address matching configuration
- Receive status of fcs error, abort, other errors
- Symbol timers with four match registers
- Timestamp record for transmit and receive
- Interrupt support for RX address match, address failure, timeout, symbol timer match, receive header, receive packet, transmit completed
- Transmit status: timeout, CCA busy, no acknowledgment
- Transmit control, instant transmit, transmit with CCA, slotted transmit

41.7.6 System integration

41.7.6.1 Power domain

Zigbee/Thread parts are localized in power domain MCU. When this power domain is switched off, which happens in power down mode for example, all internal settings are lost; the only option is to retain certain specific settings within the radio controller. See dedicated chapter in power modes on how to apply retention.

41.7.6.2 Memory mapping

Zigbee/Thread data-path has rights to access the whole of the SRAM, through a movable 128 KB window. If the data-path tries to access addresses outside the SRAM address space, it will lead to system failure.

41.8 Calibrations

41.8.1 Introduction

Various aspects of the radio need to be calibrated to take account of process variations and in some cases temperature variation.

The radio software driver manages the calibration of the radio and the re-use of retained values as part of the radio initialization functions.

Internal temperature sensor information must be provided by the Application software to the radio software driver to get optimum RF performance over temperature.

41.8.2 Calibrated parameters

Calibration is the process of determining radio parameters for each device. All calibrations either have the results stored in flash during device factory testing, or in retention registers for maintaining the values during a power down cycle. Those calibration parameters are read back by driver upon device initialization and sent back to the device if not already in retention registers. When retention registers already set, which means that calibration is not required after a power down cycle if temperature drift doesn't exceed a given threshold. When this threshold is breached, a subset of calibration results is updated by radio software driver.

Calibrated parameters independently of the temperature:

- Resistor calibration allows better control of current consumption spread.
- Resistor/Capacitor time constant product calibration guarantees filtering cut-off frequency control. Better interfere rejection control is got in such way.
- RF bandgap reference calibration allows better control of current consumption spread.
- I/Q mismatch calibration compensates for phase and magnitude errors in the I and Q channels of the radio receiver. This provides better robustness to interferer located at image frequency.

- PMU bandgap calibration. The PMU is outside the radio but it supplies an accurate bandgap reference to the radio.

Calibrated parameters updated over temperature

- Synthesizer calibration provides accurate frequency selection in the VCO, while providing good phase noise performance and temperature drift effect compensation.
- VCO gain calibration balances the gain of the two-modulation path driving the synthesizer to control modulation characteristics / quality of the transmitter. Then EVM performances are guaranteed.
- Receiver DC-offset calibration compensates for any DC component in the I and Q channels. It guarantees the interferer robustness together with RSSI accuracy.

41.8.3 Radio initialization and temperature management

The radio software driver is triggered by the MAC when radio operation is requested.

The radio driver can manage recalibration if current temperature deviates too much from last calibration temperature.

It needs actions from the application and from the MAC.

For that, the application should regularly provide current temperature to the radio driver using vRadio_Temp_Update(int16_t s16Temp) API function where s16Temp is the current temperature in signed 12-bit 11.1 format (i.e. value is given in half of °C).

Examples:

- if the temperature is +20°C, the value of s16Temp is 0x0028
- if the temperature is -10°C, the value of s16Temp is 0xFFEC

Moreover, MAC software must periodically call into radio driver to check for the need to re-calibrate; this should be done between regular radio activities. Then the radio driver can perform re-calibration if temperature drifts since last calibration is substantial. If no calibration is needed, then the radio driver exits function immediately.

An internal “always on” register is used to maintain temperature context (current temperature (T_C), temperature of last calibration (T_{CAL}), calibration request flag and a few other flags such as current temperature received and initial calibration done flags).

During the first radio initialization function call by MAC, a calibration is launched and the last calibration temperature (T_{CAL}), is updated with current temperature (T_C). So application software must update the current temperature information (T_C) at device start-up and before first radio initialization. Otherwise a default value is used for T_C (device factory test temperature, about 26°C). This value is stored in flash page "N-2" with other parameter values coming from device factory calibration process.

At next radio initialization function call, after a power down cycle for instance, current temperature (T_C) is compared to last calibration temperature (T_{CAL}) to decide if calibration should be rerun.

Current temperature (T_C) is updated each time vRadio_Temp_Update() API is called. It is compared to the temperature of the last calibration (T_{CAL}), and if the absolute difference between these two temperatures is greater than a threshold (default threshold value is 40°C), a calibration request flag is raised and written to the "always on" register.

Each time calibration is run, the calibration flag is reset and last calibration temperature (T_{CAL}) is updated with current temperature (T_C).

41.8.4 Radio initialization and retention registers management

After power-on of the chip (cold start), calibration parameters that do not change with temperature are read back from flash by driver during radio calibration process. Depending on temperature, other parameters are read back from flash or are set by hardware calibration algorithms that are triggered by the driver during the calibration procedure.

Calibration results can be maintained in registers that hold their values during power-down mode, these are called retention registers. After a power-down cycle where retention has been used, it is only necessary to initialize the radio. If retention is not used, then full calibration will be required during sleep modes.

When radio initialization function is called by MAC, the radio driver checks if retention values are available by comparing the contents of a set of calibration registers with their reset value. If the register contents differ from reset values, the driver assumes that retention values are available; in which case calibration is not needed again.

If retention values aren't available then calibration procedure is run whatever temperature.

If temperature change is higher than threshold, calibration process is run whatever the result of retention availability test.

41.8.5 Calibration time duration

The hardware calibration process needs time to execute. Main contributors are Receiver DC-offset compensation, VCO gain calibration and Synthesizer calibration.

These algorithms have been optimized to minimize radio initialization time.

However, radio initialization time is significantly longer when VCO gain calibration algorithm is triggered.

To minimize VCO gain calibration time effect, the values resulting from this calibration are saved in flash with associated calibration temperature. That way, this specific calibration need to be run only 3 or 4 times in the life of the chip (to cover the whole temperature range). Once the hardware calibration has been launched for a given temperature, the result can then be re-used for all temperature within ±40°C range by using results saved in flash.

Based on the combinations between the need of calibration or not, and the availability of VCO gain calibration parameters in flash, the initialization time can have 3 values as indicated by table below.

Table 104. Radio initialization time

Type of calibration	Reason	Duration	Occurrence
No calibration needed, retention values available	Warm start after low-power mode with retention	290 µs	High
Calibration with VCO gain calibration parameters available in flash	Cold start, temperature change, warm start after low power without retention.	5.1 ms	Low
Calibration with new VCO gain calibration needed	Cold start, temperature change, warm start after low power without retention, all these at a temperature never reached before during the life of the product.	76.2 ms	Very low (can happen only 3 or 4 times in the product life)

These initialization time can be found defined as macros in radio.h file:

```
#define RADIO_INIT_TIME_WITH_KMOD_INIT_US (76210)

#define RADIO_INIT_TIME_WITHOUT_KMOD_INIT_US (5110)

#define RADIO_INIT_TIME_WITH_RETENTION_US (290)
```

41.9 Antenna Diversity

Antenna diversity is a technique that maximizes the performance of an antenna system. It allows the radio to switch between two antennas that have very low correlation between their received signals. Typically, this is achieved by spacing two antennae around 0.25 wavelengths apart or by using two orthogonal polarizations. So, if performance is poor, the radio system can switch to the other antenna, with a different probability of success.

In Zigbee/Thread operation, using transmit diversity mode, if a packet is transmitted and no acknowledgment is received, the radio system can switch to the other antenna for the retry. Alternatively, antenna diversity can be enabled so that antenna switching will occur in receive mode when waiting for a packet. Receive diversity operates a combined HW timer and SW power threshold mode. In general, the antenna is switched every 60 µs. However, if two preamble symbols are detected, then the antenna switching stops; the software will check whether the signal strength exceeds a threshold. If the signal is too weak, then the antenna selected is switched and the automatic switching will restart. If the signal is strong, the packet reception will continue. The overall system performance depends upon various factors such as the attenuation / isolation between the two antennas, the RF characteristics of the signals received on each antenna.

The JN5189(T)/JN5188(T) provides an output (ADO) on DIO7, DIO9 or DIO19 and optionally its complement (ADE) on DIO6, that can be used to control an antenna switch; this enables antenna diversity to be implemented easily (see the following two figures).

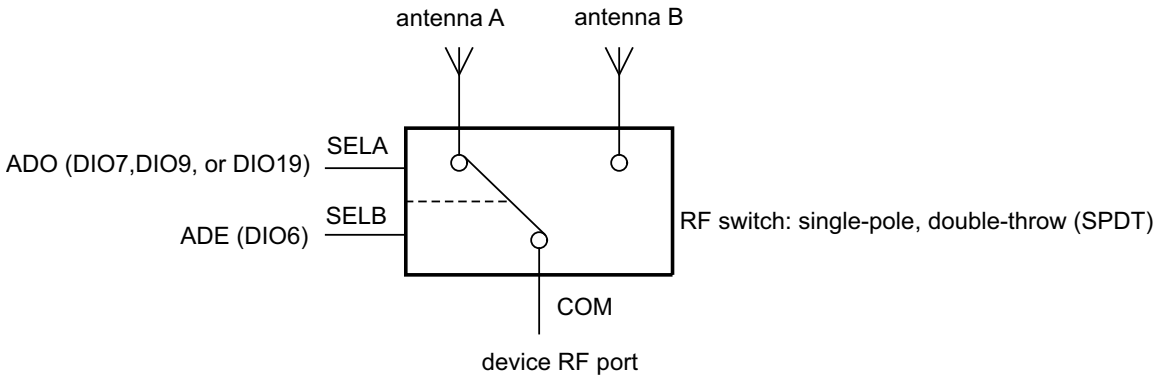


Fig 122. Simple antenna diversity implementation using external RF switch

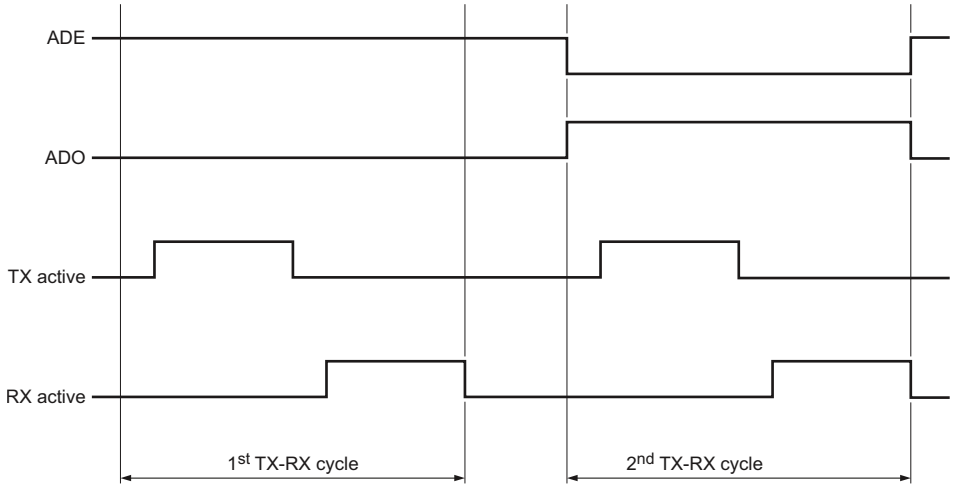


Fig 123. Zigbee/Thread antenna diversity ADO and ADE signals for TX with acknowledgement

If only one DIO pin can be used, then either ADE or ADO can be connected to the first switch control pin and the same signal inverted on the second pin with an inverter on the PCB.

41.10 Radio TX/RX

Two GPIO pins can optionally be used to provide control signals for RF circuitry (e.g. switches and PA) in high power range extenders.

GPIO8, GPIO4 or GPIO20 /RFTX is asserted when the radio is in the transmit state and similarly, GPIO5, GPIO15 or GPIO21/RFRX is asserted when the radio is in the receiver state.

Example waveforms of RFTX and RFRX are shown in Figure 123 as the TX active (RFTX) and RX active (RFRX) signals

41.11 Radio controller APIs

To simplify use of the radio, a software radio controller library is provided, this should be used to control the radio. The main initialization function, vRadio_RadioInit controls the 32M XTAL, radio settings, managing calibrations. The function vRadio_Standard_Init is used to configure protocol specific settings in the radio controller and where necessary in the individual modems.

The radio controller APIs also support accessing RSSI information and modes / settings to support test activities. The XTAL32M is enabled outside the Radio controller libraries, with CLOCK_EnableClock(kCLOCK_Xtal32M). Within the radio APIs, the XTAL configuration needs to be modified and so within the Radio Initialization functions the function vRadio_ActivateXtal32MRadioBiasing() will be called.

41.12 PHY Controller

To expand on the overall functionality of the radio controller, an introduction to the management of transmit and reception is given.

To manage transmit and receive, there is a state machine to control the sequencing of the enablement to the radio so that the correct sequence occurs and also so that the radio is ready at the time required by the baseband controller. More specifically, the sequence for a transmit and receive is described.

During a transmit:

- The PLL is enabled and the required carrier frequency is selected
- The transmit blocks are enabled
- The PA is controlled to ramp the output power up just before packet starts, to avoid noise generation at the antenna.
- Data from the baseband/modem is transmitted by modulating the PLL frequency
- After the packet has been transmitted, the PA power is ramped down, to avoid noise generation at the antenna.
- The radio blocks are switched off

During a receive:

- The PLL is enabled and the required carrier frequency is selected
- The receive blocks are enabled
- The AGC is active to control the radio gain until a packet is detected
- The demodulator processes the received data and determines the end of the packet
- The radio blocks are switched off

The calibration mechanisms are controlled by state machines which manage the radio configuration specifically for the calibrations. The results are stored by the PHY controller and then applied to the standard transmit and receive operations.

The results of the calibration are used when a transmit or receive operation occurs. For instance, in receive, the DC offset module uses the calibration results to cancel the expected offsets. Whenever a radio enters transmit or receive, the operating point for the PLL is managed by the hardware by interpolating between the calibration results for the PLL.

Calibrations are performed during the radio initialization at application start-up. The calibration repeated if necessary by using the radio API. This may be needed due to large changes in temperature.

During power-down mode, it is possible to maintain the results of the radio calibration within the PHY controller so that after wakeup from the power mode state it is not necessary to repeat the calibrations.

The PHY controller and the associated calibrations are managed by the Radio software drivers.

42.1 How to read this chapter

This chapter presents the device features where an internal NTAG device is fitted. The JN5189T and JN5188T have an internal NTAG device.

42.2 Features

- NTAG I^2C Plus device NT3H2211 is fitted
- Connections to external NFC antenna are provided through device pins LA and LB.
- NFC NTAG is powered from the core die and needs software control to enable it
- I^2C2 module is connected to the NTAG I^2C interface
- Field detect pin from the NTAG can be used to wake the device from power-down and deep power-down
- When the device is in power-down state, the NTAG device can still be read and written using the NFC antenna

42.3 Basic configuration

Configuration to use the NTAG from CPU is accomplished as follows:

- Provide power to the NTAG device by configuring ASYNC_SYSCON_NFCTAGPADSCTRL register and also set the output using ASYNC_SYSCON_NFCTAG_VDD register.
- Enable the I^2C2 interface to make communication with the NTAG possible
- Configure the I^2C2 IO cells that connect to the NTAG, using ASYNC_SYSCON_NFCTAGPADSCTRL register.
- Use the functionality of the I^2C2 interface to read or write to the NTAG device

To use the NTAG from the NFC antenna, configure as follows:

- Bring NFC reader / writer close to the NFC antenna; energy coupled through the antenna will activate the NTAG device
- Perform reader or write operation through the NFC antenna. No interaction with the internal CPU is necessary

42.4 Pin description

The device pins LA and LB are connected directly to the NTAG device; no configuration is required. An NFC antenna must be connected to these device pins. Refer to the NTAG data sheet for guidance on the specification of the antenna. Additionally, layout guidelines and the requirement of any necessary matching components should also be considered.

42.5 General description

The architecture of the core die and NTAG within the device is shown in Figure 124. For functional details of the NT3H2211 NTAG I²C Plus device, see the NXP NT3H2111/NT3H2211 Data Sheet. All features of the NT3H2211 are supported with the exception of the energy harvesting features.

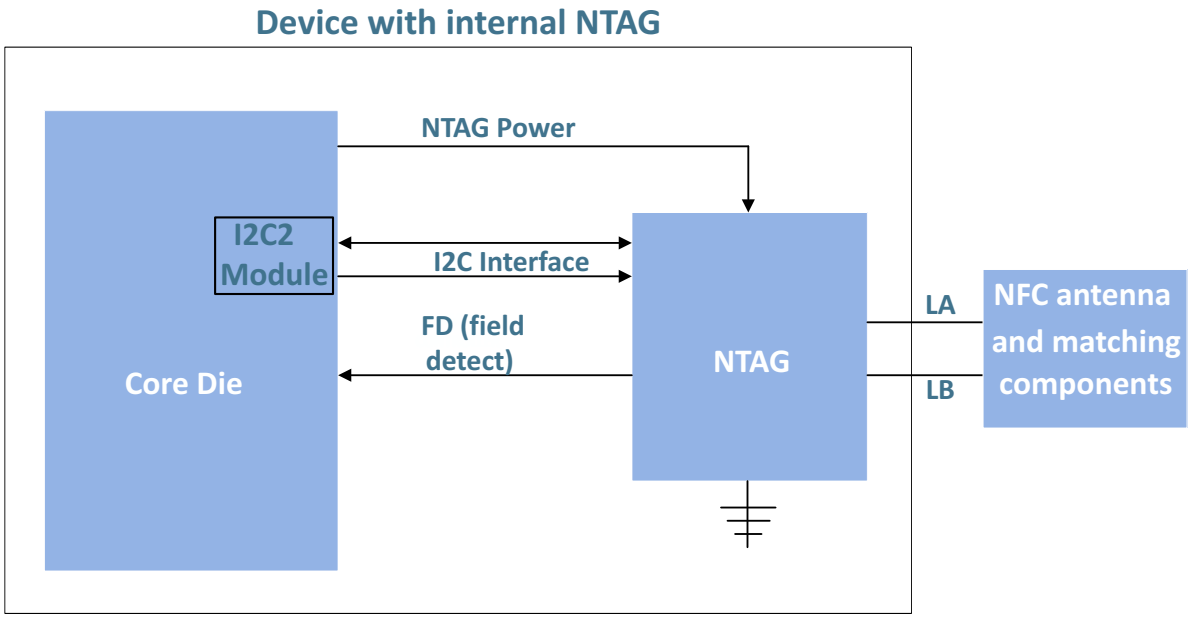


Fig 124. Architecture of device with internal NTAG

42.6 Functional description

42.6.1 Providing power to the NTAG device

The following tables show how to enable the power to the NTAG device

Table 105. NTAG power/VDD settings in ASYNC_SYSCON_NFCTAGPADSCTRL

Control signal	Function	Setting
VDD_EDN	Enable weak pull-down	0 (disable)
VDD_EPUN	Enable weak pull-up (active low)	0 (enable)
VDD_EHS0	Driver slew rate LSB	1 (set high slew rate)
VDD_EHS1	Driver slew rate MSB	1 (set high slew rate)
VDD_OD	Pseudo open-drain enable	0 (standard IO mode)

Table 106. NTAG power/VDD settings in ASYNC_SYSCON_NFCTAG_VDD

Control signal	Function	Setting
NFCTAG_VDD_OUT	Output value for the NFC Tag VDD IO, if enabled with NFCTAG_VDD_OE.	1 (Set output high)
NFCTAG_VDD_OE	Output enable for the NFC Tag VDD IO cell.	1 (enable IO cell)

Once the NTAG is powered, it is necessary to configure the IO cells associated with the NTAG/I²C interface see Section 42.6.2.

42.6.2 Configuring the I²C Interface

There are dedicated IO cells for connecting to the embedded NTAG device. These are configured in the NFCTAGPADSCTRL register as shown in the following table.

Table 107. NTAG I2C_SLA and I2C_SDA IO cells

Control signal	Function	Setting
I2C_XXX_EPD	Enable weak pull-down on IO pad. • 0= disabled. • 1=pull-down enabled	0 (disable pull-down)
I2C_XXX_EPUN	Enable weak pull-up on IO pad, active low • 0=pullup enabled • 1=pullup disabled	0 (pullup enabled)
I2C_XXX_EHS0	IO Driver slew rate LSB. (I2C_SDA_EHS1, I2C_SDA_EHS0). RESERVED: use default value (0)	0 (reserved value)
I2C_XXX_INVERT	Input polarity: • 0: Input function is not inverted; • 1: Input function is inverted	0 (no inversion on input)
I2C_XXX_ENZI	Receiver enable, active high	1 (enable receiver/input path)
I2C_XXX_FILTEROFF	input glitch filter control: • 0: Filter enabled. Short noise pulses are filtered out; • 1: Filter disabled. No input filtering is done	0 (enable filter for robust operation)
I2C_XXX_EHS1	IO Driver slew rate MSB. (I2C_SCL_EHS1, I2C_SCL_EHS0). RESERVED: use default value (0)	0 (reserved value)
I2C_XXX_OD	open-drain mode control: • 0: Normal push-pull output; • 1: Open-drain. Simulated open-drain output (high drive disabled).	1 (use the pseudo open-drain mode for I2C signaling)

Once the NTAG is powered and the I²C IO cells are configured, then the I²C interface can be enabled and used to read/write the NTAG. This I²C interface is I²C2 module and controlled the same way as the other I²C interface, see Chapter 25 “Inter-Integrated Circuit (I²C)”.

42.6.3 Field detect device wake-up

The NTAG device has an output, field detect (FD), that will be asserted when an NFC field is in close proximity to the NFC antenna. This signal is connected to the core die and can be used to generate wake-up event from a power-down state or a deep power-down state.

Before entering either of these power-down states, it is necessary to enable the event to trigger a wake-up event. The wakeup event will be on either the rising edge or falling edge of the FD signal.

To configure for wake-up from deep-sleep, the SYSCON_STARTER0[NFCTAG] control bit should be set.

To configure for wake-up from power-down or deep power-down the PMC_DPDWKSRC[NTAG_FD] should be set.

In deep power-down, it is also necessary to set the PMC_CTRL[NTAGWAKUPRESETENABLE] control bit if the device is not configured to wake on an IO event. Alternatively, the device may be enabled to wake from an IO event, PMC_CTRL[WAKUPRESETENABLE] = 1, in which case, the GPIO cells will also have option of causing the wake-up and the PMC_CTRL[NTAGWAKUPRESETENABLE] does not need to be set.

Wake-up event due to FD event will result in PMC_WAKEIOCAUSE[NTAG_FD] status bit set.

42.6.4 Field detect interrupt

An edge on the field detect signal can cause a CPU interrupt, the NFC Tag interrupt. The interrupt is only created by a rising edge of the internal FD signal. Therefore to be sensitive to a falling edge of FD from the NTAG device, it is necessary to invert to signal using the ASYNC_SYSCON_NFCTAGPADSCTRL[INT_INVERT] control bit.

To determine the current status of the FD signal, it is necessary to use the state of the ASYNC_SYSCON_NFCTAGPADSCTRL[INT_INVERT] control bit to see which mode transition causes an interrupt and hence the status of the FD signal can be inferred. Alternatively, the NTAG device itself shows the field detect information in the RF_FIELD_PRESENT status bit, see NXP NT3H2111/NT3H2211 Data Sheet for more information.

43.1 Arm Cortex-M4 Details

Arm Limited publishes the document “Cortex™-M4 Devices Generic User Guide”, which is available on their website at:

- For the online manual, go to “infocenter.arm.com”, then search for “cortex-m4 user guide”. This will bring up links to chapters of the user guide.
- There are links at the bottom of user guide chapters to download a PDF file of the user guide.

This section of this manual describes the Cortex-M4 implementation options and other distinctions that apply for the JN5189(T)/JN5188(T) devices.

43.1.1 Cortex-M4 implementation options

The Cortex™-M4 CPU provides a number of implementation options. These are given below for the JN5189(T)/JN5188(T).

- The MPU is included for the Cortex-M4. The MPU provides fine grain memory control, enabling applications to implement security privilege levels, separating code, data and stack on a task-by-task basis.
- 56 interrupt slots are implemented for the Cortex-M4.
- 3 interrupt priority bits are implemented on the Cortex-M4, providing 8 priority levels.
- Sleep mode power-saving: NXP microcontrollers extend the number of reduced power modes beyond what is directly supported by the Cortex-M4. Details of reduced power modes and wake-up possibilities can be found in Chapter 11.
- Reset of the Cortex-M4 resets the CPU register bank.
- Memory features: The memory map for JN5189(T)/JN5188(T) devices is shown in Section 2.1.2.

In addition, there are debug and trace options, see Chapter 29.

44.1 Abbreviations

Table 108. Abbreviations

Acronym	Description
ADC	Analog-to-Digital Converter
AHB	Advanced High-performance Bus
AMBA	Advanced Microcontroller Bus Architecture
APB	Advanced Peripheral Bus
API	Application Programming Interface
BOD	BrownOut Detection
Boot	At power-up or chip reset, the process of starting the device in a controlled manner and executing the start-up code from internal Flash.
CRC	Cyclic Redundancy Check
DMA	Direct Memory Access
FIFO	First-In-First-Out
FMC	Flash Memory Controller
FRO	Internal Free-Running Oscillator, tuned to the factory specified frequency.
GPIO	General Purpose Input/Output
I2C or IIC	Inter-Integrated Circuit bus
I2S	Inter-IC Sound or Integrated Interchip Sound. A serial audio data communication method.
ISP	In-System Programming. These are methods of programming any on-chip flash on a blank or previously programmed device.
ISR	Interrupt Service Routine
JTAG	Joint Test Action Group
MPU	Memory Protection Unit
NVIC	Nested Vectored Interrupt Controller
PDM	Pulse Density Modulation. This is the data format used by the digital microphone inputs.
PLL	Phase-Locked Loop
PMU	Power Management Unit
POR	Power-On Reset
PWM	Pulse Width Modulator
RAM	Random Access Memory
RTC	Real Time Clock
SPI	Serial Peripheral Interface
SPIFI	SPI Flash Interface
SRAM	Static Random Access Memory
SWD	Serial-Wire Debug
TAP	Test Access Port
USART	Universal Synchronous/Asynchronous Receiver/Transmitter

44.2 References

[1] **Cortex-M4 TRM —** ARM Cortex-M4 Processor Technical Reference Manual

[2] **UM10204 —** I^2C-bus specification and user manual

[3] **NIST/FIPS spec —** Secure Hash Standard (SHS), which can be found at https://csrc.nist.gov/

44.3 Tables

44.4 Figures

44.5 Contents

Chapter 1: Introductory Information

Chapter 2: Memory Map

Chapter 3: Pin and Pad Descriptions

Chapter 4: Analog Power Management Unit

Chapter 5: Power Management

Chapter 6: Clock Distribution

Chapter 7: Reset, Boot and Wakeup

Chapter 8: Power Management Controller and SLEEPCON

Chapter 9: Nested Vectored Interrupt Controller (NVIC)

Chapter 10: System Configuration (SYSCON)

Chapter 11: Power Control API

Chapter 12: I/O Pin Configuration (IOCON)

Chapter 13: Input Multiplexing (INPUTMUX)

Chapter 14: General Purpose I/O (GPIO)

Chapter 15: Pin Interrupt and Pattern Match (PINT)

Chapter 16: Group GPIO Input Interrupt (GINT)

Chapter 17: Direct Memory Access (DMA)

Chapter 18: Pulse Width Modulation (PWM)

Chapter 19: Standard Counter/Timers (CT32B)

Chapter 20: Windowed Watchdog Timer (WWDT)

Chapter 21: Real-Time Clock (RTC)

Chapter 22: CPU System Tick Timer (SYSTICK)

Chapter 23: Universal Synchronous/Asynchronous Receiver/Transmitter (USART)

Chapter 24: Serial Peripheral Interfaces (SPI)

Chapter 25: Inter-Integrated Circuit (I^2C)

Chapter 26: Digital Microphone Interface (DMIC)

Chapter 27: 12-bit ADC Controller (ADC)

Chapter 28: Temperature Sensor

Chapter 29: Serial Wire Debug (SWD)

Chapter 30: Flash Controller

Chapter 31: SRAM Controller

Chapter 32: ROM Controller

Chapter 33: Hash-Crypt Peripheral for SHA1, SHA2 (HASH)

Chapter 34: SPI Flash Interface (SPIFI)

Chapter 35: True Random Number Generator (TRNG)

Chapter 36: ISO 7816 Smart Card Interface

Chapter 37: Async System Configuration (ASYNC_SYSCON)

Chapter 38: In-System Programming (ISP)

Chapter 39: Advanced Encryption Standard (AES)

Chapter 40: Infra-Red Modulator (CIC_IRB)

Chapter 41: 2.4 GHz Wireless Transceiver

Chapter 42: NFC Tag (NTAG)

Chapter 43: Arm Cortex-M4 Appendix

Chapter 44: Supplementary information

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number: UM11138
Rev 1.4
06/2020

NXP Semiconductors
Reference Manual

Document identifier: UM11138
Rev. 1.4, 06/2020

JN5189(T)/JN5188(T) Register Manual

Contents

Chapter 1
Power Management Controller (PMC)

1.1 PMC register descriptions

1.1.1 PMC memory map

PMC base address: 4001_2000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Power Management Control Register (CTRL)	32	RW	See description
20h	Flash Core LDO Control Register (LDOFLASHCORE)	32	RW	0000_0BC6h
30h	VBAT Brown Out Dectector Control Register (BODVBAT)	32	RW	See description
40h	High Speed FRO Control Register (FRO192M)	32	RW	See description
44h	1 MHz Free Running Oscillator Control Register (FRO1M)	32	RW	See description
50h	Analog Comparator and Analog Mux Control Register (ANAMUXCOMP)	32	RW	See description
64h	Power-Down and Deep Power-Down Wake-up Source Register (DPDWKSRC)	32	RW	See description
6Ch	FRO and XTAL Status Register (STATUSCLK)	32	RO	See description
70h	Reset Cause Register (RESETCAUSE)	32	RW	See description
80h	General Purpose always on Domain Data Storage Register 0 (AOREG0)	32	RW	0000_0000h
88h	General Purpose always on Domain Data Storage Register 2 (AOREG2)	32	RW	0000_0000h
9Ch	GPIO POR or Pin Reset Status Capture Register (PIOPORCAP)	32	RO	See description
A0h	GPIO Other Reset Status Capture Register (PIORESCAP)	32	RO	See description
B0h	Low Power Mode Module Power Control Register (PDSLEEPCFG)	32	RW	See description
B8h	Analog Blocks Power Control Register (PDRUNCFG)	32	RW	See description

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
BCh	Wake-up Source Register (WAKEIOCAUSE)	32	RO	See description
CCh	Extension of CTRL Register (CTRLNORST)	32	RW	See description

1.1.2 Power Management Control Register (CTRL)

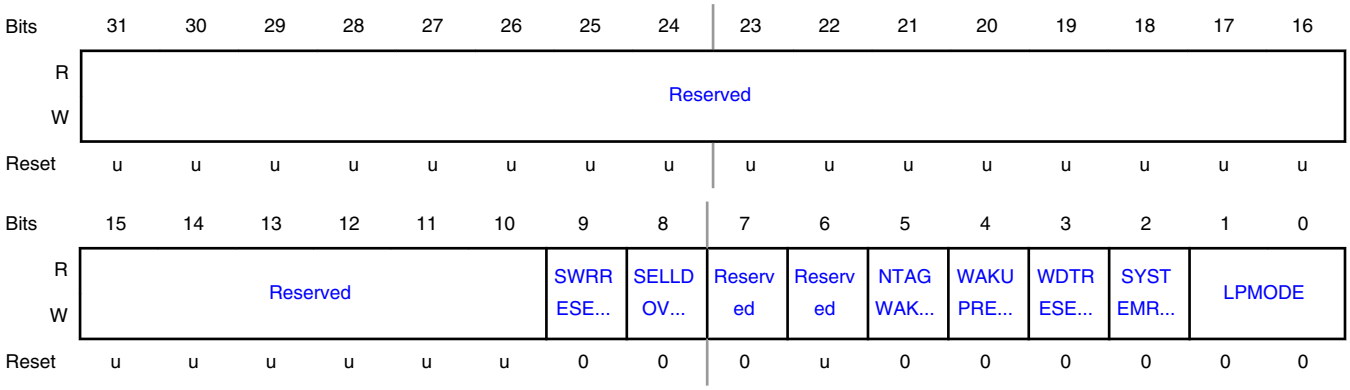
Offset

Register	Offset
CTRL	0h

Function

This register can enable some reset sources and also sets the power mode of the device. This register is reset by POR, RSTN, and WWDT.

Diagram



Fields

Field	Function
31-10 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
9 SWRRESETENABLE	Software Reset Enable If set, it enables the software reset to affect the system.

Table continues on the next page...

Table continued from the previous page...

Field	Function
8 SELLDOVOLTAGE	LDOs Current Output Levels This field is managed by the low power APIs.
7 —	RESERVED Do not modify the value in this field.
6 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
5 NTAGWAKUPRESETENABLE	Wake-up NTAG Reset Enable When set, the device can wake from deep power-down by edge on NTAG FD signal, even if I/O power domain is off (see WAKUPRESETENABLE). **NOTE** If I/O power domain is ON, wake-up by NTAG FD is enabled by default, the configuration of this bit is ignored. Do not set unless the device entering Deep Power-Down.
4 WAKUPRESETENABLE	Wake-up I/Os Reset Enable When set, the I/O power domain is not shutoff in deep power-down mode.
3 WDTRESETENABLE	Watchdog Timer Reset Enable If set, it allows a watchdog timer reset event to affect the system.
2 SYSTEMRESETENABLE	Arm System Reset Request Enable If set, it enables the Arm system reset to affect the system.
1-0 LPMODE	Power Mode Control 00b - Active. 01b - Deep Sleep. 10b - Power-Down. 11b - Deep Power-Down.

1.1.3 Flash Core LDO Control Register (LDOFLASHCORE)

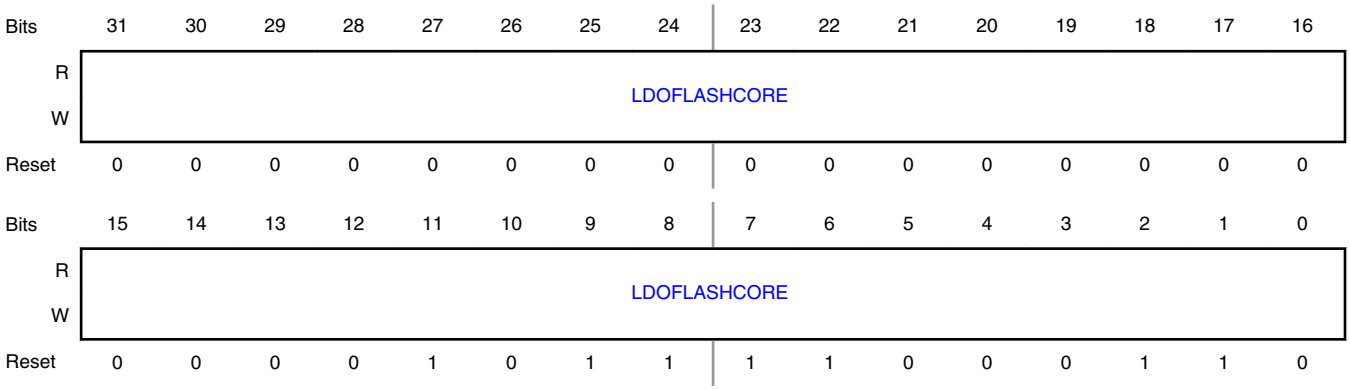
Offset

Register	Offset
LDOFLASHCORE	20h

Function

This register is controlled by the boot code and the Low power API software. it can be reset by all reset sources, except Arm system reset.

Diagram



Fields

Field	Function
31-0 LDOFLASHCORE	LDOFLASHCORE Do not modify this field.

1.1.4 VBAT Brown Out Dectector Control Register (BODVBAT)

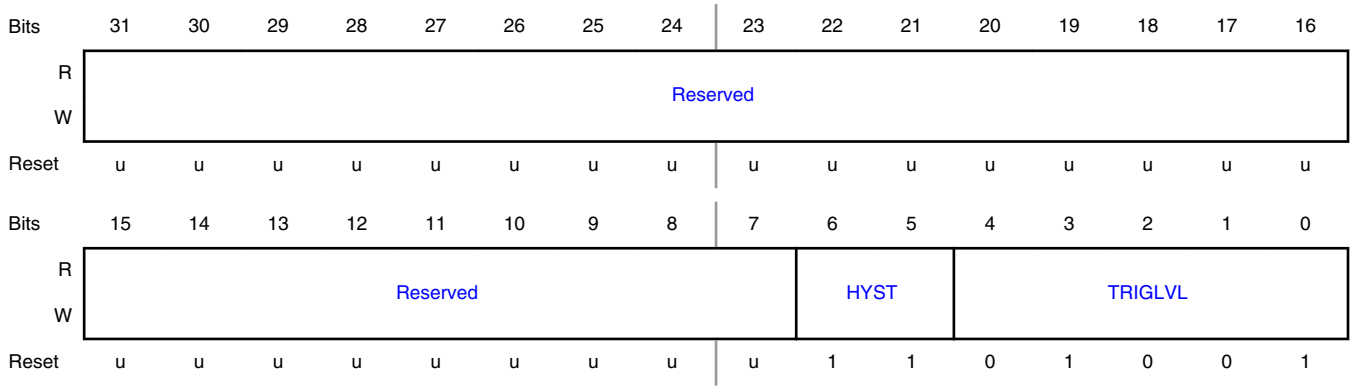
Offset

Register	Offset
BODVBAT	30h

Function

This reigster configures the brown out detector. The contents of this register are controlled by the boot code and the low-power API software and it is reset by POR, RSTN, WWDT.

Diagram



Fields

Field	Function
31-7 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
6-5 HYST	BOD Hysteresis Control Configure the hysteresis. 00b - 25 mV. 01b - 50 mV. 10b - 75 mV. 11b - 100 mV.
4-0 TRIGLVL	BOD Trigger Level Configure the BOD trigger threshold. 01001b 1.75 V. 01010b-11001b 1.8 V to 3.3 V, with 0.1 V increment per setting. Others Not valid.

1.1.5 High Speed FRO Control Register (FRO192M)

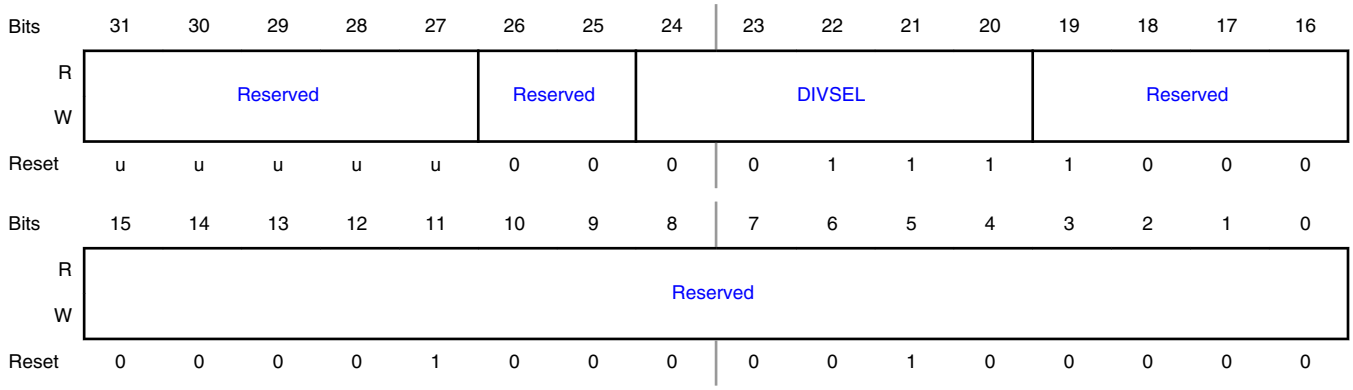
Offset

Register	Offset
FRO192M	40h

Function

This reigster enables output of the high speed FRO. It is controlled by the boot code and the Low-power API software and it is reset by POR, RSTN, WWDT.

Diagram



Fields

Field	Function
31-27 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
26-25 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
24-20 DIVSEL	Mode of Operation (clock to output). Each bit enables a clock as shown. Enables are additive, meaning that two or more clocks can be enabled together. xxxx1: 12 MHz enabled xxx1x: 32 MHz enabled xx1xx: 48 MHz enabled x1xxx: Not applicable 1xxxx: Not applicable
19-0 —	RESERVED User software should not modify the values in these fields.

1.1.6 1 MHz Free Running Oscillator Control Register (FRO1M)

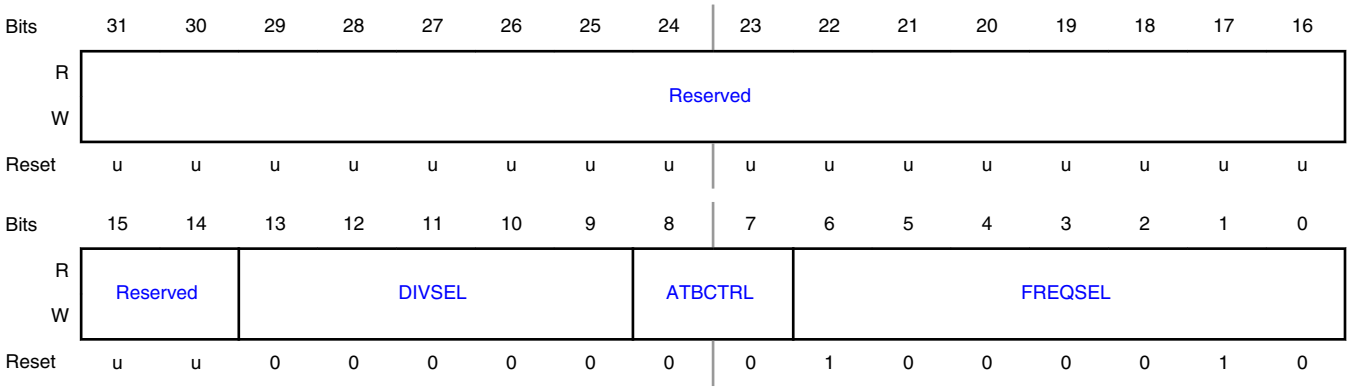
Offset

Register	Offset
FRO1M	44h

Function

Reset by all reset sources, except Arm system reset.

Diagram



Fields

Field	Function
31-14 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
13-9 DIVSEL	Divider Selection Do not modify this field.
8-7 ATBCTRL	Debug Control to Set the Analog/Digital Test Modes This field is only required for test purposes.
6-0 FREQSEL	Frequency Trimming This field is used to give accurate frequency for each device. The required setting is based upon calibration data sotred in flash during device test. This setting is applied by the clock driver function.

1.1.7 Analog Comparator and Analog Mux Control Register (ANAMUXCOMP)

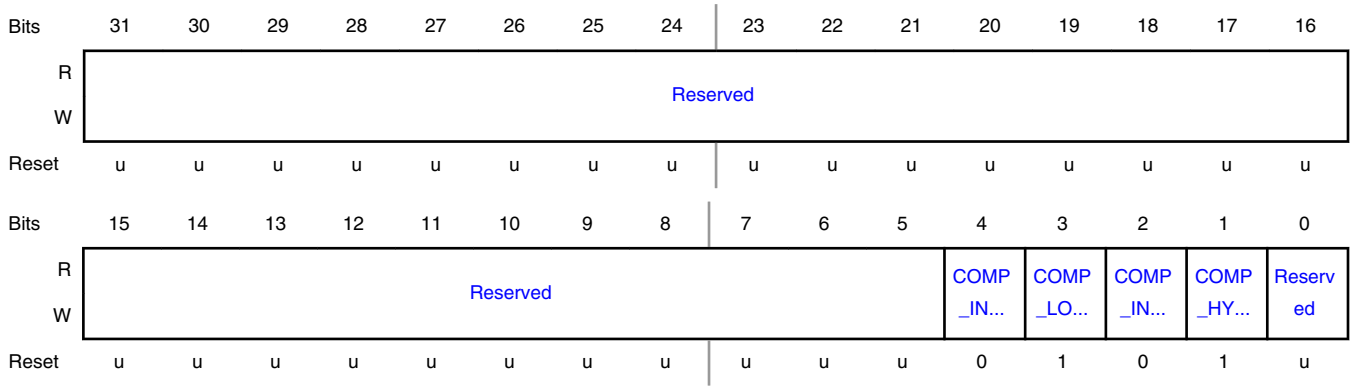
Offset

Register	Offset
ANAMUXCOMP	50h

Function

This register configures the analog compator and the input selection to the comparator. It is reset by all reset sources, except Arm system reset.

Diagram



Fields

Field	Function
31-5 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
4 COMP_INPUTSWAP	Input Swap Enable 0b - Normal configuration occurs, {_p, _n} is connected to {ACP, ACM}. 1b - Input swap is enabled, comparator{ _p, _n} ports are connected to {ACM, ACP}.
3 COMP_LOWPOWER	Comparator Low Power Mode Enable 0b - Not Enabled. 1b - Enabled.
2 COMP_INNINT	V_n Comparator Input Voltage reference inn_int input is selected for _n comparator input when sel_inn_int = 1 . This setting also requires PMU_BIASING to be active. Also flash biasing and DCDC converter needs to be enabled. If this setting = '0' then _n input comes from device pins, based on COMP_INPUTSWAP setting.
1 COMP_HYST	Hysteris Enable 0b - No hysteresis. 1b - Hysteris enabled.
0 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

1.1.8 Power-Down and Deep Power-Down Wake-up Source Register (DPDWKSRC)

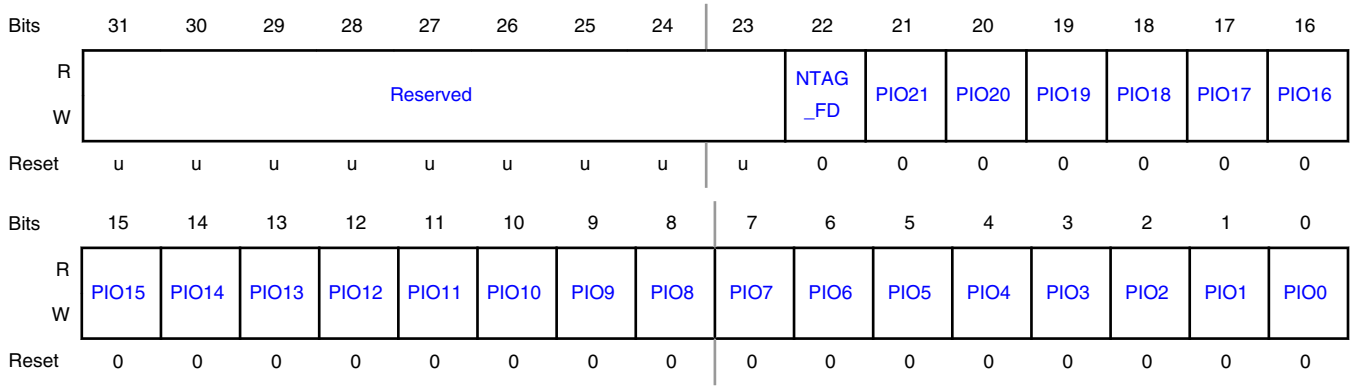
Offset

Register	Offset
DPDWKSRC	64h

Function

This register controls which IOs can cause a wake-up from power-down and deep power-down. Also for devices with an NTAG device, the wake-up control for the field detect line can also be enabled. It is reset by POR, RSTN, WWDT.

Diagram



Fields

Field	Function
31-23 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
22 NTAG_FD	Wakeup by NTAG_FD 0b - Disable wakeup from power-down and deep power-down modes by NTAG_FD. 1b - Enable wakeup from power-down and deep power-down modes by NTAG_FD.
21-0 PIOn	Wakeup by PIOn 0b - Disable wakeup from power-down and deep power-down modes by PIOn. 1b - Enable wakeup from power-down and deep power-down modes by PIOn.

1.1.9 FRO and XTAL Status Register (STATUSCLK)

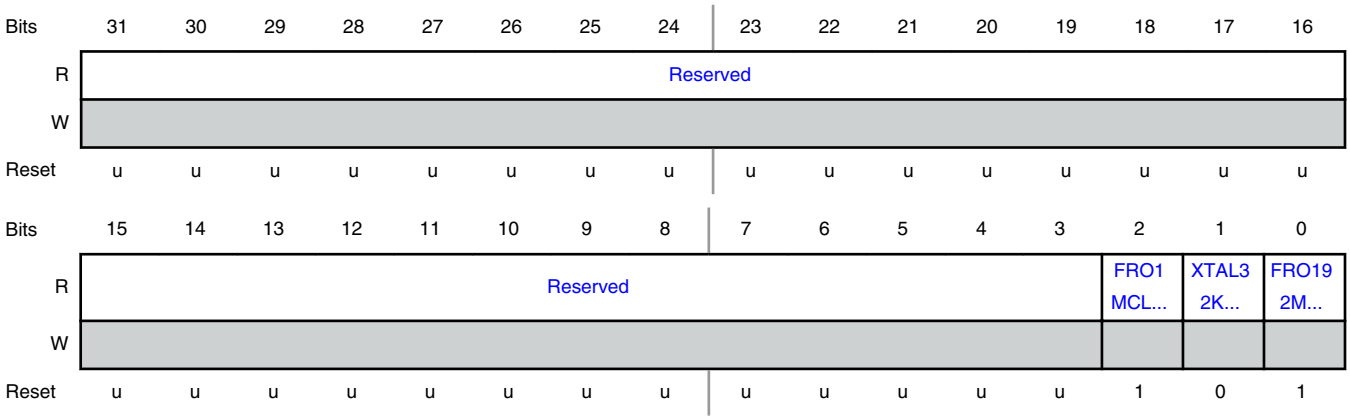
Offset

Register	Offset
STATUSCLK	6Ch

Function

This register shows the valid status flags of the FRO1M, 32K XTAL and high speed FRO. Reset by all reset sources, except Arm system reset.

Diagram



Fields

Field	Function
31-3 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
2 FRO1MCLKVALID	FRO 1 MHz Clock Valid
1 XTAL32KOK	XTAL Oscillator 32 kHz OK Signal When the XTAL is stable, a transition from 1 to 0 indicates a clock issue. Cannot be used to identify a stable clock during XTAL start.
0 FRO192MCLKVALID	High Speed FRO (FRO 192 MHz) Clock Valid Signal The FRO192M clock generator also generates the FRO12M, FRO32M and FRO48M clock signals. These sre valid when this flag is assertetd.

1.1.10 Reset Cause Register (RESETCAUSE)

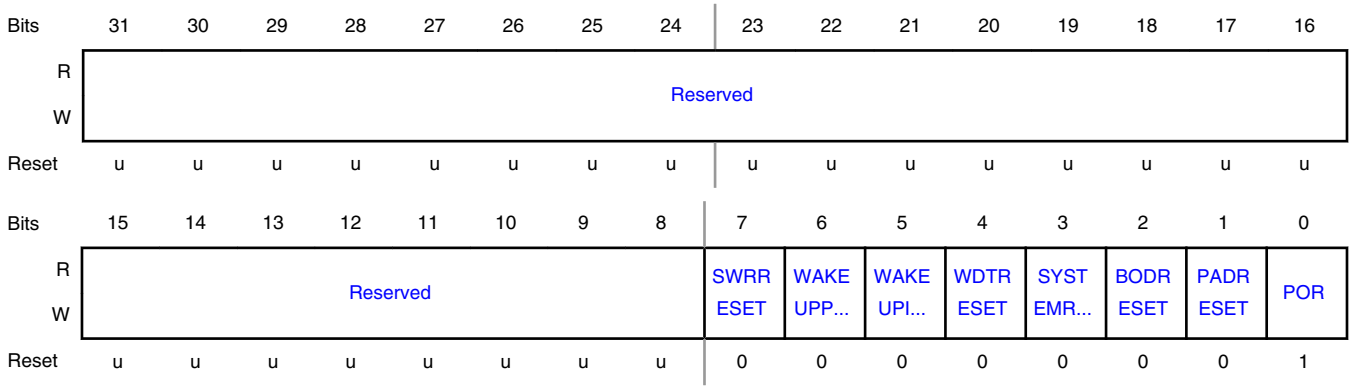
Offset

Register	Offset
RESETCAUSE	70h

Function

This register indicates which reset source was the cause of the last reset. This register is reset by POR.

Diagram



Fields

Field	Function
31-8 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
7 SWRRESET	Software Reset Read '1', the last chip reset was caused by a Software. Write '1' to clear this bit.
6 WAKEUPPWDNRESET	Power down Reset Read '1', the last CPU reset was caused by a Wake-up from Power down (many sources possible: timer, IO,etc). Write '1' to clear this bit. Check NVIC register if the device is not waken-up by IO (NVIC_GetPendingIRQ).
5 WAKEUPIORESET	Wake-up I/O Reset Read '1', the last chip reset was caused by a Wake-up I/O (GPIO or internal NTAG FD INT). Write '1' to clear this bit.
4 WDTRESET	Watchdog Timer Reset Read '1', the last chip reset was caused by the Watchdog Timer. Write '1' to clear this bit.

Table continues on the next page...

Table continued from the previous page...

Field	Function
3 SYSTEMRESET	Arm CPU Reset Read '1', the last chip reset was caused by a System Reset requested by the Arm CPU. Write '1' to clear this bit.
2 BODRESET	Brown Out Detector Reset Read '1', the last chip reset was caused by a Brown Out Detector. Write '1' to clear this bit. Software should write to 0.
1 PADRESET	Pad Reset Read '1', the last chip reset was caused by a Pad Reset. Write '1' to clear this bit.
0 POR	Power On Reset Read '1', the last chip reset was caused by a Power On Reset. Write '1' to clear this bit.

1.1.11 General Purpose always on Domain Data Storage Register 0 (AOREG0)

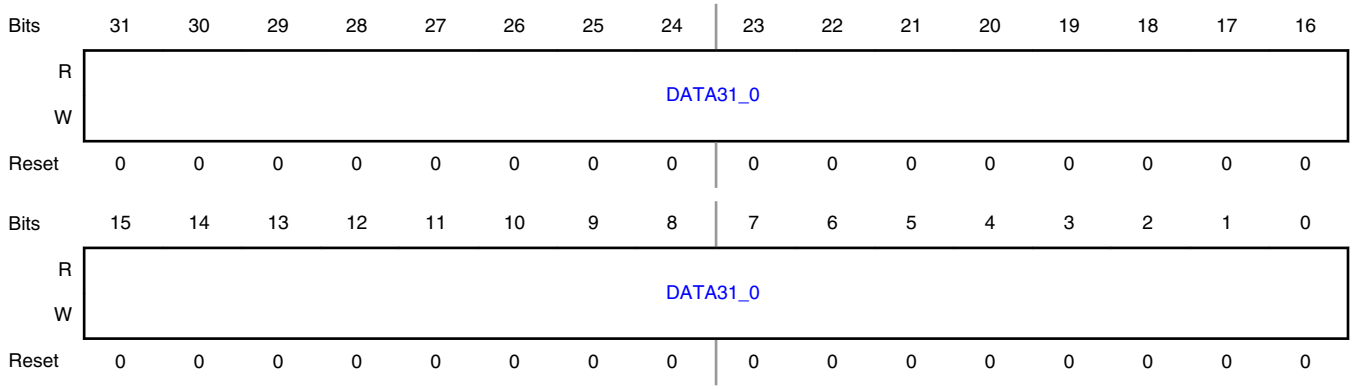
Offset

Register	Offset
AOREG0	80h

Function

This register can hold a 32-bit value that will be maintained through power-down cycles. It is a write once register and so could be used for a configurable value that must not be changed after it is set. Its value is reset by all reset sources, except Arm system reset.

Diagram



Fields

Field	Function
31-0 DATA31_0	General Purpose always on Domain Data Storage Only writable 1 time after any chip reset. After the 1st write, any further writes are blocked. After any chip reset, the write block is disabled until after next write. The chip reset includes POR, RSTN, WWDT reset, software reset and WAKEUP IO reset.

1.1.12 General Purpose always on Domain Data Storage Register 2 (AOREG2)

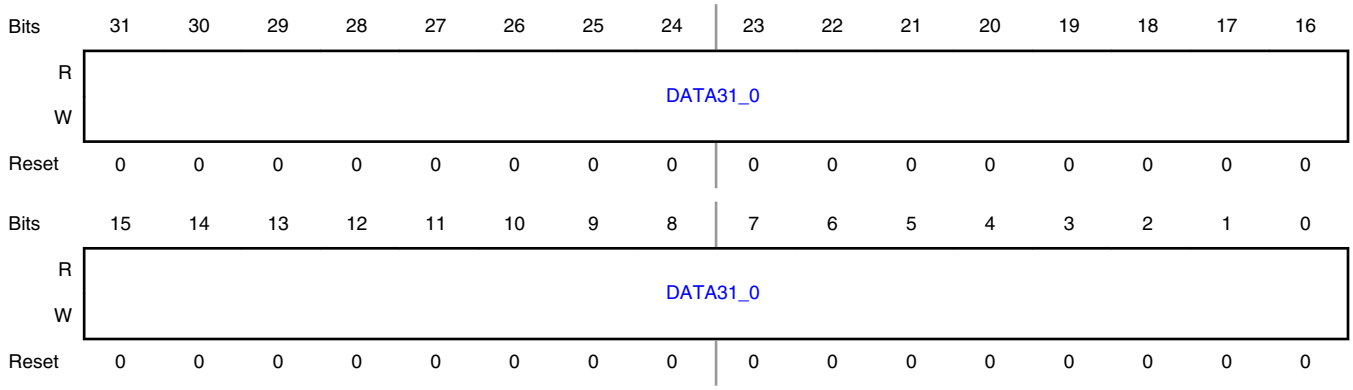
Offset

Register	Offset
AOREG2	88h

Function

The contents of this register can be held through power-down cycles, and modified when active. This register is reset by POR, RSTN.

Diagram



Fields

Field	Function
31-0 DATA31_0	General Purpose always on Domain Data Storage Only reinitialized on Power-On Reset and RSTN Pin reset.

1.1.13 GPIO POR or Pin Reset Status Capture Register (PIOPORCAP)

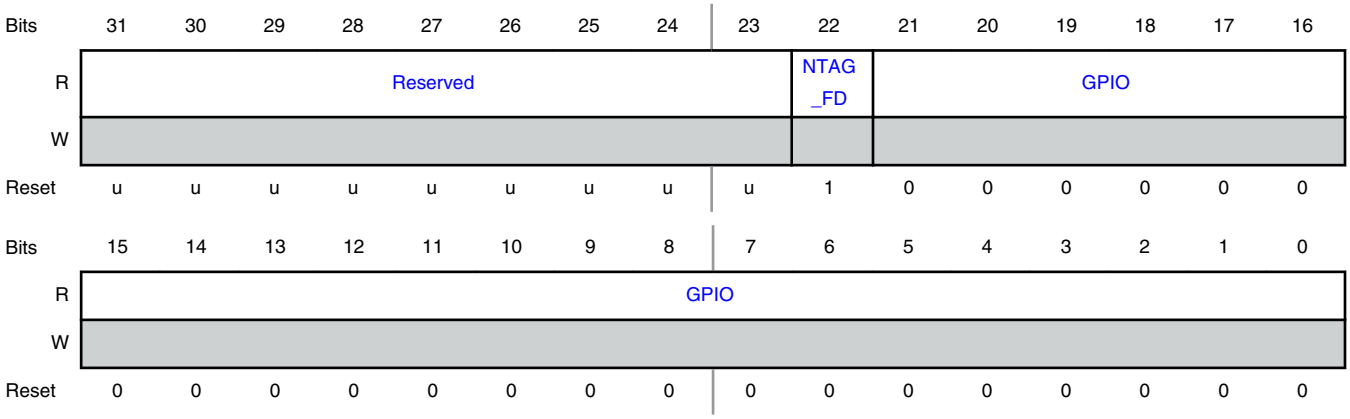
Offset

Register	Offset
PIOPORCAP	9Ch

Function

This register captures the state of the PIO at power-on-reset or pin reset. Each bit represents the power-on reset state of one PIO pin. Additionally, for devices which have the NTAG device fitted, it will store the state of the field detect line as well. This register is reset by POR, RSTN.

Diagram



Fields

Field	Function
31-23 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
22 NTAG_FD	Capture of NTAG_FD Value at Power-on-reset and Pin Reset Only valid on devices fitted with internal NFC Tag.
21-0 GPIO	Capture of PIO Values at Power-on-reset and Pin Reset

1.1.14 GPIO Other Reset Status Capture Register (PIORESCAP)

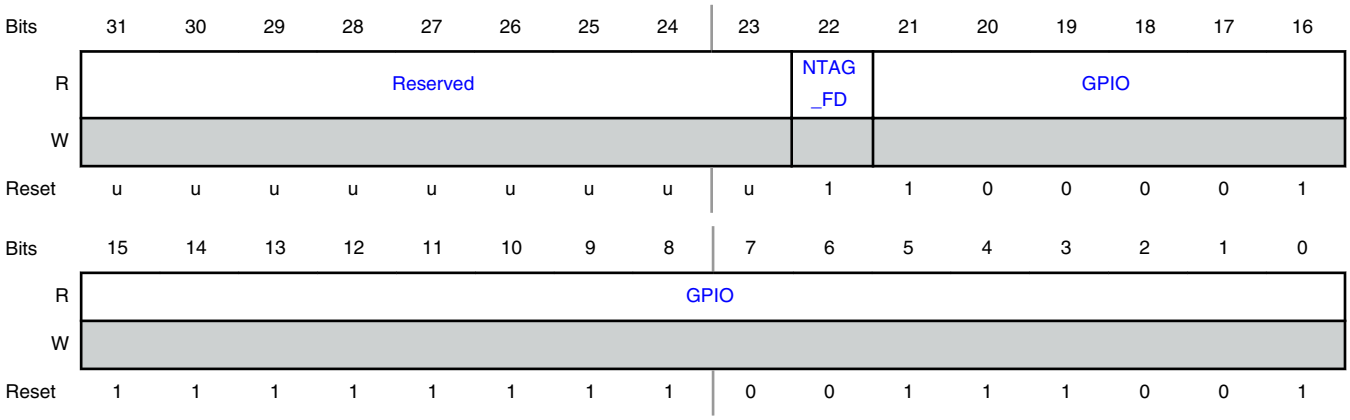
Offset

Register	Offset
PIORESCAP	A0h

Function

This register captures the state of the PIO when a reset other than a power-on reset or pin reset occurs. Each bit represents the reset state of one PIO pin. Additionally, for devices which have the NTAG device fitted, it will store the state of the field detect line as well. This register is reset by WWDT, BOD, WAKEUP IO, Arm system reset.

Diagram



Fields

Field	Function
31-23 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
22 NTAG_FD	Capture of NTAG_FD Value Only valid on devices fitted with internal NFC Tag.
21-0 GPIO	Capture of GPIO Values

1.1.15 Low Power Mode Module Power Control Register (PDSLEEPCFG)

Offset

Register	Offset
PDSLEEPCFG	B0h

Function

This register is managed by the low power APIs; it controls the power to various modules in Low-Power modes. It is reset by all reset sources, except Arm system reset. The contents of this register are managed by the SDK functions and should not need to be accessed directly from the Application.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved										PDEN_PD...	PDEN_PD...	PDEN_PD...	PDEN_PD...	PDEN_PD...	PDEN_PD...
W																
Reset	u	u	u	u	u	u	u	u	u	u	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PDEN_PD...	PDEN_PD...	PDEN_PD...	PDEN_PD...	PDEN_PD...	PDEN_PD...	Reserved	EN_PDMC...	PDEN_PD...	PDEN_PD...	PDEN_FR...	PDEN_FR...	PDEN_VB...	PDEN_LD...	PDEN_BI...	PDEN_DC...
W																
Reset	0	0	0	0	0	0	u	0	1	1	0	0	0	0	0	0

Fields

Field	Function
31-22 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
21-10 PDEN_PD_MEMn	Enable Power Down mode of SRAM n This field enables Power Down mode of SRAM n when entering in Powerdown mode. Automatically switched off in deep power down.
9 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
8 EN_PDMCU_RETENTION	Enable MCU Power Domain State Retention This field enables MCU Power Domain state retention when entering in 'Powerdown' mode for modem and radio cal values

Table continues on the next page...

Table continued from the previous page...

Field	Function
7 PDEN_PD_COMM0	Enable Comm0 Power Domain (USART0, I2C0, SPI0) Power Down Mode This field enables Comm0 power domain (USART0, I2C0, SPI0) Power Down mode when entering in Powerdown mode. In Deep power down it is disabled by hardware. In deep sleep it is always enabled.
6 PDEN_PD_FLASH	Enable Flash Power Domain Power Down mode (power shutoff) This field enables Flash power domain Power Down mode (power shutoff) when entering in DeepSleep. In PowerDown modes this domain is automatically powered off.
5 PDEN_FRO1M	FRO1M Power Control This field controls FRO1M power in Deep Sleep, Power down and Deep Power down modes. This should be disabled before entering power down (unless needed for low power timers) or deep power down mode. 0b - FRO1M is disabled. 1b - FRO1M is enabled.
4 PDEN_FRO192M	FRO192M Power Control This field controls FRO192M power in Deep Sleep, Power down and Deep Power down modes. This should be disabled before entering power down or deep power down mode. 0b - FRO192M is disabled. 1b - FRO192M is enabled.
3 PDEN_VBAT_BOD	VBAT BOD Power Control This field controls VBAT BOD power in Power down and Deep Power down modes. 0b - VBAT BOD is disabled in Power down and Deep Power down modes. 1b - VBAT BOD is enabled in Power down and Deep Power down modes.
2 PDEN_LDO_MEM	LDO Memories Power Control This field controls LDO memories power in Power down mode. Automatically switched off in deep power down. 0b - LDO is disabled in Power down mode. 1b - LDO is enabled in Power down mode.
1 PDEN_BIAS	Bias Power Control Controls Bias power in Power down and Deep Power down modes. 0b - Bias is disabled in Power down and Deep Power down modes. 1b - Bias is enabled in Power down and Deep Power down modes.

Table continues on the next page...

Table continued from the previous page...

Field	Function
0 PDEN_DCDC	DCDC Power Control This field ontrols DCDC power in Power down and Deep Power down modes. Automatically switched off in deep power down. 0b - DCDC is disabled in Power down and Deep Power down modes 1b - DCDC is enabled in Power down and Deep Power down modes.

1.1.16 Analog Blocks Power Control Register (PDRUNCFG)

Offset

Register	Offset
PDRUNCFG	B8h

Function

This register controls the power to various analog blocks. Reset by all reset sources, except Arm System Reset. The contents of this register are managed by the SDK functions and should not need to be accessed directly from the Application.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved				ENA_ANA...	ENA_XTA...	ENA_FRO...	Reserved		ENA_LDO...	Reserved					
Reset	u	u	u	u	0	0	0	0	0	0	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31-28 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
27 ENA_ANA_COMP	Analog Comparator Enabled
26 ENA_XTAL32K	XTAL32K Enabled
25 ENA_FRO32K	FRO32K Enabled
24-23 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
22 ENA_LDO_ADC	LDO ADC Enabled
21-0 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

1.1.17 Wake-up Source Register (WAKEIOCAUSE)

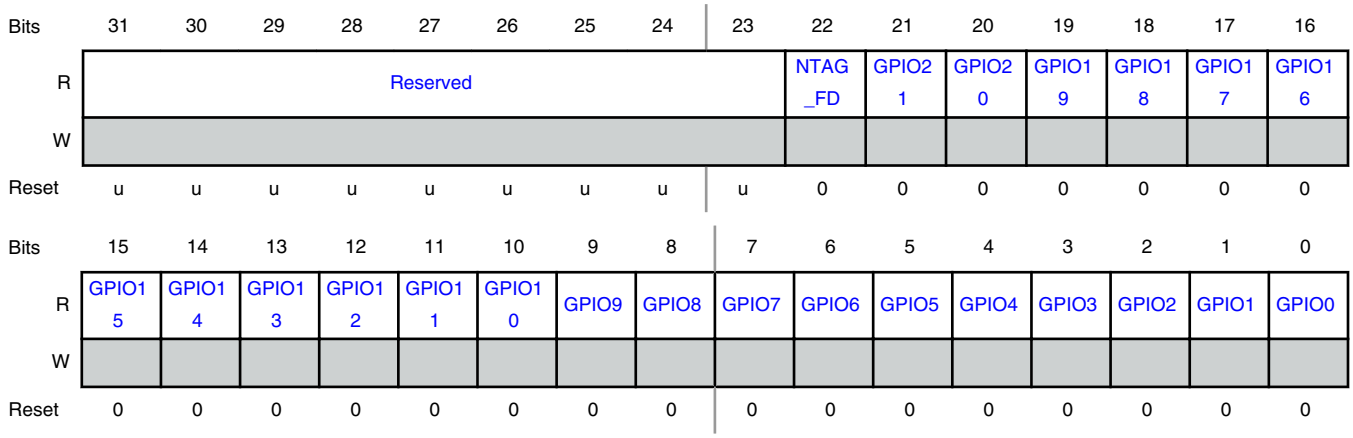
Offset

Register	Offset
WAKEIOCAUSE	BCh

Function

This register stores the wake-up source from Power-Down and Deep Power-Down modes. It allows the identification of the Wake-up source from Power-Down mode or Deep Power-Down mode. Additionally, for devices which have the NTAG device fitted, it will identify if the field detect line caused the wake-up. It is reset by POR, RSTN, WWDT.

Diagram



Fields

Field	Function
31-23 —	RESERVED Reserved. The value read from a reserved bit is not defined.
22 NTAG_FD	Wake-up Triggered by NTAG FD Only valid on devices fitted with internal NFC Tag.
21-0 GPIOn	Wake-up Triggered by PIO n

1.1.18 Extension of CTRL Register (CTRLNORST)

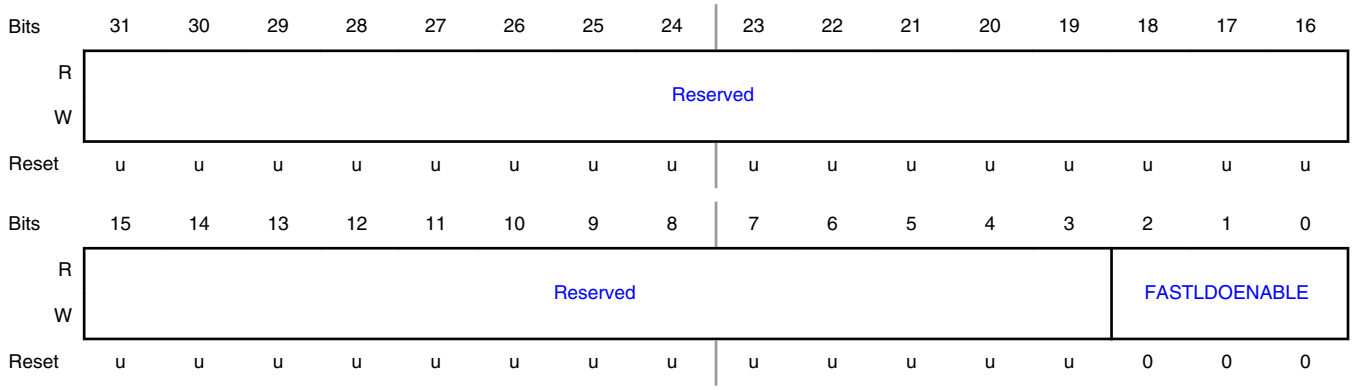
Offset

Register	Offset
CTRLNORST	CCh

Function

This register will never be reset except by POR.

Diagram



Fields

Field	Function
31-3 —	RESERVED Reserved. The value read from a reserved bit is not defined.
2-0 FASTLDOENABLE	Fast LDO Wake-up Enable 3 bits for the different wake-up sources:generic async wake up event as selected by SLEEPCON/ STARTER0/1, IO wake-up event, RSTN pad event. If required, this field should only be managed by the Low-power driver software.

Chapter 2
System Configuration (SYSCON)

2.1 SYSCON register descriptions

2.1.1 SYSCON memory map

SYSCON base address: 4000_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Memory Remap Control Register (MEMORYREMAP)	32	RW	See description
10h	AHB Matrix Priority Control Register (AHBMATPRIO)	32	RW	0000_0000h
40h	System Tick Counter Calibration Register (SYSTCKCAL)	32	RW	See description
48h	NMI Source Select Register (NMISRC)	32	RW	See description
4Ch	Asynchronous APB Control Register (ASYNCAPBCTRL)	32	RW	See description
100h	Peripheral Reset Control Register 0 (PRESETCTRL0)	32	RW	See description
104h	Peripheral Reset Control Register 1 (PRESETCTRL1)	32	RW	See description
120h	PRESETCTRL0 Bits Set Register (PRESETCTRLSET0)	32	WO	See description
124h	PRESETCTRL1 Bits Set Register (PRESETCTRLSET1)	32	WO	See description
140h	PRESETCTRL0 Bits Clear Register (PRESETCTRLCLR0)	32	WO	See description
144h	PRESETCTRL1 Bits Clear Register (PRESETCTRLCLR1)	32	WO	See description
200h	AHB Clock Control Register 0 (AHBCLKCTRL0)	32	RW	See description
204h	AHB Clock control Register 1 (AHBCLKCTRL1)	32	RW	See description
220h	AHBCLKCTRL0 Bits Set Register (AHBCLKCTRLSET0)	32	WO	See description
224h	AHBCLKCTRL1 Bits Set Register (AHBCLKCTRLSET1)	32	WO	See description

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
240h	AHBCLKCTRL0 Bits Clear Register (AHBCLKCTRLCLR0)	32	WO	See description
244h	AHBCLKCTRL1 Bits Clear Register (AHBCLKCTRLCLR1)	32	WO	See description
280h	Main Clock Source Selection Register (MAINCLKSEL)	32	RW	See description
284h	OSC32KCLK and OSC32MCLK Clock Sources Selection Register (OSC32CLKSEL)	32	RW	See description
288h	CLKOUT Clock Source Selection Register (CLKOUTSEL)	32	RW	See description
2A0h	SPIFI Clock Source Selection Register (SPIFICLKSEL)	32	RW	See description
2A4h	ADC Clock Source Selection Register (ADCCLKSEL)	32	RW	See description
2B0h	USART0 and USART1 Clock Source Selection Register (USAR TCLKSEL)	32	RW	See description
2B4h	I2C0, I2C1 and I2C2 Clock Source Selection Register (I2CCLKSEL)	32	RW	See description
2B8h	SPI0 and SPI1 Clock Source Selection Register (SPICLKSEL)	32	RW	See description
2BCh	Infra Red Clock Source Selection Register (IRCLKSEL)	32	RW	See description
2C0h	PWM Clock Source Selection Register (PWMCLKSEL)	32	RW	See description
2C4h	Watchdog Timer Clock Source Selection Register (WDTCLKSEL)	32	RW	See description
2CCh	Modem Clock Source Selection Register (MODEMCLKSEL)	32	RW	See description
2E8h	Fractional Rate Generator (FRG) Clock Source Selection Register (FRGCLKSEL)	32	RW	See description
2ECh	Digital microphone (DMIC) Subsystem Clock Source Selection Register (DMICCLKSEL)	32	RW	See description
2F0h	Wake-up Timer Clock Source Selection Register (WKTCLKSEL)	32	RW	See description
300h	SYSTICK Clock Divider Register (SYSTICKCLKDIV)	32	RW	See description

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
304h	TRACE Clock Divider Register (TRACECLKDIV)	32	RW	See description
36Ch	Watchdog Timer Clock Divider Register (WDTCLKDIV)	32	RW	See description
378h	Infra Red Clock Divider Register (IRCLKDIV)	32	RW	See description
380h	System Clock Divider Register (AHBCLKDIV)	32	RW	See description
384h	CLKOUT Clock Divider Register (CLKOUTDIV)	32	RW	See description
390h	SPIFI Clock Divider Register (SPIFICLKDIV)	32	RW	See description
394h	ADC Clock Divider Register (ADCCLKDIV)	32	RW	See description
398h	Real Time Clock Divider (1 kHz clock generation) Register (RTCC LKDIV)	32	RW	See description
3A0h	Fractional Rate Generator Divider Register (FRGCTRL)	32	RW	See description
3A8h	DMIC Clock Divider Register (DMICCLKDIV)	32	RW	See description
3ACh	Real Time Clock Divider (1 Hz clock generation) Register (RTC1 HZCLKDIV)	32	RW	See description
3FCh	Clock Configuration Registers Access Register (CLOCKGENUPDA TELOCKOUT)	32	RW	See description
59Ch	Random Number Generator Clocks Control Register (RNGCLKCT RL)	32	RW	See description
5A0h	All SRAMs Common Control Register (SRAMCTRL)	32	RW	See description
5CCh	32K Clock Enable Register (MODEMCTRL)	32	RW	See description
5D4h	XTAL 32 kHz Oscillator Capacitor Control Register (XTAL32KCAP)	32	RW	See description
5D8h	32 MHz XTAL Control Register (XTAL32MCTRL)	32	RW	See description
680h	Start Logic 0 Wake-up Enable Register (STARTER0)	32	RW	See description

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
684h	Start Logic 1 Wake-up Enable Register (STARTER1)	32	RW	See description
6A0h	Set STARTER0 Register (STARTERSET0)	32	WO	See description
6A4h	Set STARTER1 Register (STARTERSET1)	32	WO	See description
6C0h	Clear STARTER0 Bit Register (STARTERCLR0)	32	WO	See description
6C4h	Clear STARTER1 Bits Register (STARTERCLR1)	32	WO	See description
708h	I/O Retention Control Register (RETENTIONCTRL)	32	RW	See description
808h	CPSTACK (CPSTACK)	32	RW	0000_0000h
A00h	Analog Interrupt Control Rgister (ANACTRL_CTRL)	32	RW	See description
A04h	Analog Modules Outputs Current Values Register (ANACTRL_VAL)	32	RO	See description
A08h	Analog Status Register (ANACTRL_STAT)	32	RW	See description
A0Ch	Analog Modules Interrupt Enable Read and Set Register (ANAC TRL_INTENSET)	32	RW	See description
A10h	Analog Modules Interrupt Enable Clear Register (ANACTRL_INTE NCLR)	32	WO	See description
A14h	Analog Modules Interrupt Status Register (ANACTRL_INTSTAT)	32	RO	See description
A18h	Various System Clock Control Register (CLOCK_CTRL)	32	RW	See description
A20h	Wake-up Timers Control Register (WKT_CTRL)	32	RW	See description
A24h	Wake-up Timer 0 Reload Value Least Significant Bits Resister (WKT_LOAD_WKT0_LSB)	32	RW	0000_0000h
A28h	Wake-up Timer 0 Reload Value Most Significant Bits Register (WKT_ LOAD_WKT0_MSB)	32	RW	See description
A2Ch	Wake-up Timer 1 Reload Value Register (WKT_LOAD_WKT1)	32	RW	See description
A30h	Wake-up Timer 0 Current Value Least Significant Bits Register (WKT_VAL_WKT0_LSB)	32	RO	FFFF_FFFFh

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
A34h	Wake-up Timer 0 Current Value Most Significant Bits Reister (WKT_VAL_WKT0_MSB)	32	RO	See description
A38h	Wake-up Timer 1 Current Value Register (WKT_VAL_WKT1)	32	RO	See description
A3Ch	Wake-up Timers Status Register (WKT_STAT)	32	W1C	See description
A40h	Interrupt Enable Read and Set Register (WKT_INTENSET)	32	RW	See description
A44h	Interrupt Enable Clear Register (WKT_INTENCLR)	32	WO	See description
A48h	Interrupt Status Register (WKT_INTSTAT)	32	RO	See description
E08h	GPIO_INT Synchonization First Stage Bypass Register (GPIOPSYNC)	32	RW	See description
FB0h	Chip Revision ID and Number Register (DIEID)	32	RO	See description
FF0h	Test Access Security Code Register (CODESECURITYPROT)	32	WO	See description

2.1.2 Memory Remap Control Register (MEMORYREMAP)

Offset

Register	Offset
MEMORYREMAP	0h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved				QSPI_REMAP_APP_3		QSPI_REMAP_APP_2		QSPI_REMAP_APP_1		QSPI_REMAP_APP_0		Reserved	Reserved		
Reset	u	u	u	u	1	1	1	0	0	1	0	0	u	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved				Reserved	Reserved									MAP	
Reset	0	0	0	0	u	0	0	0	0	0	0	0	0	0	0	0

Fields

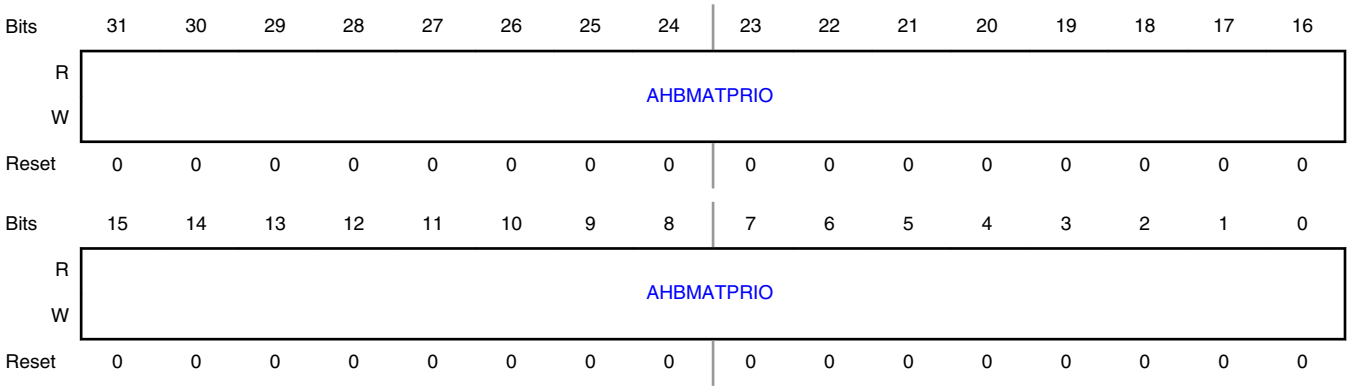
Field	Function
31-28 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
27-26 QSPI_REMAP_APP_3	QSPI Remap Address 3 Address bits to use when QSPI Flash address [19:18] = 3 (256-KB unit page). Setting 11 gives no remapping. Do not modify this field.
25-24 QSPI_REMAP_APP_2	QSPI Remap Address 2 Address bits to use when QSPI Flash address [19:18] = 2 (256-KB unit page). Setting 10 gives no remapping. Do not modify this field.
23-22 QSPI_REMAP_APP_1	QSPI Remap Address 1 Address bits to use when QSPI Flash address [19:18] = 1 (256-KB unit page). Setting 01 gives no remapping. Do not modify this field.
21-20 QSPI_REMAP_APP_0	QSPI Remap Address 0 Address bits to use when QSPI Flash address [19:18] = 0 (256-KB unit page). Setting 00 gives no remapping. Do not modify this field.
19 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
18-12 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
11 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
10-2 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
1-0 MAP	Select Vector Table Location Select the location of the vector table. 00b - Vector Table in ROM. 01b - Vector Table in RAM. 10b - Vector Table in Flash. 11b - Vector Table in Flash.

2.1.3 AHB Matrix Priority Control Register (AHBMATPRIO)

Offset

Register	Offset
AHBMATPRIO	10h

Diagram



Fields

Field	Function
31-0 AHBMATPRIO	AHB Matrix Priority The higher a setting, the higher the priority.

2.1.4 System Tick Counter Calibration Register (SYSTCKCAL)

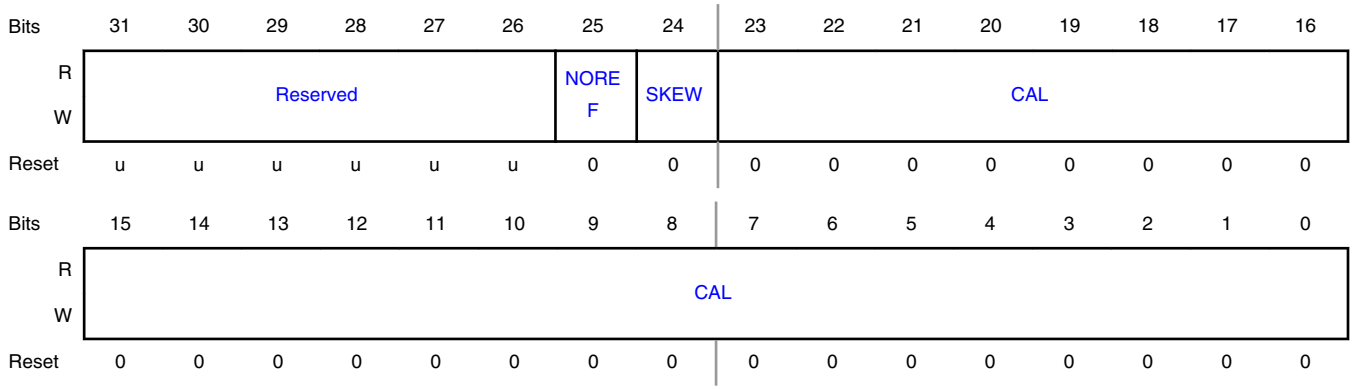
Offset

Register	Offset
SYSTCKCAL	40h

Function

This register configures the Sys tick function within the Cortex processor.

Diagram



Fields

Field	Function
31-26 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
25 NOREF	Cortex System Tick Timer SYST_CALIB[NOREF] Setting 0b - A separate reference clock is available. 1b - A separate reference clock is not available.
24 SKEW	Cortex System Tick Timer SYST_CALIB[SKEW] Setting 0b - The value of SYST_CALIB[TENMS] field is considered to be precise. 1b - The value of SYST_CALIB[TENMS] is not considered to be precise.
23-0 CAL	Cortex System Tick Timer Calibration Value It is readable from Cortex SYST_CALIB[TENMS] register field. Set this value to be the number of clock periods to give 10 ms period. SYSTICK frequency is a function of the mainclk and SYSTICKCLKDIV register. If the tick timer is configured to use the System clock directly, this value must reflect the 10 ms tick count for that clock.

2.1.5 NMI Source Select Register (NMISRC)

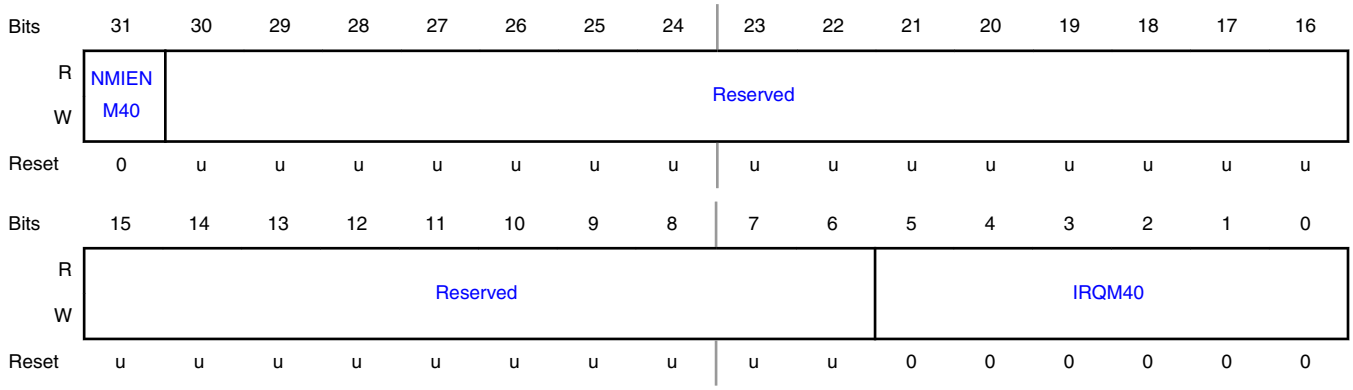
Offset

Register	Offset
NMISRC	48h

Function

The Non-maskable interrupt (NMI) can be enabled in this register. The NMI can be driven by any other interrupt signal that is part of the interrupts to the NVIC. The source selection is configured in this register.

Diagram



Fields

Field	Function
31 NMIENM40	Enable NMI by IRQM40 Write a 1 to this bit enables the Non-Maskable Interrupt (NMI) source selected by IRQM40. The NMI Interrupt should be disabled before changing the source selection (IRQM40)
30-6 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
5-0 IRQM40	Interrupt Source Number of NMI by NMIENM40 The number of the interrupt source within the interrupt array that acts as the Non-Maskable Interrupt (NMI) for the Cortex-M4, if enabled by NMIENM40. This can also cause the device to wakeup from sleep.

2.1.6 Asynchronous APB Control Register (ASYNCAPBCTRL)

Offset

Register	Offset
ASYNCAPBCTRL	4Ch

Function

The Asynchronous APB contains the Ctimers and Asynchronous System Controller. Access to this region of the memory map is enabled in this register.

Diagram

Fields

Field	Function
31-1 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
0 ENABLE	Enable Asynchronous APB Bridge and Subsystem

2.1.7 Peripheral Reset Control Register 0 (PRESETCTRL0)

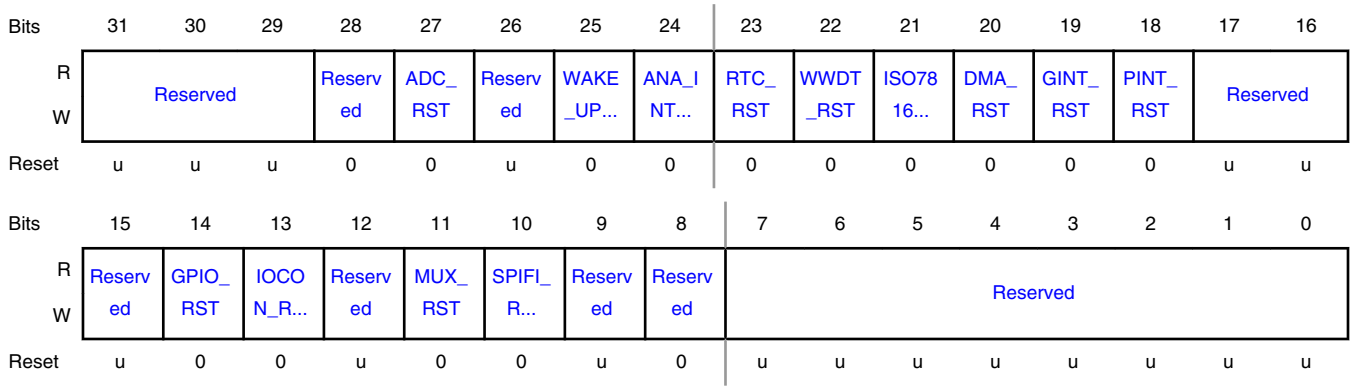
Offset

Register	Offset
PRESETCTRL0	100h

Function

The peripherals can be individually reset using the PRESETCTRL0 and PRESETCTRL1 registers.

Diagram



Fields

Field	Function
31-29 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
28 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
27 ADC_RST	ADC Controller Reset Control 0b - Clear reset to the ADC controller. 1b - Assert reset to the ADC controller.
26 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
25 WAKE_UP_TIMERS_RST	Wake up Timers Reset Control 0b - Clear reset to the SYSCON low-power wake-up timers. 1b - Assert reset to the SYSCON low-power wake-up timers
24 ANA_INT_CTRL_RST	Analog Modules Interrupt Controller Reset Control for BOD and Comparator interrupt status 0b - Clear reset to the Analog Modules Interrupt Controller. 1b - Assert reset to the Analog Modules Interrupt Controller.
23 RTC_RST	Real Time Clock (RTC) Reset Control 0b - Clear reset to the Real Time Clock module. 1b - Assert reset tothe Real Time Clock module.
22 WWDT_RST	Watchdog Timer Reset Control 0b - Clear reset to the Watchdog Timer module. 1b - Assert reset to the Watchdog Timer module.
21 ISO7816_RST	ISO7816 Smart Card Interface Module Reset Control 0b - Clear reset to the ISO7816 module. 1b - Assert reset to the ISO7816 module.
20 DMA_RST	DMA Controller Reset Control 0b - Clear reset to the DMA Controller. 1b - Assert reset to the DMA Controller.

Table continues on the next page...

Table continued from the previous page...

Field	Function
19 GINT_RST	Group Interrupt (GINT) Reset Control. 0b - Clear reset to the GINT module. 1b - Assert reset to the GINT module.
18 PINT_RST	Pin Interrupt (PINT) Reset Control. 0b - Clear reset to the PINT module. 1b - Assert reset to the PINT module.
17-15 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
14 GPIO_RST	GPIO Reset Control 0b - Clear reset to the GPIO module. 1b - Assert reset to the GPIO module.
13 IOCON_RST	I/O Controller Reset Control 0b - Clear reset to the I/O Controller Module. 1b - Assert reset to the I/O Controller Module.
12 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
11 MUX_RST	Input Mux (INPUTMUX) Reset Control 0b - Clear reset to the INPUTMUX module. 1b - Assert reset to the INPUTMUX module.
10 SPIFI_RST	Quad SPI Flash Controller (SPIFI) Reset Control 0b - Clear reset to the SPIFI module. 1b - Assert reset to the SPIFI module.
9 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
8 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
7-0 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

2.1.8 Peripheral Reset Control Register 1 (PRESETCTRL1)

Offset

Register	Offset
PRESETCTRL1	104h

Function

The peripherals can be individually reset using the PRESETCTRL0 and PRESETCTRL1 registers.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved				HASH_RST	DMIC_RST	RFP_RST	AES_RST	MODEM_M...	Reserved	ZIGBEE_...	I2C2_RST	RNG_RST	PWM_RST	IR_RST	SPI1_RST
Reset	u	u	u	u	0	0	0	0	0	u	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SPI0_RST	I2C1_RST	I2C0_RST	USART1_...	USART0_...	Reserved										
Reset	0	0	0	0	0	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31-28 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
27 HASH_RST	HASH Reset Control 0b - Clear reset to the HASH module. 1b - Assert reset to the HASH module.

Table continues on the next page...

Table continued from the previous page...

Field	Function
26 DMIC_RST	DMIC Reset Control 0b - Clear reset to the DMIC module. 1b - Assert reset to the DMIC module.
25 RFP_RST	RFP (Radio controller) Reset Control 0b - Clear reset to the RFP Radio Controller. 1b - Assert reset to the RFP Radio Controller.
24 AES_RST	AES256 Reset Control 0b - Clear reset to the AES256 module. 1b - Assert reset to the AES256 module.
23 MODEM_MASTER_RST	MODEM AHB Master Interface Reset Control 0b - Clear reset to the Modem AHB Master Interface. 1b - Assert reset to the Modem AHB Master Interface.
22 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
21 ZIGBEE_RST	Zigbee/Thread Reset Control 0b - Clear reset to the Zigbee/Thread MAC and Modem. 1b - Assert reset to the Zigbee/Thread MAC and Modem.
20 I2C2_RST	I2C2 Reset Control 0b - Clear reset to the I2C2 module. 1b - Assert reset to the I2C2 module.
19 RNG_RST	Random Number Generator (RNG) Reset Control 0b - Clear reset to the RNG module. 1b - Assert reset to the RNG module.
18 PWM_RST	PWM Reset Control 0b - Clear reset to the PWM module. 1b - Assert reset to the PWM module.
17 IR_RST	Infra Red Modulator Reset Control 0b - Clear reset to the Infra-Red Modulator module. 1b - Assert reset to the Infra-Red Modulator module.

Table continues on the next page...

Table continued from the previous page...

Field	Function
16 SPI1_RST	SPI1 Reset Control 0b - Clear reset to the SPI1 module. 1b - Assert reset to the SPI1 module.
15 SPI0_RST	SPI0 Reset Control 0b - Clear reset to the SPI0 module. 1b - Assert reset to the SPI0 module.
14 I2C1_RST	I2C1 Reset Control 0b - Clear reset to the I2C1 module. 1b - Assert reset to the I2C1 module.
13 I2C0_RST	I2C0 Reset Control 0b - Clear reset to the I2C0 module. 1b - Assert reset to the I2C0 module.
12 USART1_RST	USART1 Reset Control 0b - Clear reset to the USART1 module. 1b - Assert reset to the USART1 module.
11 USART0_RST	USART0 Reset Control 0b - Clear reset to the USART0 module. 1b - Assert reset to the USART0 module.
10-0 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

2.1.9 PRESETCTRL0 Bits Set Register (PRESETCTRLSET0)

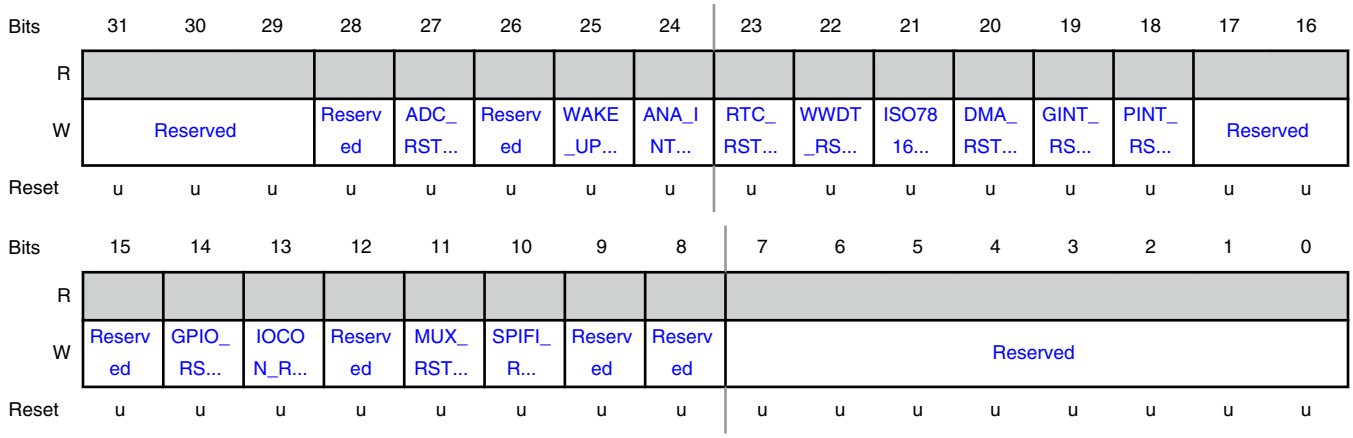
Offset

Register	Offset
PRESETCTRLSET0	120h

Function

Set bits in PRESETCTRL0. It is recommended to change the PRESETCTRL0 register by using the related PRESETCTRLSET and PRESETCTRLCLR registers.

Diagram



Fields

Field	Function
31-29 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
28 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
27 ADC_RST_SET	ADC_RST Set Writing one to this field sets the PRESETCTRL0[ADC_RST] bit.
26 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
25 WAKE_UP_TIMERS_RST_SET	WAKE_UP_TIMERS_RST Set Writing one to this field sets the PRESETCTRL0[WAKE_UP_TIMERS_RST] bit.
24 ANA_INT_CTRL_RST_SET	ANA_INT_CTRL_RST Set Writing one to this field sets the PRESETCTRL0[ANA_INT_CTRL_RST] bit
23 RTC_RST_SET	RTC_RST Set Writing one to this field sets the PRESETCTRL0[RTC_RST] bit
22 WWDT_RST_SET	WWDT_RST Set Writing one to this field sets the PRESETCTRL0[WWDT_RST] bit

Table continues on the next page...

Table continued from the previous page...

Field	Function
21 ISO7816_RST_SET	ISO7816_RST Set Writing one to this field sets the PRESETCTRL0[ISO7816_RST] bit
20 DMA_RST_SET	DMA_RST Set Writing one to this field sets the PRESETCTRL0[DMA_RST] bit
19 GINT_RST_SET	GINT_RST Set Writing one to this foe;d sets the PRESETCTRL0[GINT_RST] bit
18 PINT_RST_SET	PINT_RST Set Writing one to this field sets the PRESETCTRL0[PINT_RST] bit
17-15 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
14 GPIO_RST_SET	GPIO_RST Set Writing one to this field sets the PRESETCTRL0[GPIO_RST] bit
13 IOCON_RST_SET	IOCON_RST Set Writing one to this field sets the PRESETCTRL0[IOCON_RST] bit
12 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
11 MUX_RST_SET	MUX_RST Set Writing one to this field sets the PRESETCTRL0[MUX_RST] bit
10 SPIFI_RST_SET	SPIFI_RST Set Writing one to this field sets the PRESETCTRL0[SPIFI_RST] bit
9 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
8 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
7-0 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

2.1.10 PRESETCTRL1 Bits Set Register (PRESETCTRLSET1)

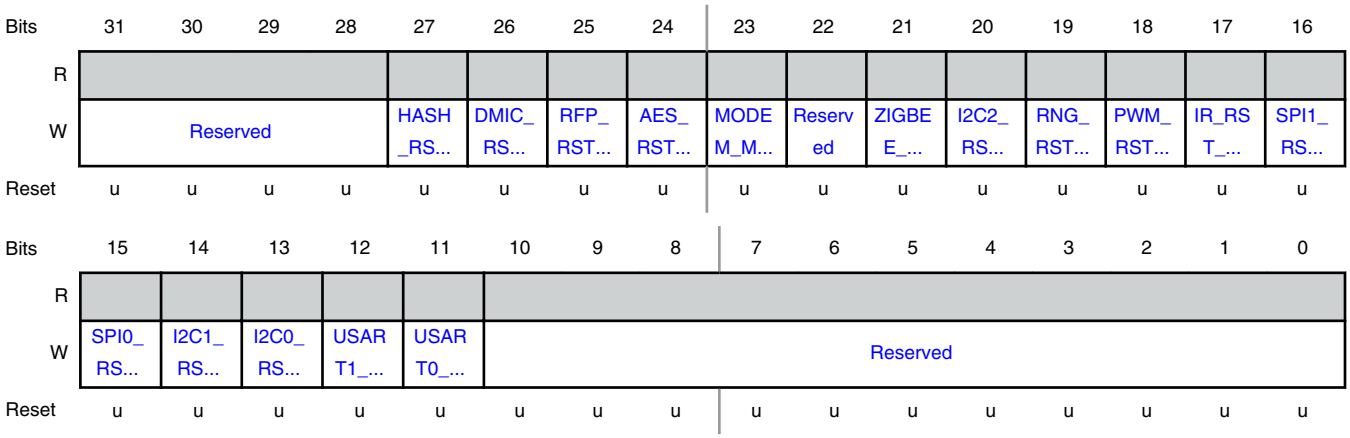
Offset

Register	Offset
PRESETCTRLSET1	124h

Function

Set bits in PRESETCTRL1. It is recommended that changes to PRESETCTRL1 registers be accomplished by using the related PRESETCTRLSET1 and PRESETCTRLCLR1 registers.

Diagram



Fields

Field	Function
31-28 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
27 HASH_RST_SET	HASH_RST Writing one to this field sets the PRESETCTRL1[HASH_RST] bit.
26 DMIC_RST_SET	DMIC_RST Writing one to this field sets the PRESETCTRL1[DMIC_RST] bit.
25 RFP_RST_SET	RFP_RST Set Writing one to this field sets the PRESETCTRL1[RFP_RST] bit.
24 AES_RST_SET	AES_RST Set Writing one to this field sets the PRESETCTRL1[AES_RST] bit.
23 MODEM_MASTER_RST_SET	MODEM_MASTER_RST Writing one to this field sets the PRESETCTRL1[MODEM_MASTER_RST] bit.
22 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
21 ZIGBEE_RST_SET	ZIGBEE_RST Set Writing one to this field sets the PRESETCTRL1[ZIGBEE_RST] bit.
20 I2C2_RST_SET	I2C2_RST Set Writing one to this field sets the I2C2_RST bit in the PRESETCTRL1 register
19 RNG_RST_SET	RNG_RST Set Writing one to this field sets the PRESETCTRL1[RNG_RST] bit.
18 PWM_RST_SET	PWM_RST Set Writing one to this field sets the PRESETCTRL1[PWM_RST] bit.
17 IR_RST_SET	IR_RST Set Writing one to this field sets the PRESETCTRL1[IR_RST] bit.
16 SPI1_RST_SET	SPI1_RST Set Writing one to this field sets the PRESETCTRL1[SPI1_RST] bit.

Table continues on the next page...

Table continued from the previous page...

Field	Function
15 SPI0_RST_SET	SPI0_RST Set Writing one to this field sets the PRESETCTRL1[SPI0_RST] bit.
14 I2C1_RST_SET	I2C1_RST Set Writing one to this field sets the PRESETCTRL1[I2C1_RST] bit.
13 I2C0_RST_SET	I2C0_RST Set Writing one to this field sets the PRESETCTRL1[I2C0_RST] bit.
12 USART1_RST_SET	USART1_RST Set Writing one to this field sets the PRESETCTRL1[USART1_RST] bit.
11 USART0_RST_SET	USART0_RST Set Writing one to this field sets the PRESETCTRL1[USART0_RST] bit.
10-0 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

2.1.11 PRESETCTRL0 Bits Clear Register (PRESETCTRLCLR0)

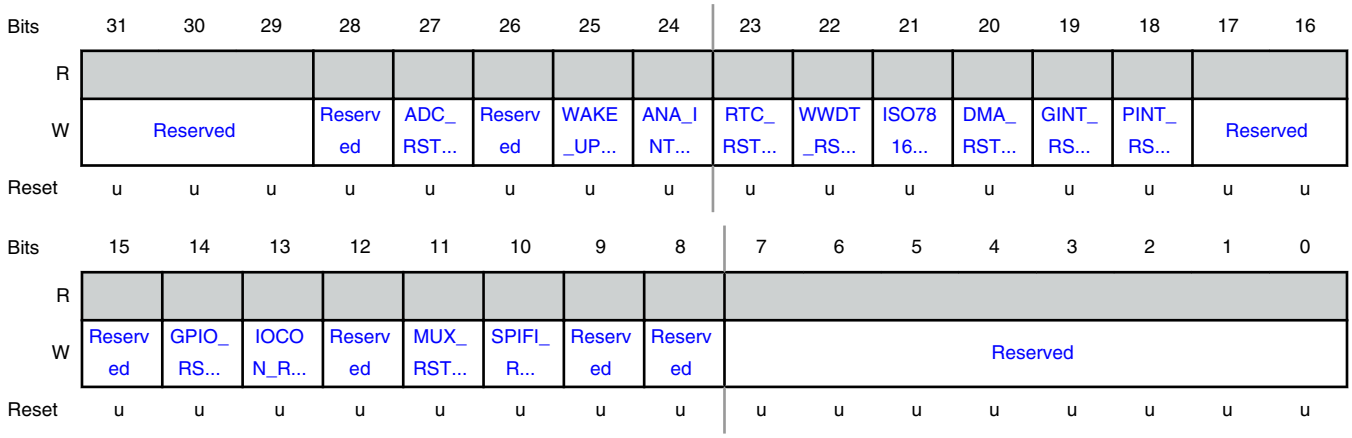
Offset

Register	Offset
PRESETCTRLCLR0	140h

Function

Clear bits in PRESETCTRL0. It is recommended that changes to PRESETCTRL0 registers be accomplished by using the related PRESETCTRLSET0 and PRESETCTRLCLR0 registers.

Diagram



Fields

Field	Function
31-29 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
28 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
27 ADC_RST_CLR	ADC_RST Clear Writing one to this field clears the PRESETCTRL0[ADC_RST] bit.
26 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
25 WAKE_UP_TIMERS_RST_CLR	WAKE_UP_TIMERS_RST Clear Writing one to this field clears the PRESETCTRL0[WAKE_UP_TIMERS_RST] bit.
24 ANA_INT_CTRL_RST_CLR	ANA_INT_CTRL_RST Clear Writing one to this field clears the PRESETCTRL0[ANA_INT_CTRL_RST] bit.
23 RTC_RST_CLR	RTC_RST Writing one to this field clears the PRESETCTRL0[RTC_RST] bit.
22 WWDT_RST_CLR	WWDT_RST Clear Writing one to this field clears the PRESETCTRL0[WWDT_RST] bit.

Table continues on the next page...

Table continued from the previous page...

Field	Function
21 ISO7816_RST_CLR	ISO7816_RST Clear Writing one to this field clears the PRESETCTRL0[ISO7816_RST] bit.
20 DMA_RST_CLR	DMA_RST Clear Writing one to this field clears the PRESETCTRL0[DMA_RST] bit.
19 GINT_RST_CLR	GINT_RST Clear Writing one to this field clears the PRESETCTRL0[GINT_RST] bit.
18 PINT_RST_CLR	PINT_RST Clear Writing one to this field clears the PRESETCTRL0[PINT_RST] bit.
17-15 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
14 GPIO_RST_CLR	GPIO_RST Clear Writing one to this field clears the PRESETCTRL0[GPIO_RST] bit.
13 IOCON_RST_CLR	IOCON_RST Clear Writing one to this field clears the PRESETCTRL0[IOCON_RST] bit.
12 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
11 MUX_RST_CLR	MUX_RST Clear Writing one to this field clears the PRESETCTRL0[MUX_RST] bit.
10 SPIFI_RST_CLR	SPIFI_RST Clear Writing one to this field clears the PRESETCTRL0[SPIFI_RST] bit.
9 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
8 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
7-0 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

2.1.12 PRESETCTRL1 Bits Clear Register (PRESETCTRLCLR1)

Offset

Register	Offset
PRESETCTRLCLR1	144h

Function

Clear bits in PRESETCTRL1. It is recommended that changes to PRESETCTRL1 registers be accomplished by using the related PRESETCTRLSET1 and PRESETCTRLCLR1 registers.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	Reserved				HASH_RS...	DMIC_RS...	RFP_RST...	AES_RST...	MODEM_M...	Reserved	ZIGBEE_...	I2C2_RS...	RNG_RST...	PWM_RST...	IR_RST_...	SPI1_RS...
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	SPI0_RS...	I2C1_RS...	I2C0_RS...	USART1_...	USART0_...	Reserved										
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31-28 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
27 HASH_RST_CLR	HASH_RST Clear Writing one to this register clears the PRESETCTRL1[HASH_RST] bit.
26 DMIC_RST_CLR	DMIC_RST Clear Writing one to this register clears the PRESETCTRL1[DMIC_RST] bit.
25 RFP_RST_CLR	RFP_RST Clear Writing one to this register clears the PRESETCTRL1[RFP_RST] bit.
24 AES_RST_CLR	AES_RST Clear Writing one to this register clears the PRESETCTRL1[AES_RST] bit.
23 MODEM_MASTER_RST_CLR	MODEM_MASTER_RST Clear Writing one to this register clears the PRESETCTRL1[MODEM_MASTER_RST] bit.
22 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
21 ZIGBEE_RST_CLR	ZIGBEE_RST Clear Writing one to this field clears the PRESETCTRL1[ZIGBEE_RST] bit.
20 I2C2_RST_CLR	I2C2_RST Clear Writing one to this field clears the PRESETCTRL1[I2C2_RST] bit.
19 RNG_RST_CLR	RNG_RST Clear Writing one to this field clears the PRESETCTRL1[RNG_RST] bit.
18 PWM_RST_CLR	PWM_RST Clear Writing one to this field clears the PRESETCTRL1[PWM_RST] bit.
17 IR_RST_CLR	IR_RST Clear Writing one to this field clears the PRESETCTRL1[IR_RST] bit.
16 SPI1_RST_CLR	SPI1_RST Clear Writing one to this field clears the PRESETCTRL1[SPI1_RST] bit.

Table continues on the next page...

Table continued from the previous page...

Field	Function
15 SPI0_RST_CLR	SPI0_RST Clear Writing one to this field clears the PRESETCTRL1[SPI0_RST] bit.
14 I2C1_RST_CLR	I2C1_RST Clear Writing one to this field clears the PRESETCTRL1[I2C1_RST] bit.
13 I2C0_RST_CLR	I2C0_RST Clear Writing one to this field clears the PRESETCTRL1[I2C0_RST] bit.
12 USART1_RST_CLR	USART1_RST Clear Writing one to this field clears the PRESETCTRL1[USART1_RST] bit.
11 USART0_RST_CLR	USART0_RST Clear Writing one to this field clears the PRESETCTRL1[USART0_RST] bit.
10-0 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

2.1.13 AHB Clock Control Register 0 (AHBCLKCTRL0)

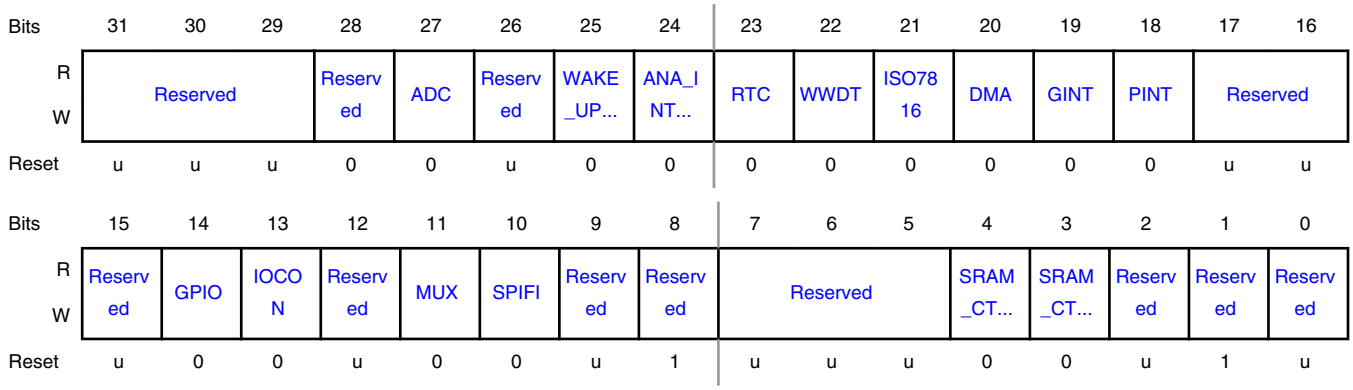
Offset

Register	Offset
AHBCLKCTRL0	200h

Function

Peripherals on the AHB bus have individual clocks which can be enabled and disabled individually in AHBCLKCTRL0 and AHBCLKCTRL1.

Diagram



Fields

Field	Function
31-29 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
28 —	RESERVED Do not modify the value in this field.
27 ADC	Enable the clock for the ADC controller 0b - Disable. 1b - Enable.
26 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
25 WAKE_UP_TIMERS	Enable the clock for the Wake-up Timers 0b - Disable. 1b - Enable.
24 ANA_INT_CTRL	Enable the clock for the Analog Interrupt Control module (for BOD and comparator status and interrupt control) 0b - Disable. 1b - Enable.
23 RTC	Enable the clock for the RTC 0b - Disable. 1b - Enable.

Table continues on the next page...

Table continued from the previous page...

Field	Function
22 WWDT	Enable the clock for the Watchdog Timer 0b - Disable. 1b - Enable.
21 ISO7816	Enable the clock for the ISO7816 smart card interface 0b - Disable. 1b - Enable.
20 DMA	Enable the clock for the DMA controller 0b - Disable. 1b - Enable.
19 GINT	Enable the clock for the Group interrupt block (GINT) 0b - Disable. 1b - Enable.
18 PINT	Enable the clock for the Pin interrupt block (PINT) 0b - Disable. 1b - Enable.
17-15 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
14 GPIO	Enable the clock for the GPIO 0b - Disable. 1b - Enable.
13 IOCON	Enable the clock for the I/O controller block 0b - Disable. 1b - Enable.
12 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
11 MUX	Enable the clock for the Input Mux 0b - Disable. 1b - Enable.

Table continues on the next page...

Table continued from the previous page...

Field	Function
10 SPIFI	Enable the clock for the Quad SPI Flash controller Note: SPIFI IOs need to be configured as high drive. 0b - Disable. 1b - Enable.
9 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
8 —	RESERVED Do not modify the value in this field.
7-5 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
4 SRAM_CTRL1	Enable the clock for the SRAM Controller 1 (SRAM 8 to SRAM 11) 0b - Disable. 1b - Enable.
3 SRAM_CTRL0	Enable the clock for the SRAM Controller 0 (SRAM 0 to SRAM 7) 0b - Disable. 1b - Enable.
2 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
1 —	RESERVED Do not modify the value in this field.
0 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

2.1.14 AHB Clock control Register 1 (AHBCLKCTRL1)

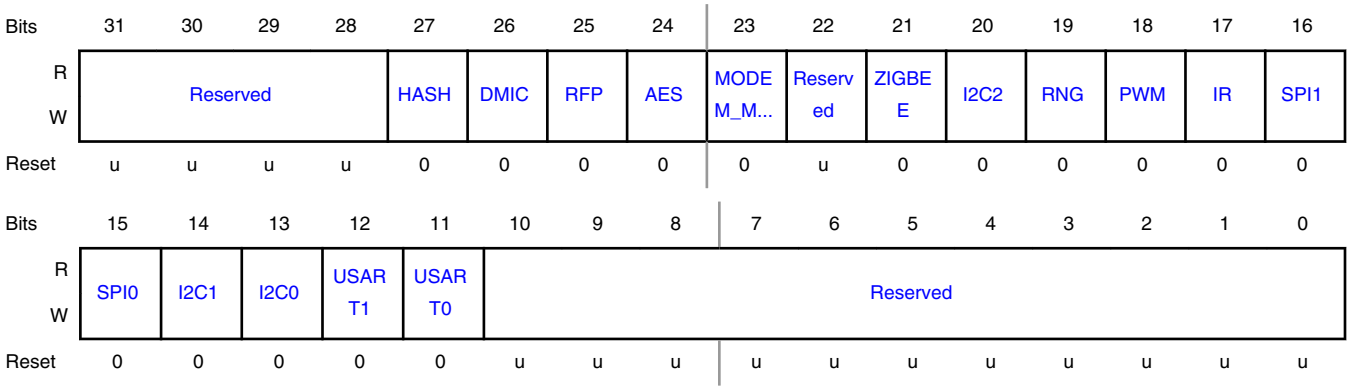
Offset

Register	Offset
AHBCLKCTRL1	204h

Function

Peripherals on the AHB bus have individual clocks which can be enabled or disabled individually in AHBCLKCTRL0 and AHBCLKCTRL1.

Diagram



Fields

Field	Function
31-28 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
27 HASH	Enable the clock for the Hash 0b - Disable. 1b - Enable.
26 DMIC	Enable the clock for the DMIC 0b - Disable. 1b - Enable.
25 RFP	Enable the clock for the RFP (Radio Front End controller) 0b - Disable. 1b - Enable.
24 AES	Enable the clock for the AES 0b - Disable. 1b - Enable.
23 MODEM_MASTER	Enable the clock for the Modem AHB Master Interface 0b - Disable. 1b - Enable.

Table continues on the next page...

Table continued from the previous page...

Field	Function
22 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
21 ZIGBEE	Enable the clock for the Zigbee/Thread Modem 0b - Disable. 1b - Enable.
20 I2C2	Enable the clock for the I2C2 0b - Disable. 1b - Enable.
19 RNG	Enable the clock for the Random Number Generator 0b - Disable. 1b - Enable.
18 PWM	Enable the clock for the PWM 0b - Disable. 1b - Enable.
17 IR	Enable the clock for the Infra Red 0b - Disable. 1b - Enable.
16 SPI1	Enable the clock for the SPI1 0b - Disable. 1b - Enable.
15 SPI0	Enable the clock for the SPI0 0b - Disable. 1b - Enable.
14 I2C1	Enable the clock for the I2C1 0b - Disable. 1b - Enable.
13 I2C0	Enable the clock for the I2C0 0b - Disable. 1b - Enable.

Table continues on the next page...

Table continued from the previous page...

Field	Function
12 USART1	Enable the clock for the USART1 0b - Disable. 1b - Enable.
11 USART0	Enable the clock for the USART0 0b - Disable. 1b - Enable.
10-0 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

2.1.15 AHBCLKCTRL0 Bits Set Register (AHBCLKCTRLSET0)

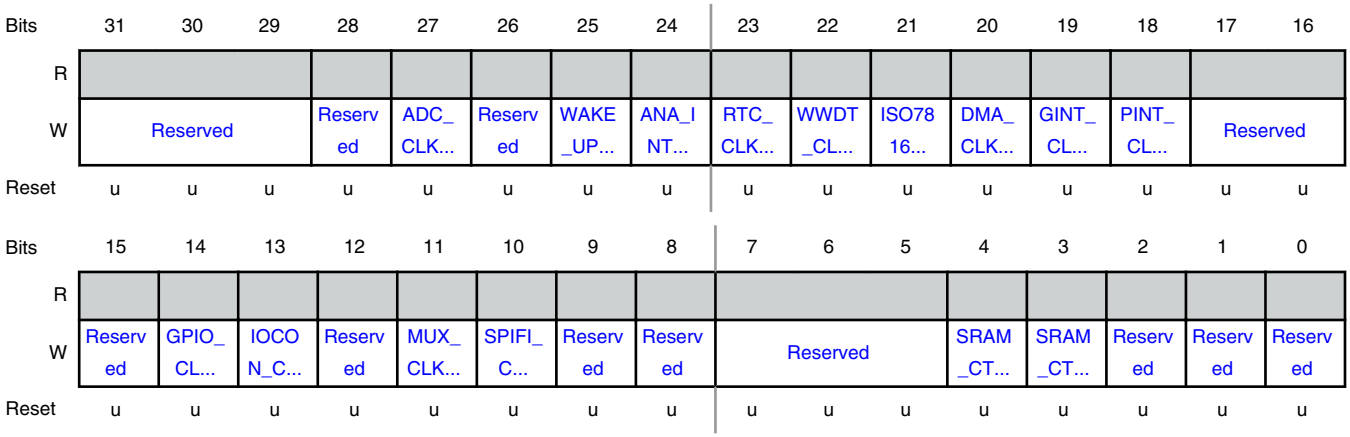
Offset

Register	Offset
AHBCLKCTRLSET0	220h

Function

The clocks for the AHB Peripherals can be enabled or disabled individually in AHBCLKCTRL0 and AHBCLKCTRL1. However, it can be more efficient to enable one or more clocks by just setting bits in the AHBCLKCTRLSET0 and AHBCLKCTRLSET1.

Diagram



Fields

Field	Function
31-29 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
28 —	RESERVED Do not modify the value in this field.
27 ADC_CLK_SET	ADC Set Writing one to this field sets the AHBCLKCTRL0[ADC] bit.
26 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
25 WAKE_UP_TIMERS_CLK_SET	WAKE_UP_TIMERS Set Writing one to this field sets the AHBCLKCTRL0[WAKE_UP_TIMERS] bit.
24 ANA_INT_CTRL_CLK_SET	ANA_INT_CTRL Set Writing one to this field sets the ANA_INT_CTRL bit in the AHBCLKCTRL0 register.
23 RTC_CLK_SET	RTC Set Writing one to this field sets the AHBCLKCTRL0[RTC] bit.
22 WWDT_CLK_SET	WWDT Set Writing one to this field sets the AHBCLKCTRL0[WWDT] bit.
21 ISO7816_CLK_SET	ISO7816 Set Writing one to this field sets the AHBCLKCTRL0[ISO7816] bit.
20 DMA_CLK_SET	DMA Set Writing one to this field sets the AHBCLKCTRL0[DMA] bit.
19 GINT_CLK_SET	GINT Set Writing one to this field sets the AHBCLKCTRL0[GINT] bit.
18 PINT_CLK_SET	PINT Set Writing one to this field sets the AHBCLKCTRL0[PINT] bit.

Table continues on the next page...

Table continued from the previous page...

Field	Function
17-15 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
14 GPIO_CLK_SET	GPIO Set Writing one to this field sets the AHBCLKCTRL0[GPIO] bit.
13 IOCON_CLK_SET	IOCON Set Writing one to this field sets the AHBCLKCTRL0[IOCON] bit.
12 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
11 MUX_CLK_SET	MUX Set Writing one to this field sets the AHBCLKCTRL0[MUX] bit.
10 SPIFI_CLK_SET	SPIFI Set Writing one to this field sets the AHBCLKCTRL0[SPIFI] bit.
9 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
8 —	RESERVED Do not modify the value in this field.
7-5 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
4 SRAM_CTRL1_CLK_SET	SRAM_CTRL1 Set Writing one to this field sets the AHBCLKCTRL0[SRAM_CTRL1] bit.
3 SRAM_CTRL0_CLK_SET	SRAM_CTRL0 Set Writing one to this field sets the AHBCLKCTRL0[SRAM_CTRL0] bit.

Table continues on the next page...

Table continued from the previous page...

Field	Function
2 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
1 —	RESERVED Do not modify the value in this field.
0 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

2.1.16 AHBCLKCTRL1 Bits Set Register (AHBCLKCTRLSET1)

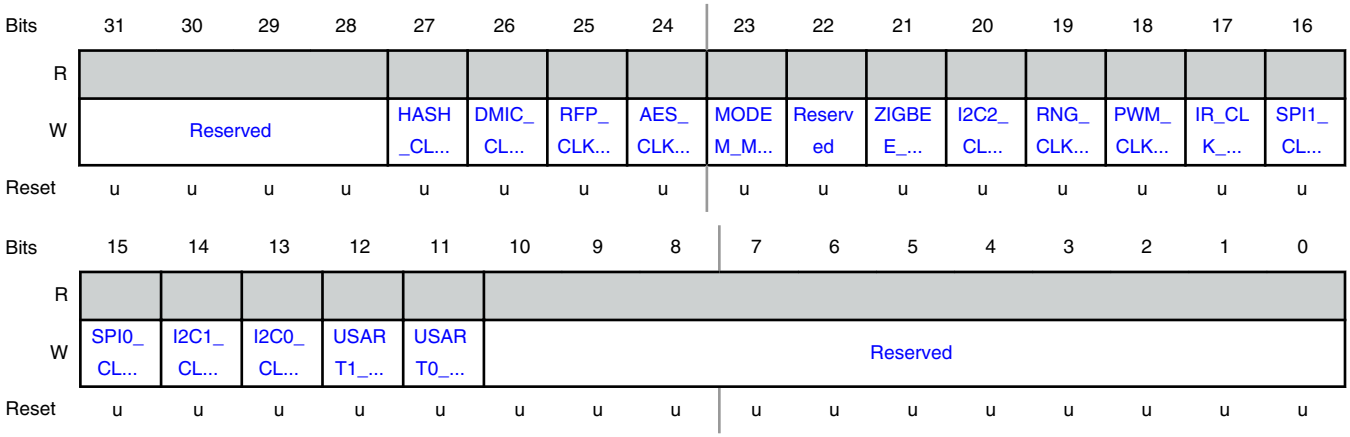
Offset

Register	Offset
AHBCLKCTRLSET1	224h

Function

The clocks for the AHB Peripherals can be enabled or disabled individually in AHBCLKCTRL0 and AHBCLKCTRL1. However, it can be more efficient to enable one or more clocks by just setting bits in the AHBCLKCTRLSET0 and AHBCLKCTRLSET1.

Diagram



Fields

Field	Function
31-28 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
27 HASH_CLK_SET	HASH Writing one to this field sets the AHBCLKCTRL1[HASH] bit.
26 DMIC_CLK_SET	DMIC Set Writing one to this field sets the AHBCLKCTRL1[DMIC] bit.
25 RFP_CLK_SET	RFP Set Writing one to this field sets the AHBCLKCTRL1[RFP] bit.
24 AES_CLK_SET	AES Set Writing one to this field sets the AHBCLKCTRL1[AES] bit.
23 MODEM_MASTER_CLK_SET	MODEM_MASTER Set Writing one to this field sets the AHBCLKCTRL1[MODEM_MASTER] bit.
22 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
21 ZIGBEE_CLK_SET	ZIGBEE Set Writing one to this field sets the AHBCLKCTRL1[ZIGBEE] bit.
20 I2C2_CLK_SET	I2C2 Set Writing one to this field sets the AHBCLKCTRL1[I2C2] bit.
19 RNG_CLK_SET	RNG Set Writing one to this field sets the AHBCLKCTRL1[RNG] bit.
18 PWM_CLK_SET	PWM Set Writing one to this field sets the AHBCLKCTRL1[PWM] bit.
17 IR_CLK_SET	IR Set Writing one to this field sets the AHBCLKCTRL1[IR] bit.

Table continues on the next page...

Table continued from the previous page...

Field	Function
16 SPI1_CLK_SET	SPI1 Set Writing one to this field sets the AHBCLKCTRL1[SPI1] bit.
15 SPI0_CLK_SET	SPI0 Set Writing one to this field sets the AHBCLKCTRL1[SPI0] bit.
14 I2C1_CLK_SET	I2C1 Set Writing one to this field sets the AHBCLKCTRL1[I2C1] bit.
13 I2C0_CLK_SET	I2C0 Set Writing one to this field sets the AHBCLKCTRL1[I2C0] bit.
12 USART1_CLK_SET	USART1 Set Writing one to this field sets the AHBCLKCTRL1[USART1] bit.
11 USART0_CLK_SET	USART0 Set Writing one to this field sets the AHBCLKCTRL1[USART0] bit.
10-0 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

2.1.17 AHBCLKCTRL0 Bits Clear Register (AHBCLKCTRLCLR0)

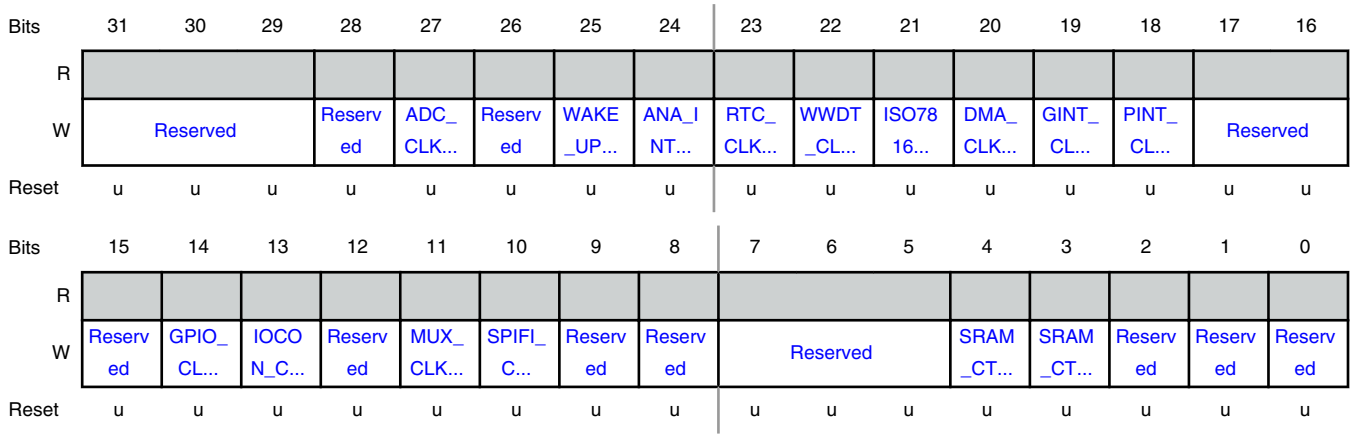
Offset

Register	Offset
AHBCLKCTRLCLR0	240h

Function

The clocks for the AHB Peripherals can be enabled or disabled individually in AHBCLKCTRL0 and AHBCLKCTRL1. However, it can be more efficient to disable one or more clocks by just setting bits in the AHBCLKCTRLCLR0 and AHBCLKCTRLCLR1.

Diagram



Fields

Field	Function
31-29 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
28 —	RESERVED Do not modify the value in this field.
27 ADC_CLK_CLR	ADC Clear Writing one to this field clears the AHBCLKCTRL0[ADC] bit.
26 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
25 WAKE_UP_TIMERS_CLK_SET	WAKE_UP_TIMERS Clear Writing one to this field clears the AHBCLKCTRL0[WAKE_UP_TIMERS] bit.
24 ANA_INT_CTRL_CLK_SET	ANA_INT_CTRL Clear Writing one to this field clears the AHBCLKCTRL0[ANA_INT_CTRL] bit.
23 RTC_CLK_CLR	RTC Clear Writing one to this field clears the AHBCLKCTRL0[RTC] bit.
22 WWDT_CLK_CLR	WWDT Clear Writing one to this field clears the AHBCLKCTRL0[WWDT] bit.

Table continues on the next page...

Table continued from the previous page...

Field	Function
21 ISO7816_CLK_CLR	ISO7816 Clear Writing one to this field clears the AHBCLKCTRL0[ISO7816] bit.
20 DMA_CLK_CLR	DMA Clear Writing one to this field clears the AHBCLKCTRL0[DMA] bit.
19 GINT_CLK_CLR	GINT Clear Writing one to this field clears the AHBCLKCTRL0[GINT] bit.
18 PINT_CLK_CLR	PINT Clear Writing one to this field clears the AHBCLKCTRL0[PINT] bit.
17-15 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
14 GPIO_CLK_CLR	GPIO Clear Writing one to this field clears the AHBCLKCTRL0[GPIO] bit.
13 IOCON_CLK_CLR	IOCON Clear Writing one to this field clears the AHBCLKCTRL0[IOCON] bit.
12 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
11 MUX_CLK_CLR	MUX Clear Writing one to this field clears the AHBCLKCTRL0[MUX] bit.
10 SPIFI_CLK_CLR	SPIFI Clear Writing one to this field clears the AHBCLKCTRL0[SPIFI] bit.
9 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
8 —	RESERVED Do not modify the value in this field.
7-5 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
4 SRAM_CTRL1_CLK_CLR	SRAM_CTRL1 Clear Writing one to this field clears the AHBCLKCTRL0[SRAM_CTRL1] bit.
3 SRAM_CTRL0_CLK_CLR	SRAM_CTRL0 Clear Writing one to this field clears the AHBCLKCTRL0[SRAM_CTRL0] bit.
2 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
1 —	RESERVED Do not modify the value in this field.
0 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

2.1.18 AHBCLKCTRL1 Bits Clear Register (AHBCLKCTRLCLR1)

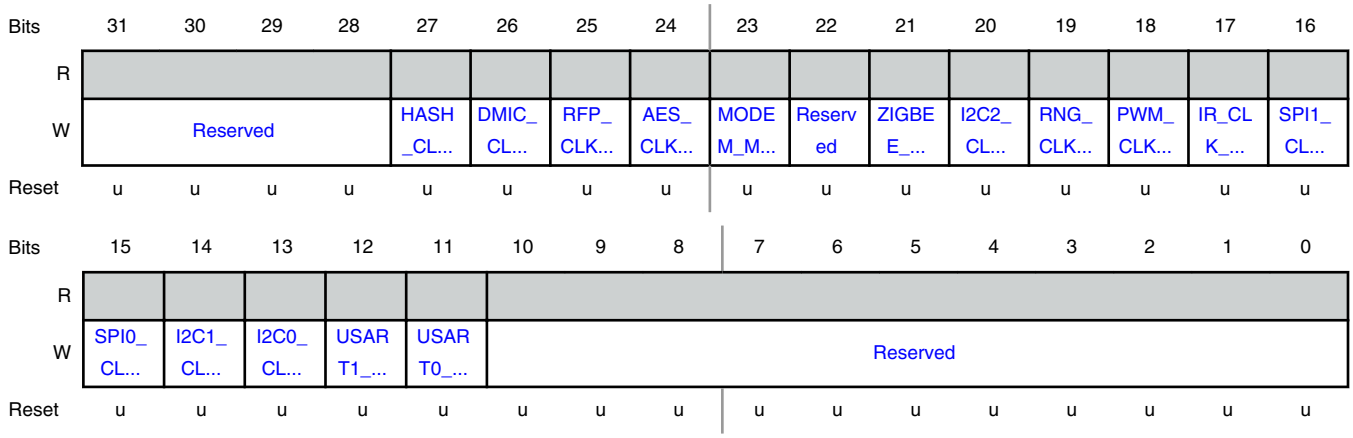
Offset

Register	Offset
AHBCLKCTRLCLR1	244h

Function

The clocks for the AHB Peripherals can be enabled and disabled individually in AHBCLKCTRL0 and AHBCLKCTRL1. However, it can be more efficient to disable one or more clocks by just setting bits in the AHBCLKCTRLCLR0 and AHBCLKCTRLCLR1.

Diagram



Fields

Field	Function
31-28 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
27 HASH_CLK_CLR	HASH Clear Writing one to this field clears the AHBCLKCTRL1[HASH] bit.
26 DMIC_CLK_CLR	DMIC Clear Writing one to this field clears the AHBCLKCTRL1[DMIC] bit.
25 RFP_CLK_CLR	RFP Clear Writing one to this field clears the AHBCLKCTRL1[RFP] bit.
24 AES_CLK_CLR	AES Clear Writing one to this field clears the AHBCLKCTRL1[AES] bit.
23 MODEM_MASTER_CLK_CLR	MODEM_MASTER Clear Writing one to this field clears the AHBCLKCTRL1[MODEM_MASTER] bit.
22 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
21 ZIGBEE_CLK_CLR	ZIGBEE Clear Writing one to this field clears the AHBCLKCTRL1[ZIGBEE] bit.

Table continues on the next page...

Table continued from the previous page...

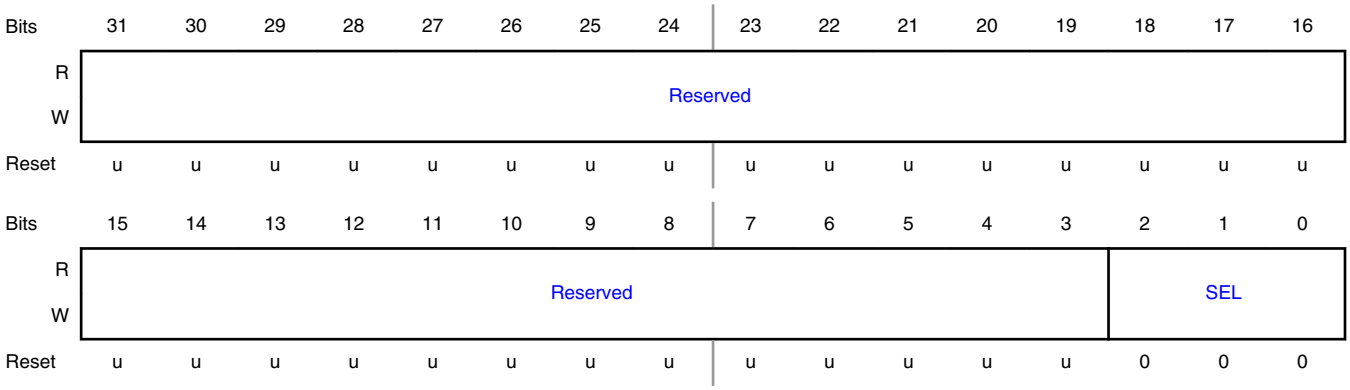
Field	Function
20 I2C2_CLK_CLR	I2C2 Clear Writing one to this field clears the AHBCLKCTRL1[I2C2] bit.
19 RNG_CLK_CLR	RNG Clear Writing one to this field clears the AHBCLKCTRL1[RNG] bit.
18 PWM_CLK_CLR	PWM Clear Writing one to this field clears the AHBCLKCTRL1[PWM] bit.
17 IR_CLK_CLR	IR Clear Writing one to this field clears the AHBCLKCTRL1[IR] bit.
16 SPI1_CLK_CLR	SPI1 Clear Writing one to this field clears the AHBCLKCTRL1[SPI1] bit.
15 SPI0_CLK_CLR	SPI0 Clear Writing one to this field clears the AHBCLKCTRL1[SPI0] bit.
14 I2C1_CLK_CLR	I2C1 Clear Writing one to this field clears the AHBCLKCTRL1[I2C1] bit.
13 I2C0_CLK_CLR	I2C0 Clear Writing one to this field clears the AHBCLKCTRL1[I2C0] bit.
12 USART1_CLK_CLR	USART1 Clear Writing one to this field clears the AHBCLKCTRL1[USART1] bit.
11 USART0_CLK_CLR	USART0 Clear Writing one to this field clears the AHBCLKCTRL1[USART0] bit.
10-0 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

2.1.19 Main Clock Source Selection Register (MAINCLKSEL)

Offset

Register	Offset
MAINCLKSEL	280h

Diagram



Fields

Field	Function
31-3 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
2-0 SEL	Main Clock Source Selection 000b - 12 MHz free running oscillator (FRO) 010b - 32 MHz crystal oscillator (XTAL) 011b - 32 MHz free running oscillator (FRO) 100b - 48 MHz free running oscillator (FRO)

2.1.20 OSC32KCLK and OSC32MCLK Clock Sources Selection Register (OSC32CLKSEL)

Offset

Register	Offset
OSC32CLKSEL	284h

Function

Note: this register cannot be locked by CLOCKGENUPDATELOCKOUT register.

Diagram

Fields

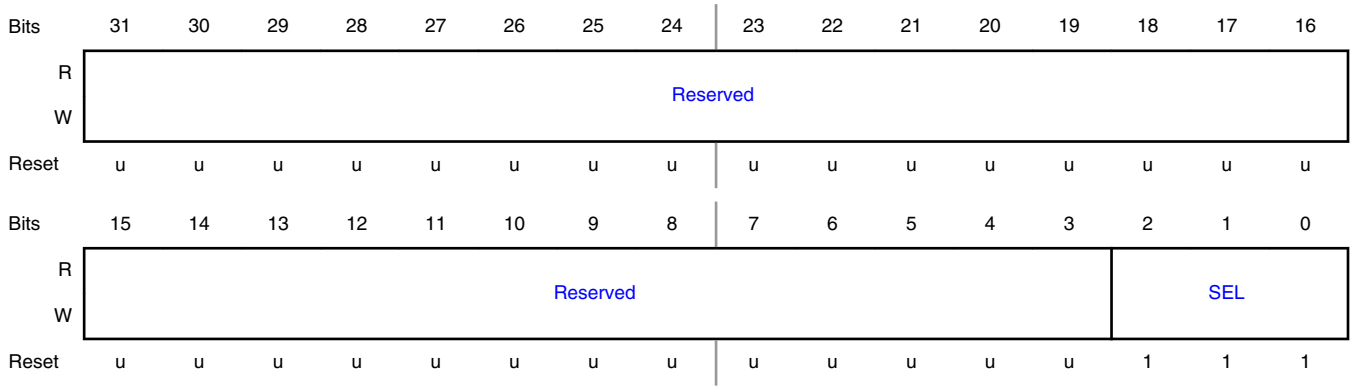
Field	Function
31-2 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
1 SEL32KHZ	OSC32KCLK Clock Source Selection 0b - 32 KHz free running oscillator (FRO) 1b - 32 KHz crystal oscillator
0 SEL32MHZ	OSC32MCLK Clock Source Selection 0b - 32 MHz free running oscillator (FRO) 1b - 32 MHz crystal oscillator

2.1.21 CLKOUT Clock Source Selection Register (CLKOUTSEL)

Offset

Register	Offset
CLKOUTSEL	288h

Diagram



Fields

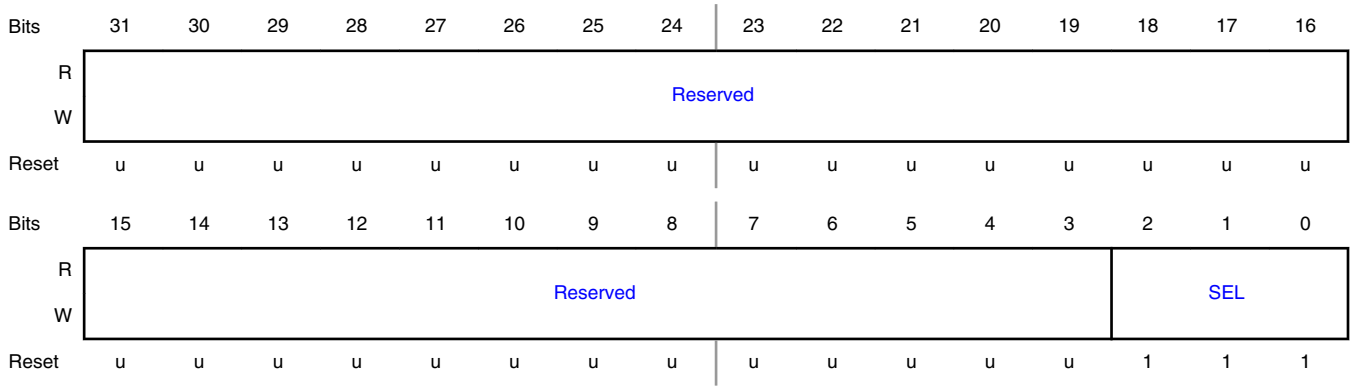
Field	Function
31-3 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
2-0 SEL	CLKOUT Clock Source Selection 000b - CPU and System Bus clock 001b - 32 kHz crystal oscillator (XTAL) 010b - 32 kHz free running oscillator (FRO) 011b - 32 MHz crystal oscillator (XTAL) 101b - 48 MHz free running oscillator (FRO) 110b - 1 MHz free running oscillator (FRO) 111b - No clock

2.1.22 SPIFI Clock Source Selection Register (SPIFICLKSEL)

Offset

Register	Offset
SPIFICLKSEL	2A0h

Diagram



Fields

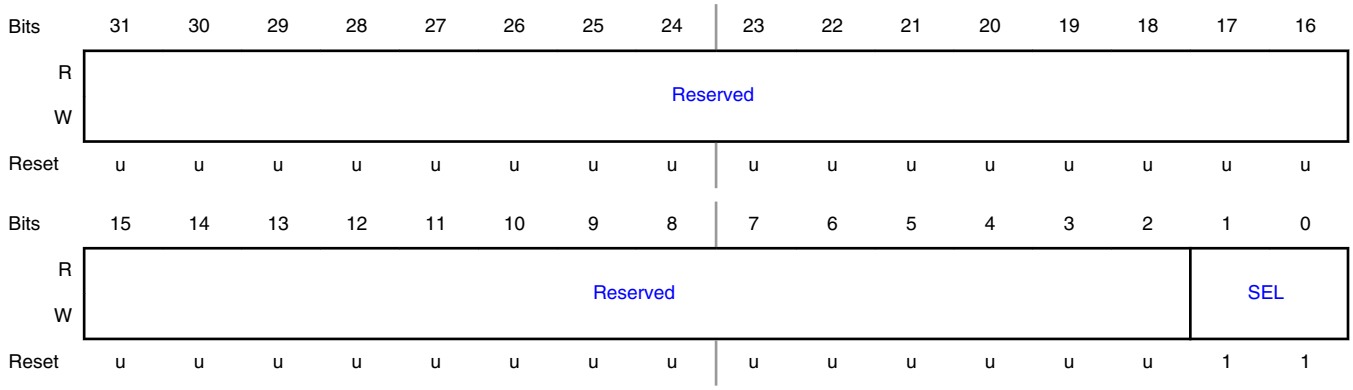
Field	Function
31-3 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
2-0 SEL	SPIFICLK Clock Source Selection 000b - CPU and System Bus clock 001b - 32 MHz crystal oscillator (XTAL) 010b - Reserved 011b - Reserved 100b - No clock 101b - No clock 110b - No clock 111b - No clock

2.1.23 ADC Clock Source Selection Register (ADCCLKSEL)

Offset

Register	Offset
ADCCLKSEL	2A4h

Diagram



Fields

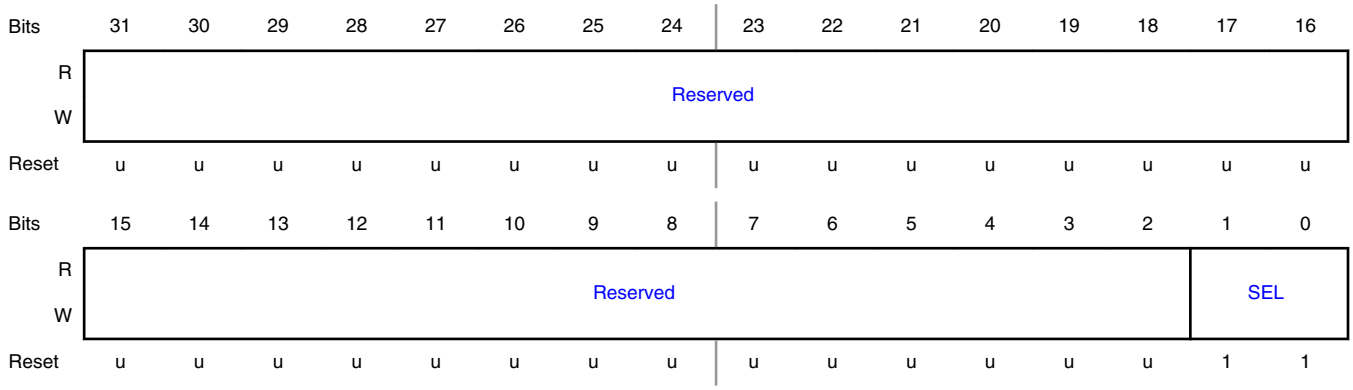
Field	Function
31-2 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
1-0 SEL	ADCCLK Clock Source Selection 00b - 32 MHz crystal oscillator (XTAL) 01b - FRO 12 MHz 10b - No clock 11b - No clock

2.1.24 USART0 and USART1 Clock Source Selection Register (USARTCLKSEL)

Offset

Register	Offset
USARTCLKSEL	2B0h

Diagram



Fields

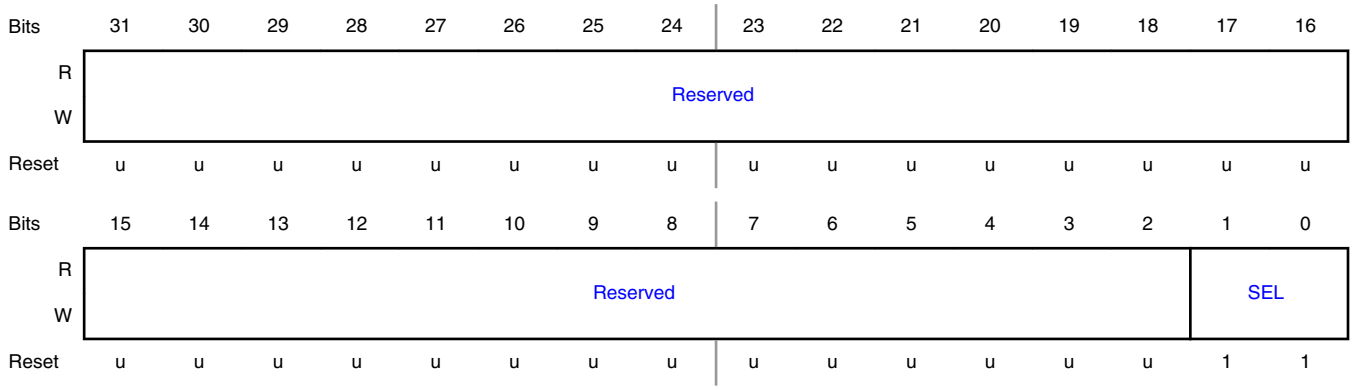
Field	Function
31-2 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
1-0 SEL	USARTCLK (USART0/1) Clock Source Selection 00b - Either 32 MHz FRO or 32 MHz XTAL (see OSC32CLKSEL) 01b - 48 MHz free running oscillator (FRO) 10b - Fractional Rate Generator clock (see FRGCLKSEL) 11b - No clock

2.1.25 I2C0, I2C1 and I2C2 Clock Source Selection Register (I2CCLKSEL)

Offset

Register	Offset
I2CCLKSEL	2B4h

Diagram



Fields

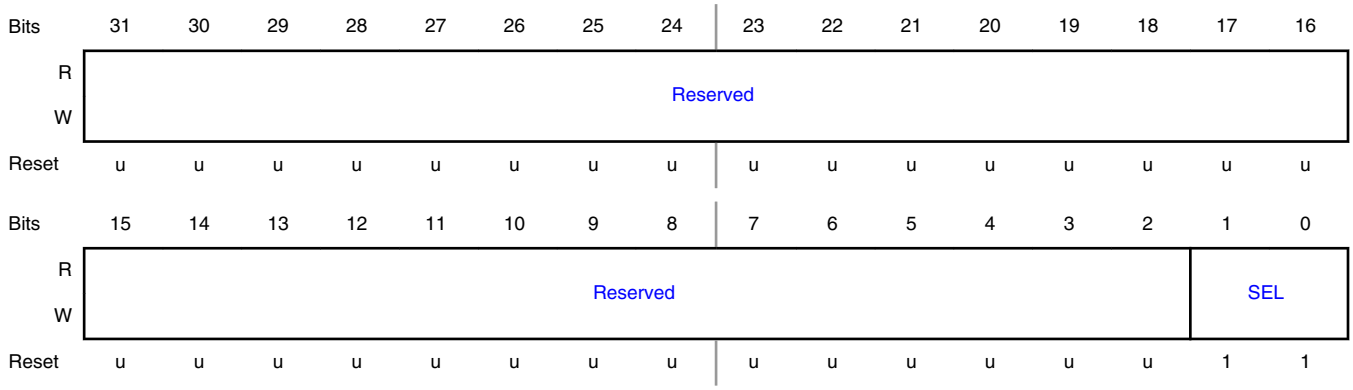
Field	Function
31-2 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
1-0 SEL	I2CCLK (I2C0/1/2) Clock Source Selection 00b - Either 32 MHz FRO or 32 MHz XTAL (see OSC32CLKSEL) 01b - 48 MHz free running oscillator (FRO) 10b - No clock 11b - No clock

2.1.26 SPI0 and SPI1 Clock Source Selection Register (SPICLKSEL)

Offset

Register	Offset
SPICLKSEL	2B8h

Diagram



Fields

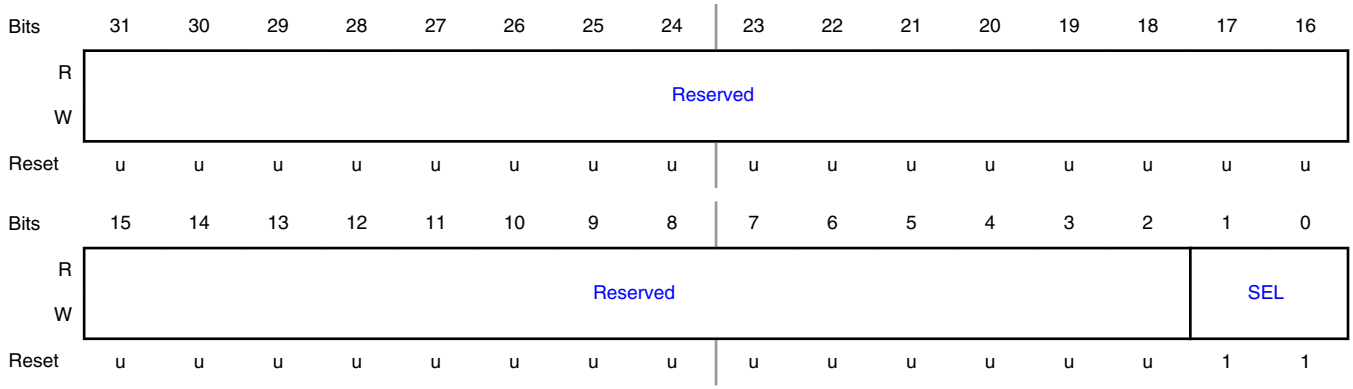
Field	Function
31-2 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
1-0 SEL	SPICLK (SPI0/1) Clock Source Selection 00b - Either 32 MHz FRO or 32 MHz XTAL (see OSC32CLKSEL) 01b - 48 MHz free running oscillator (FRO) 10b - No clock 11b - No clock

2.1.27 Infra Red Clock Source Selection Register (IRCLKSEL)

Offset

Register	Offset
IRCLKSEL	2BCh

Diagram



Fields

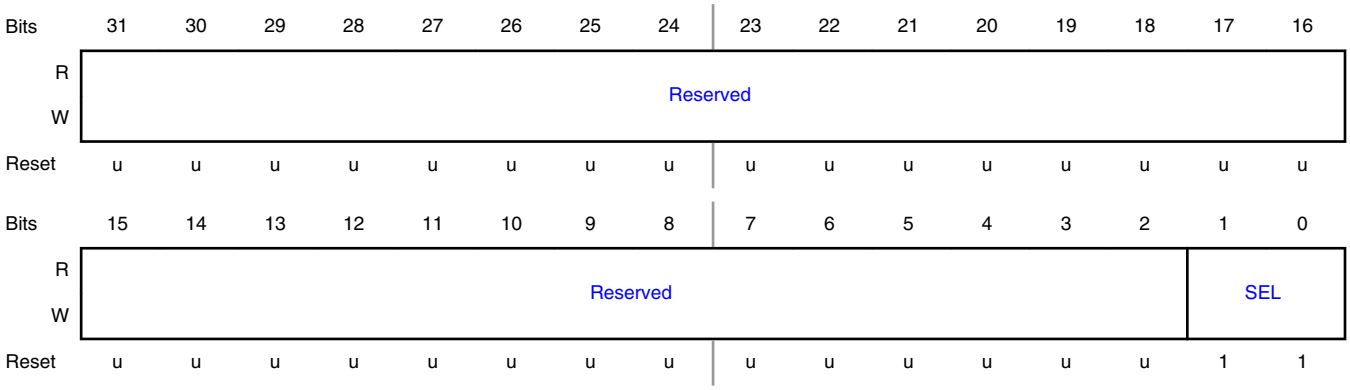
Field	Function
31-2 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
1-0 SEL	IRCLK (IR Blaster) Clock Source Selection 00b - Either 32 MHz FRO or 32 MHz XTAL (see OSC32CLKSEL) 01b - 48 MHz free running oscillator (FRO) 10b - No clock 11b - No clock

2.1.28 PWM Clock Source Selection Register (PWMCLKSEL)

Offset

Register	Offset
PWMCLKSEL	2C0h

Diagram



Fields

Field	Function
31-2 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

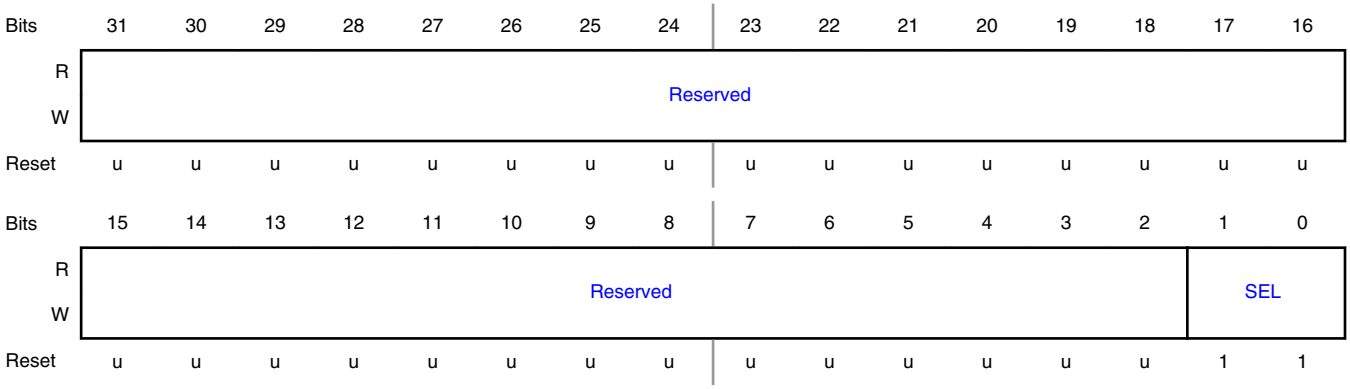
Field	Function
1-0 SEL	PWMCLK (PWM) Clock Source Selection 00b - Either 32 MHz FRO or 32 MHz XTAL (see OSC32CLKSEL) 01b - 48 MHz free running oscillator (FRO) 10b - No clock 11b - No Clock

2.1.29 Watchdog Timer Clock Source Selection Register (WDTCLKSEL)

Offset

Register	Offset
WDTCLKSEL	2C4h

Diagram



Fields

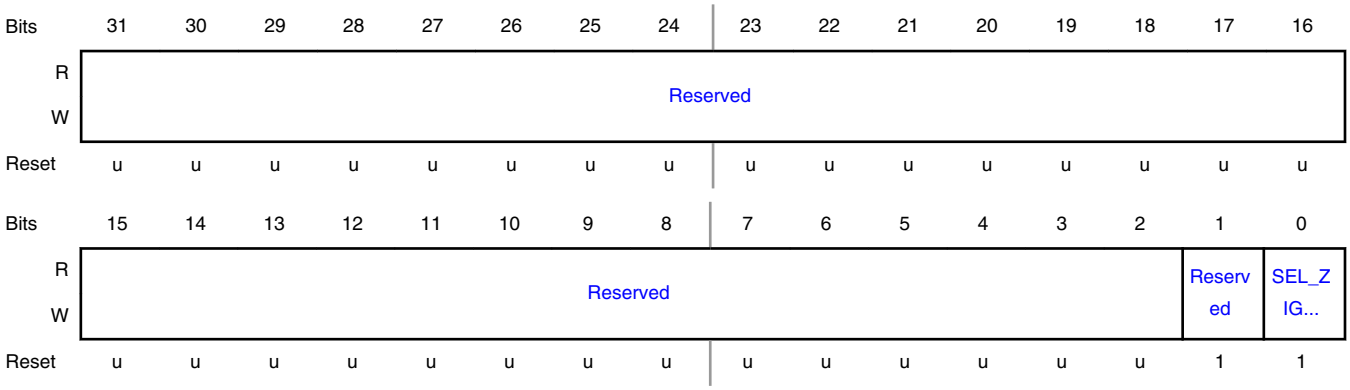
Field	Function
31-2 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
1-0 SEL	WDTCLK (Watchdog Timer) Clock Source Selection 00b - Either 32 MHz FRO or 32 MHz XTAL (see OSC32CLKSEL) 01b - Either 32 kHz FRO or 32 kHz XTAL (see OSC32CLKSEL) 10b - 1 MHz FRO 11b - No Clock

2.1.30 Modem Clock Source Selection Register (MODEMCLKSEL)

Offset

Register	Offset
MODEMCLKSEL	2CCh

Diagram



Fields

Field	Function
31-2 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
1 —	Reserved Reserved. User software should write ones to RESERVED1 bits. The value read from a RESERVED1 bit is not defined.
0 SEL_ZIGBEE	Zigbee/Thread Modem Clock Source Selection 0b - 32 MHz XTAL 1b - No Clock

2.1.31 Fractional Rate Generator (FRG) Clock Source Selection Register (FRGCLKSEL)

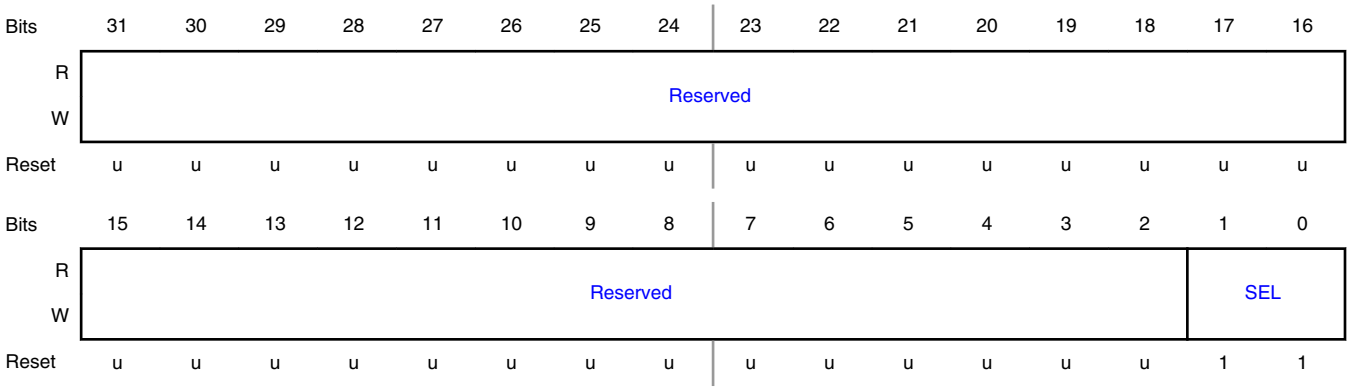
Offset

Register	Offset
FRGCLKSEL	2E8h

Function

The FRG is one of the USART clocking options.

Diagram



Fields

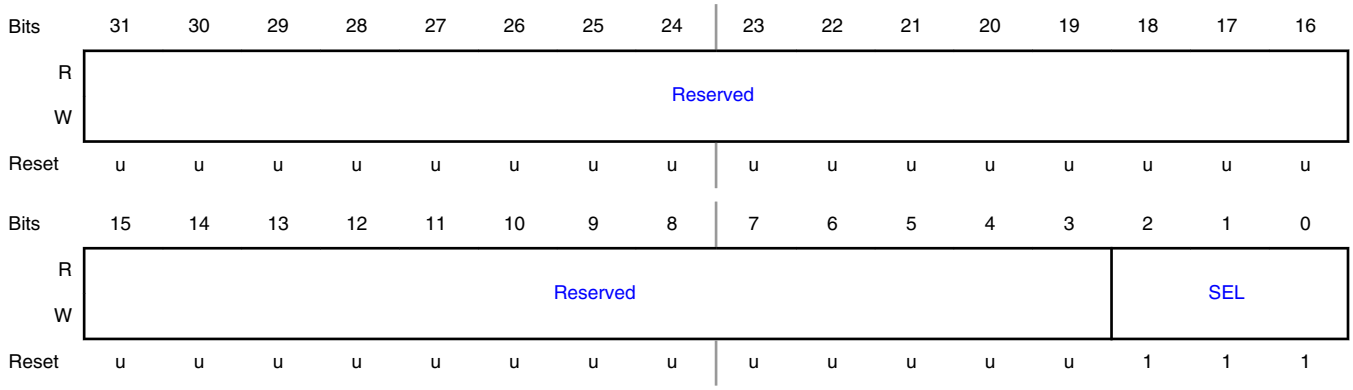
Field	Function
31-2 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
1-0 SEL	Fractional Rate Generator Clock Source Selection 00b - System Bus clock 01b - Either 32 MHz FRO or 32 MHz XTAL (see OSC32CLKSEL) 10b - 48 MHz free running oscillator (FRO) 11b - No Clock

2.1.32 Digital microphone (DMIC) Subsystem Clock Source Selection Register (DMICCLKSEL)

Offset

Register	Offset
DMICCLKSEL	2ECh

Diagram



Fields

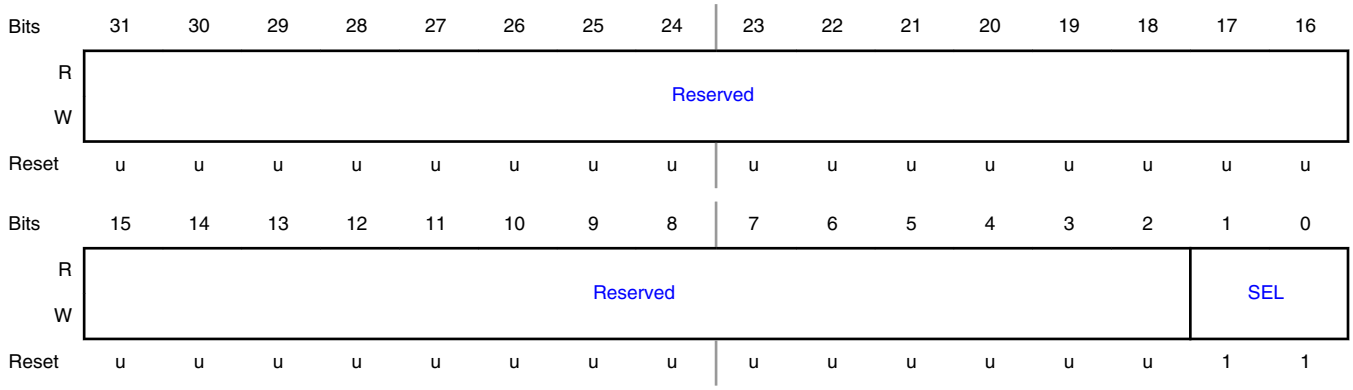
Field	Function
31-3 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
2-0 SEL	DMIC Clock Source Selection 000b - System Bus clock 001b - Either 32 kHz FRO or 32 kHz XTAL (see OSC32CLKSEL) 010b - 48 MHz free running oscillator (FRO) 011b - External clock 100b - 1 MHz free running oscillator (FRO) 101b - 12 MHz free running oscillator (FRO) 110b - No clock 111b - No Clock

2.1.33 Wake-up Timer Clock Source Selection Register (WKTCLKSEL)

Offset

Register	Offset
WKTCLKSEL	2F0h

Diagram



Fields

Field	Function
31-2 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
1-0 SEL	Wake-up Timers Clock Source Selection 00b - Either 32 kHz FRO or 32 kHz XTAL (see OSC32CLKSEL) 01b - Reserved 10b - No clock 11b - No Clock

2.1.34 SYSTICK Clock Divider Register (SYSTICKCLKDIV)

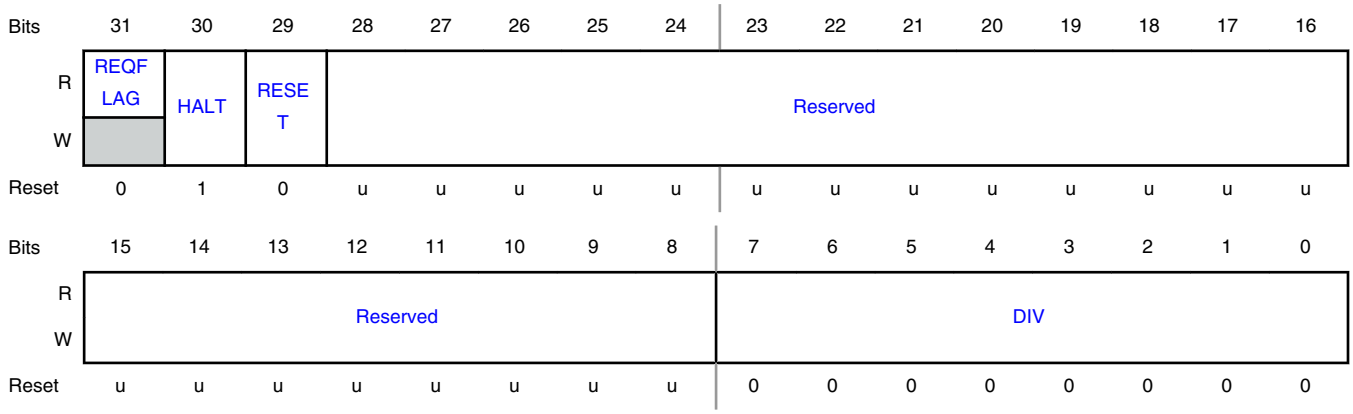
Offset

Register	Offset
SYSTICKCLKDIV	300h

Function

The SYSTICK clock can drive the SYSTICK function within the processor.

Diagram



Fields

Field	Function
31 REQFLAG	Divider Satus Flag Set when a change is made to the divider value, cleared when the change is complete.
30 HALT	Halts Divider Counter This allows the dividers clock source to be changed without the risk of a glitch at the output.
29 RESET	Reset Divider Counter Used to make sure a new divider value is used right away rather than completing the previous count.
28-8 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
7-0 DIV	Clock Divider Setting Divider ratio is (DIV+1). E.g. 0: Divide by 1 and 255: Divide by 256.

2.1.35 TRACE Clock Divider Register (TRACECLKDIV)

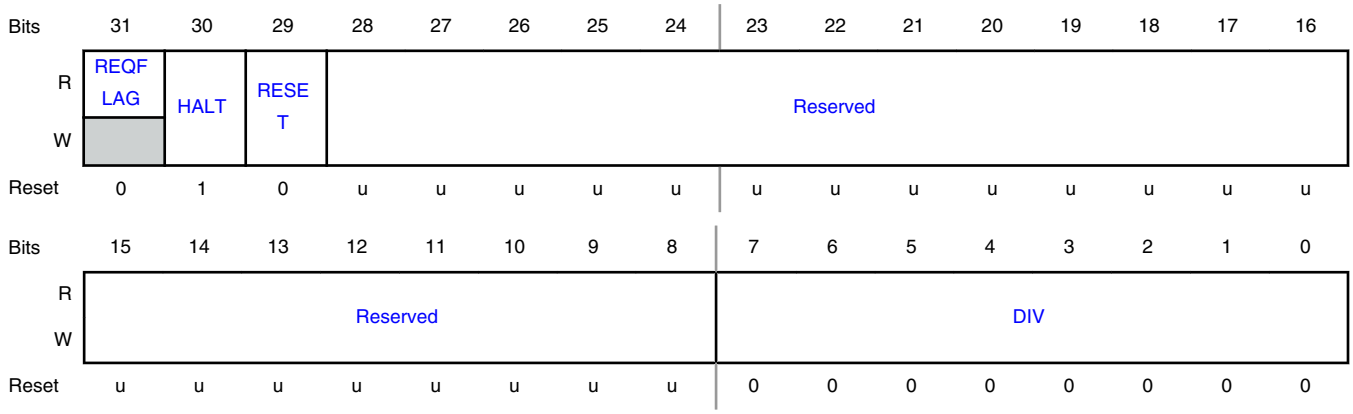
Offset

Register	Offset
TRACECLKDIV	304h

Function

This register is used for part of the Serial debugger (SWD) feature.

Diagram



Fields

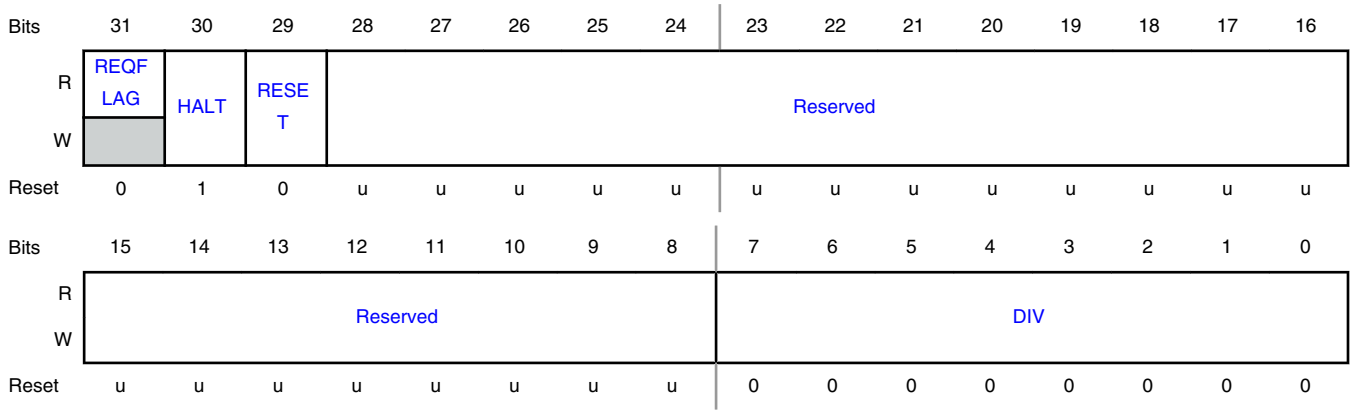
Field	Function
31 REQFLAG	Divider Satus Flag Set when a change is made to the divider value, cleared when the change is complete.
30 HALT	Halts Divider Counter This allows the dividers clock source to be changed without the risk of a glitch at the output.
29 RESET	Resets Divider Counter Used to make sure a new divider value is used right away rather than completing the previous count.
28-8 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
7-0 DIV	Clock Divider Setting Divider ratio is (DIV+1). E.g. 0: Divide by 1 and 255: Divide by 256.

2.1.36 Watchdog Timer Clock Divider Register (WDTCLKDIV)

Offset

Register	Offset
WDTCLKDIV	36Ch

Diagram



Fields

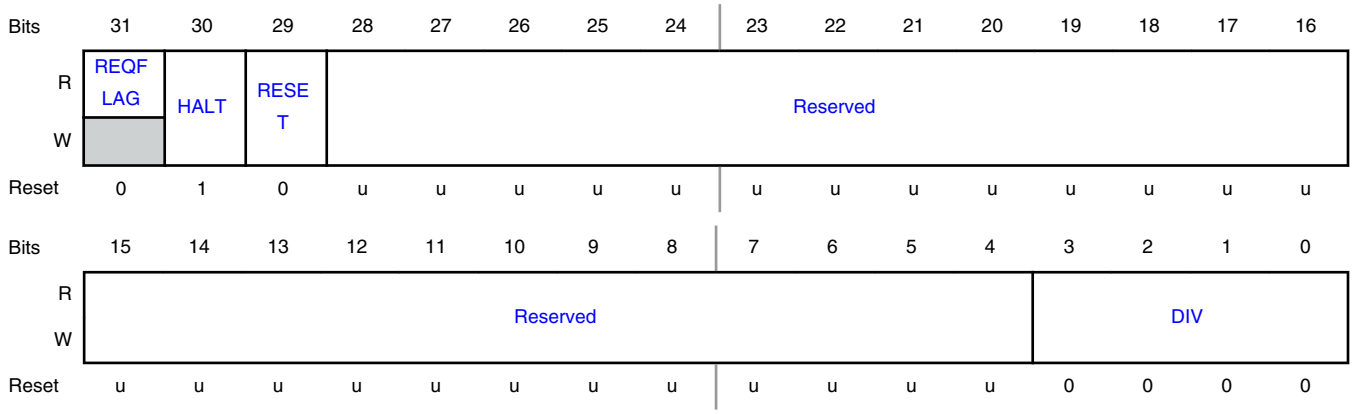
Field	Function
31 REQFLAG	Divider Satus Flag Set when a change is made to the divider value, cleared when the change is complete.
30 HALT	Halts Divider Counter This allows the dividers clock source to be changed without the risk of a glitch at the output.
29 RESET	Resets Divider Counter Used to make sure a new divider value is used right away rather than completing the previous count.
28-8 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
7-0 DIV	Clock Divider Setting Divider ratio is (DIV+1). E.g. 0: Divide by 1 and 255: Divide by 256.

2.1.37 Infra Red Clock Divider Register (IRCLKDIV)

Offset

Register	Offset
IRCLKDIV	378h

Diagram



Fields

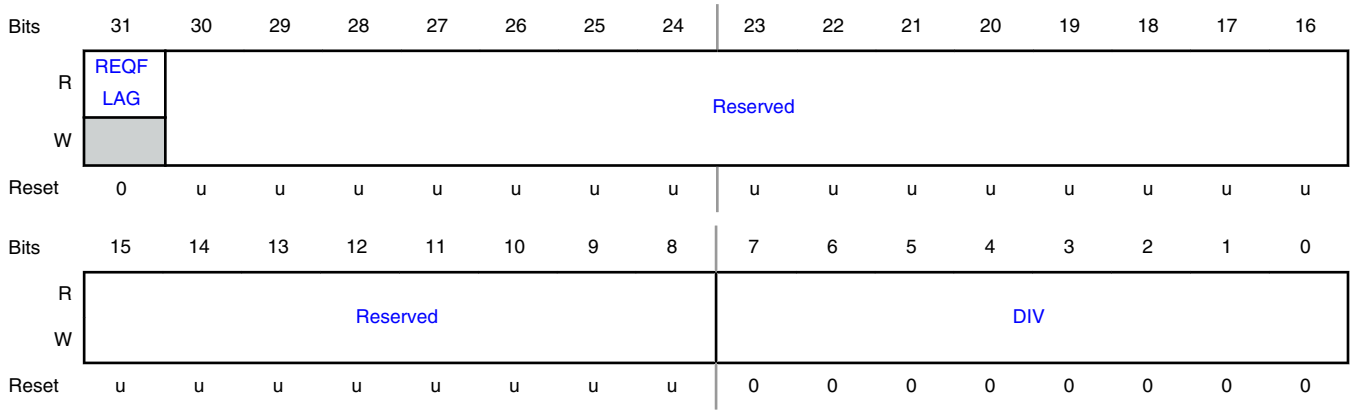
Field	Function
31 REQFLAG	Divider Satus Flag Set when a change is made to the divider value, cleared when the change is complete.
30 HALT	Halts Divider Counter This allows the dividers clock source to be changed without the risk of a glitch at the output.
29 RESET	Resets Divider Counter Used to make sure a new divider value is used right away rather than completing the previous count.
28-4 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
3-0 DIV	Clock Divider Setting Divider ratio is (DIV+1). E.g. 0: Divide by 1 and 15: Divide by 16.

2.1.38 System Clock Divider Register (AHBCLKDIV)

Offset

Register	Offset
AHBCLKDIV	380h

Diagram



Fields

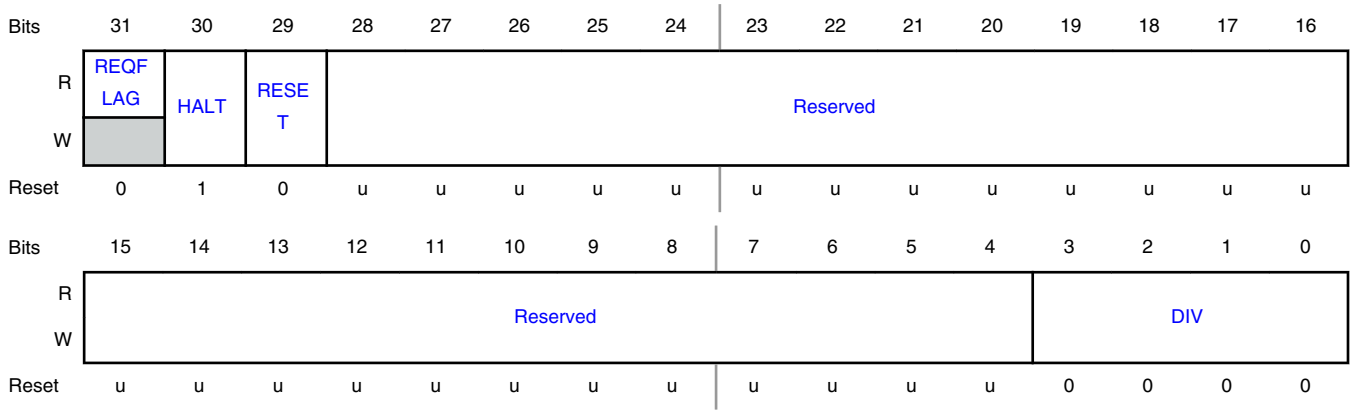
Field	Function
31 REQFLAG	Divider Satus Flag Set when a change is made to the divider value, cleared when the change is complete.
30-8 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
7-0 DIV	Clock Divider Setting Divider ratio is (DIV+1). E.g. 0: Divide by 1 and 255: Divide by 256.

2.1.39 CLKOUT Clock Divider Register (CLKOUTDIV)

Offset

Register	Offset
CLKOUTDIV	384h

Diagram



Fields

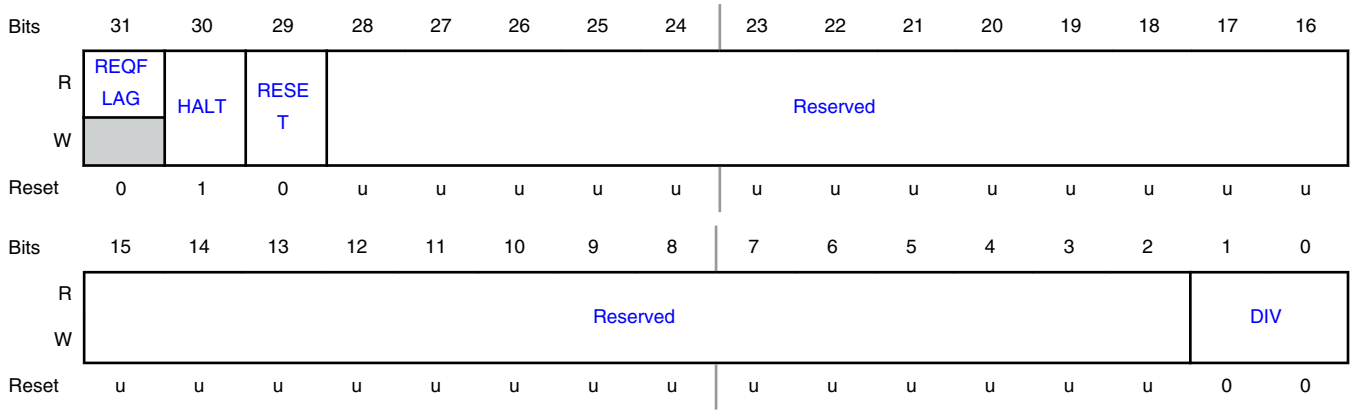
Field	Function
31 REQFLAG	Divider Satus Flag Set when a change is made to the divider value, cleared when the change is complete.
30 HALT	Halts Divider Counter This allows the dividers clock source to be changed without the risk of a glitch at the output.
29 RESET	Resets Divider Counter Used to make sure a new divider value is used right away rather than completing the previous count.
28-4 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
3-0 DIV	Clock Divider Setting Divider ratio is (DIV+1). E.g. 0: Divide by 1 and 15: Divide by 16.

2.1.40 SPIFI Clock Divider Register (SPIFICLKDIV)

Offset

Register	Offset
SPIFICLKDIV	390h

Diagram



Fields

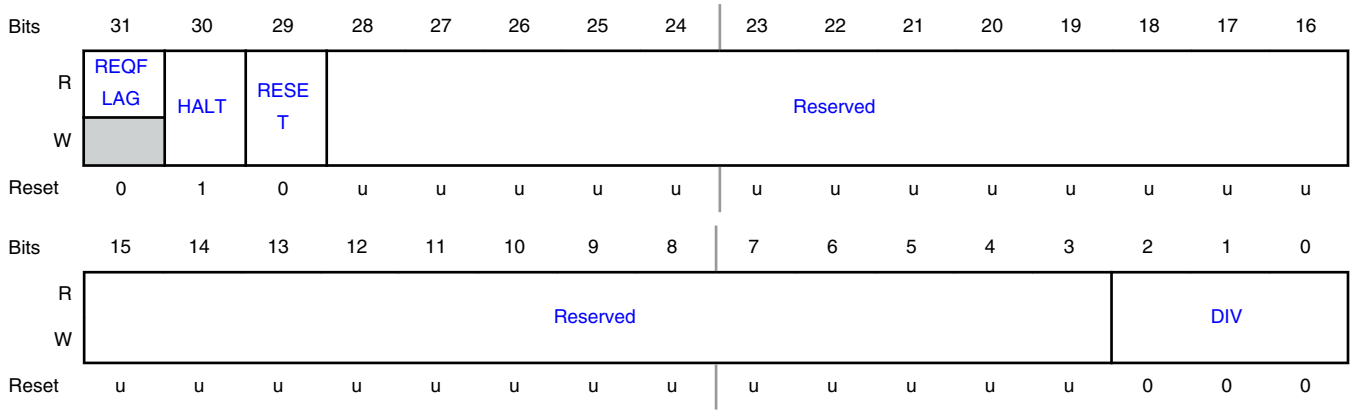
Field	Function
31 REQFLAG	Divider Satus Flag Set when a change is made to the divider value, cleared when the change is complete.
30 HALT	Halts Divider Counter This allows the dividers clock source to be changed without the risk of a glitch at the output.
29 RESET	Resets Divider Counter Used to make sure a new divider value is used right away rather than completing the previous count.
28-2 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
1-0 DIV	Clock Divider Setting Divider ratio is (DIV+1). E.g. 0: Divide by 1 and 3: Divide by 4.

2.1.41 ADC Clock Divider Register (ADCCLKDIV)

Offset

Register	Offset
ADCCLKDIV	394h

Diagram



Fields

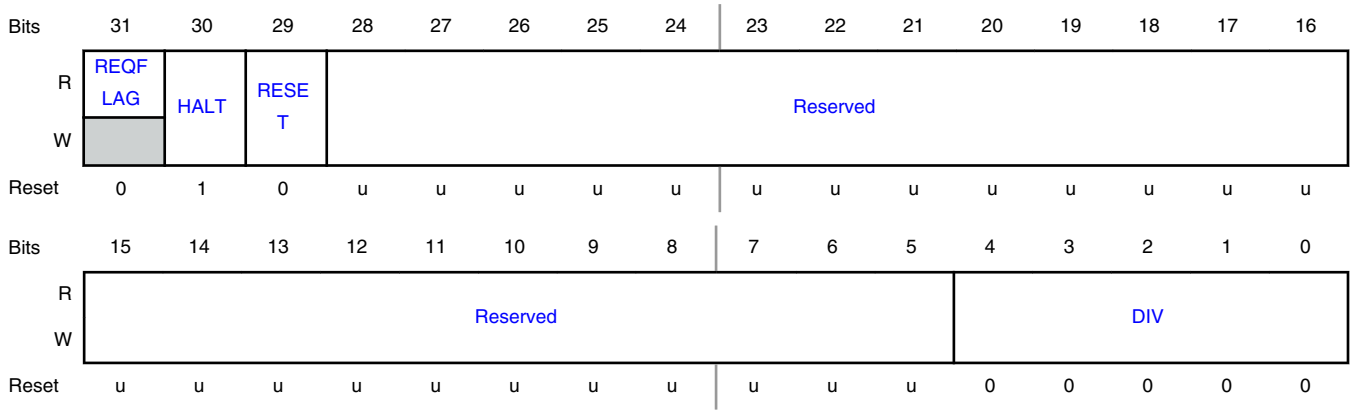
Field	Function
31 REQFLAG	Divider Satus Flag Set when a change is made to the divider value, cleared when the change is complete.
30 HALT	Halts Divider Counter This allows the dividers clock source to be changed without the risk of a glitch at the output.
29 RESET	Resets Divider Counter Used to make sure a new divider value is used right away rather than completing the previous count.
28-3 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
2-0 DIV	Clock Divider Setting Divider ratio is (DIV+1). E.g. 0: Divide by 1 and 7: Divide by 8.

2.1.42 Real Time Clock Divider (1 kHz clock generation) Register (RTCCLKDIV)

Offset

Register	Offset
RTCCLKDIV	398h

Diagram



Fields

Field	Function
31 REQFLAG	Divider Satus Flag Set when a change is made to the divider value, cleared when the change is complete.
30 HALT	Halts Divider Counter This allows the dividers clock source to be changed without the risk of a glitch at the output.
29 RESET	Resets Divider Counter Used to make sure a new divider value is used right away rather than completing the previous count.
28-5 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
4-0 DIV	Clock Divider Setting Divider ratio is (DIV+1). E.g. 0: Divide by 1 and 31: Divide by 32.

2.1.43 Fractional Rate Generator Divider Register (FRGCTRL)

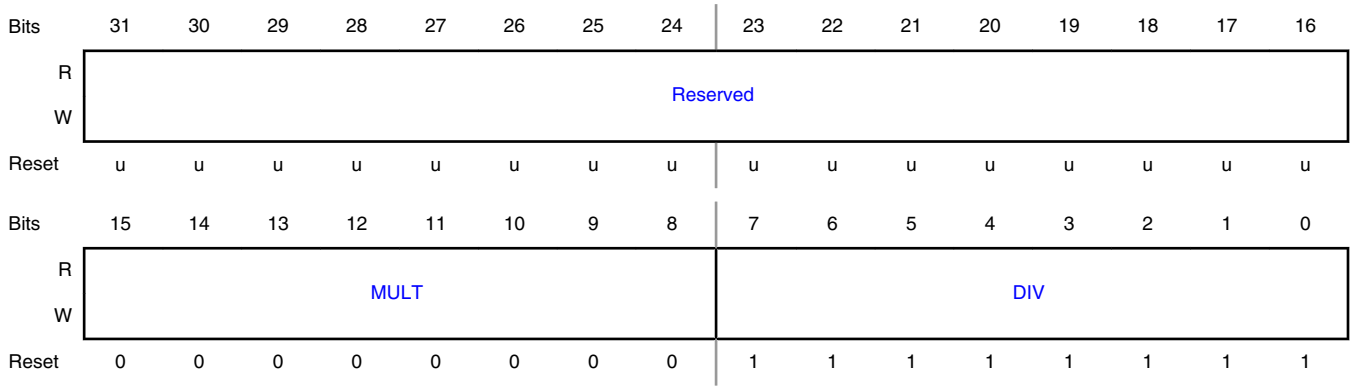
Offset

Register	Offset
FRGCTRL	3A0h

Function

The FRG is one of the USART clocking options.

Diagram



Fields

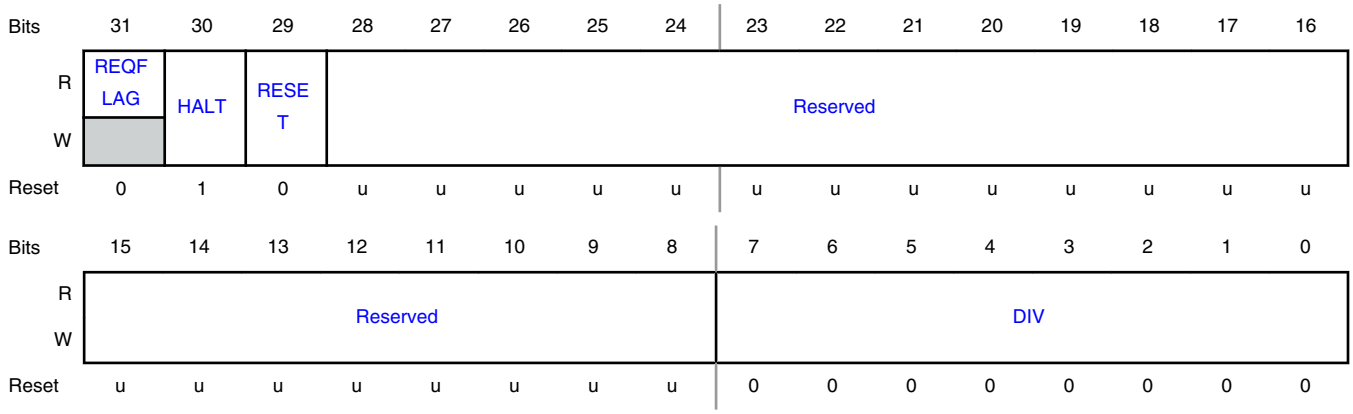
Field	Function
31-16 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
15-8 MULT	Numerator of the Fractional Divider MULT is equal to the programmed value
7-0 DIV	Fractional Divider Denominator of the fractional divider is equal to the (DIV+1). Always set to 0xFF to use with the fractional baud rate generator : fout = fin / (1 + MULT/(DIV+1))

2.1.44 DMIC Clock Divider Register (DMICCLKDIV)

Offset

Register	Offset
DMICCLKDIV	3A8h

Diagram



Fields

Field	Function
31 REQFLAG	Divider Satus Flag Set when a change is made to the divider value, cleared when the change is complete.
30 HALT	Halts Divider Counter This allows the dividers clock source to be changed without the risk of a glitch at the output.
29 RESET	Resets Divider Counter Used to make sure a new divider value is used right away rather than completing the previous count.
28-8 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
7-0 DIV	Clock Divider Setting Divider ratio is (DIV+1). E.g. 0: Divide by 1 and 255: Divide by 256.

2.1.45 Real Time Clock Divider (1 Hz clock generation) Register (RTC1HZCLKDIV)

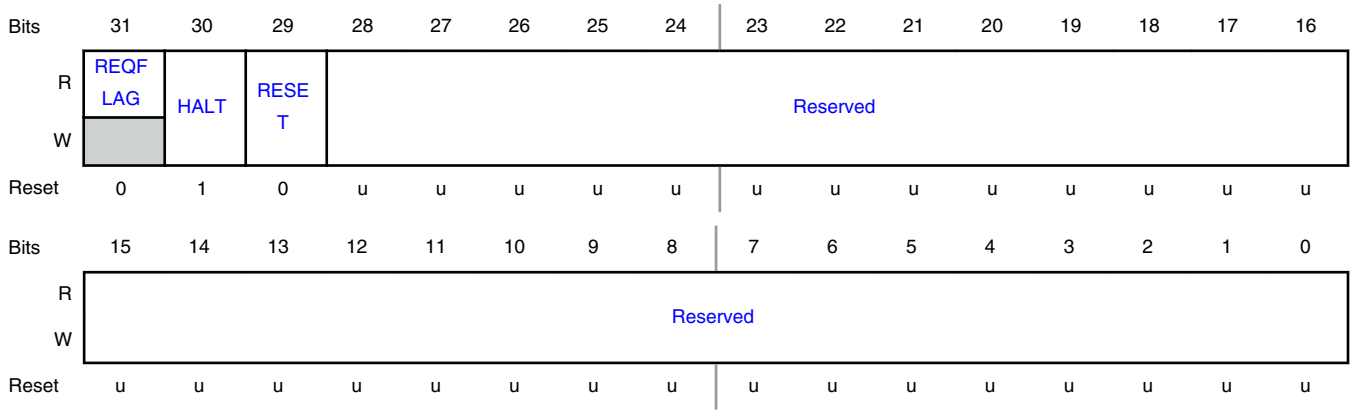
Offset

Register	Offset
RTC1HZCLKDIV	3ACh

Function

The divider is fixed to 32768 Hz

Diagram



Fields

Field	Function
31 REQFLAG	Divider Satus Flag Set when a change is made to the divider value, cleared when the change is complete.
30 HALT	Halts Divider Counter This allows the dividers clock source to be changed without the risk of a glitch at the output.
29 RESET	Resets Divider Counter Used to make sure a new divider value is used right away rather than completing the previous count.
28-0 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

2.1.46 Clock Configuration Registers Access Register (CLOCKGENUPDATELOCKOUT)

Offset

Register	Offset
CLOCKGENUPDATELOCKOUT	3FCh

Diagram

Fields

Field	Function
31-1 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
0 LOCK	Disable Access to Clock Control Registers When set, it disables access to clock control registers (like xxxDIV, xxxSEL). It affects all clock control registers except OSC32CLKSEL.

2.1.47 Random Number Generator Clocks Control Register (RNGCLKCTRL)

Offset

Register	Offset
RNGCLKCTRL	59Ch

Diagram

Fields

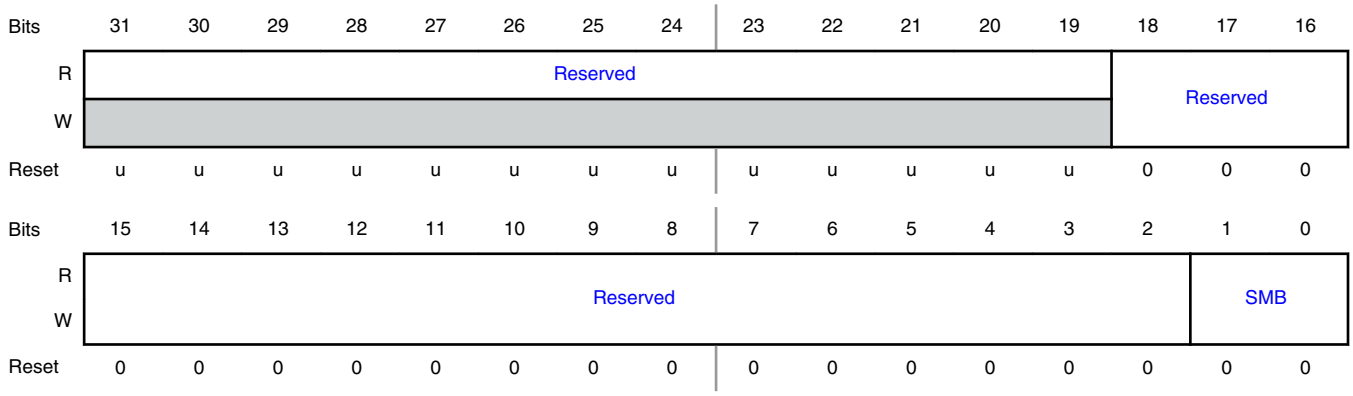
Field	Function
31-1 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
0 ENABLE	Enable Clocks used by the Random Number Generator (RNG)

2.1.48 All SRAMs Common Control Register (SRAMCTRL)

Offset

Register	Offset
SRAMCTRL	5A0h

Diagram



Fields

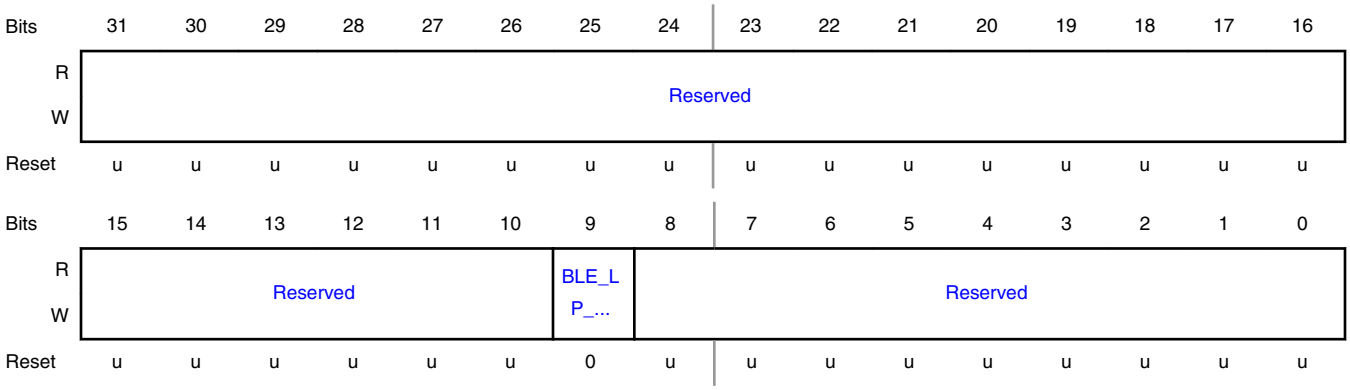
Field	Function
31-19 —	Reserved
18-2 —	Reserved
1-0 SMB	SMB This field should only be modified by the APIs.

2.1.49 32K Clock Enable Register (MODEMCTRL)

Offset

Register	Offset
MODEMCTRL	5CCh

Diagram



Fields

Field	Function
31-10 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
9 BLE_LP_OSC32K_EN	Enable 32K Clock When set, it enables the 32K clock to the USART 0/1, SPI0/1, PMC and the frequency measure block. Note: This control bit affects peripheral clocking.
8-0 —	RESERVED Reserved. User software should not modify the values in this field

2.1.50 XTAL 32 kHz Oscillator Capacitor Control Register (XTAL 32KCAP)

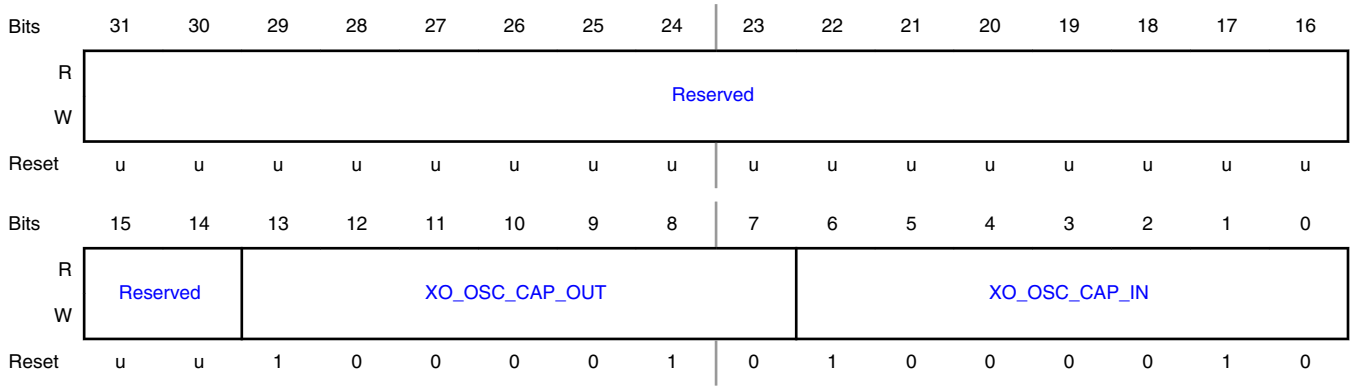
Offset

Register	Offset
XTAL32KCAP	5D4h

Function

This register is used for selection of the internal capacitors to be used on each of the XTAL 32 kHz pins.

Diagram



Fields

Field	Function
31-14 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
13-7 XO_OSC_CAP_OUT	XTAL_32K_N Internal Capacitor Setting Internal Capacitor setting for XTAL_32K_N. This setting selects the internal capacitance, to ground, that is connected to this XTAL pin. During device testing the capacitor banks are calibrated so accurate setting of the capacitance can be achieved. Software functions are provided to support the setting of this field. Capacitance range is to apporximately 24 pF.
6-0 XO_OSC_CAP_IN	XTAL_32K_P Internal Capacitor Setting This field selects the internal capacitance, to ground, that is connected to this XTAL pin. During device testing, the capacitor banks are calibrated, so accurate setting of the capacitance can be achieved. Software functions are provided to support the setting of this field. Capacitance range is to apporximately 24 pF.

2.1.51 32 MHz XTAL Control Register (XTAL32MCTRL)

Offset

Register	Offset
XTAL32MCTRL	5D8h

Diagram

Fields

Field	Function
31-2 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
1 —	Reserved Reserved. User software should write ones to RESERVED1 bits. The value read from a RESERVED1 bit is not defined.
0 DEACTIVATE_PMC_CTRL	32 MHz XTAL Enabled The XTAL is auto-started, by control from the PMC, on a power-up. Set this bit to independently control the XTAL using controls in ASYNC_SYSCON[XTAL32MCTRL]. This field is managed by the low level SDK functions and should not need to be modified from application code. 0b - Enable XTAL 32 MHz controls coming from PMC. 1b - Disable XTAL 32 MHz controls coming from PMC.

2.1.52 Start Logic 0 Wake-up Enable Register (STARTER0)

Offset

Register	Offset
STARTER0	680h

Function

Enable an interrupt for wake-up from deep-sleep mode. Some bits can also control wake-up from power-down mode

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved	NFCTAG	RTC	I2C2	PWM10	PWM9	PWM8	PWM7	PWM6	PWM5	PWM4	PWM3	PWM2	PWM1	PWM0	SPI1
Reset	u	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SPI0	I2C1	I2C0	USART1	USART0	TIMER1	TIMER0	SPIFI	PINT3	PINT2	PINT1	PINT0	IRBLAST...	GINT	DMA	WDT_BOD
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
30 NFCTAG	NFC Tag Interrupt Wake-up Only supported on devices with internal NFC tag. 0b - Wake-up disabled. 1b - Wake-up enabled. Valid from Deep-Sleep
29 RTC	Real Time Clock (RTC) Interrupt Wake-up 0b - Wake-up disabled. 1b - Wake-up enabled. Valid from Deep-Sleep and Power down.
28 I2C2	I2C2 Interrupt Wake-up 0b - Wake-up disabled. 1b - Wake-up enabled. Valid from Deep-Sleep.
27 PWM10	PWM10 Interrupt Wake-up 0b - Wake-up disabled. 1b - Wake-up enabled. Valid from Deep-Sleep.
26 PWM9	PWM9 Interrupt Wake-up 0b - Wake-up disabled. 1b - Wake-up enabled. Valid from Deep-Sleep and.
25 PWM8	PWM8 Interrupt Wake-up 0b - Wake-up disabled. 1b - Wake-up enabled. Valid from Deep-Sleep.

Table continues on the next page...

Table continued from the previous page...

Field	Function
24 PWM7	PWM7 interrupt wake-up. 0b - Wake-up disabled. 1b - Wake-up enabled. Valid from Deep-Sleep.
23 PWM6	PWM6 Interrupt Wake-up 0b - Wake-up disabled. 1b - Wake-up enabled. Valid from Deep-Sleep.
22 PWM5	PWM5 Interrupt Wake-up 0b - Wake-up disabled. 1b - Wake-up enabled. Valid from Deep-Sleep.
21 PWM4	PWM4 Interrupt Wake-up 0b - Wake-up disabled. 1b - Wake-up enabled. Valid from Deep-Sleep.
20 PWM3	PWM3 Interrupt Wake-up 0b - Wake-up disabled. 1b - Wake-up enabled. Valid from Deep-Sleep.
19 PWM2	PWM2 Interrupt Wake-up 0b - Wake-up disabled. 1b - Wake-up enabled. Valid from Deep-Sleep.
18 PWM1	PWM1 Interrupt Wake-up 0b - Wake-up disabled. 1b - Wake-up enabled. Valid from Deep-Sleep.
17 PWM0	PWM0 Interrupt Wake-up 0b - Wake-up disabled. 1b - Wake-up enabled. Valid from Deep-Sleep.
16 SPI1	SPI1 Interrupt Wake-up 0b - Wake-up disabled. 1b - Wake-up enabled. Valid from Deep-Sleep.
15 SPI0	SPI0 Interrupt Wake-up 0b - Wake-up disabled. 1b - Wake-up enabled. Valid from Deep-Sleep and Power-down.

Table continues on the next page...

Table continued from the previous page...

Field	Function
14 I2C1	I2C1 Interrupt Wake-up 0b - Wake-up disabled. 1b - Wake-up enabled. Valid from Deep-Sleep.
13 I2C0	I2C0 Interrupt Wake-up 0b - Wake-up disabled. 1b - Wake-up enabled. Valid from Deep-Sleep and Power-down.
12 USART1	USART1 Interrupt Wake-up 0b - Wake-up disabled. 1b - Wake-up enabled. Valid from Deep-Sleep.
11 USART0	USART0 Interrupt Wake-up 0b - Wake-up disabled. 1b - Wake-up enabled. Valid from Deep-Sleep and Power-down.
10 TIMER1	Counter/Timer1 Interrupt Wake-up 0b - Wake-up disabled. 1b - Wake-up enabled. Valid from Deep-Sleep.
9 TIMER0	Counter/Timer0 Interrupt Wake-up 0b - Wake-up disabled. 1b - Wake-up enabled. Valid from Deep-Sleep.
8 SPIFI	SPI Flash Interface (SPIFI) Interrupt Wake-up 0b - Wake-up disabled. 1b - Wake-up enabled. Valid from Deep-Sleep.
7 PINT3	Pattern Interupt 3 (PINT3) Interrupt Wake-up 0b - Wake-up disabled. 1b - Wake-up enabled. Valid from Deep-Sleep.
6 PINT2	Pattern Interupt 2 (PINT2) Interrupt Wake-up 0b - Wake-up disabled. 1b - Wake-up enabled. Valid from Deep-Sleep.
5 PINT1	Pattern Interupt 1 (PINT1) Interrupt Wake-up 0b - Wake-up disabled. 1b - Wake-up enabled. Valid from Deep-Sleep.

Table continues on the next page...

Table continued from the previous page...

Field	Function
4 PINT0	Pattern Interupt 0 (PINT0) Interrupt Wake-up 0b - Wake-up disabled. 1b - Wake-up enabled. Valid from Deep-Sleep.
3 IRBLASTER	Infra Red Blaster Interrupt Wake-up 0b - Wake-up disabled. 1b - Wake-up enabled. Valid from Deep-Sleep.
2 GINT	Group Interrupt 0 (GINT0) Interrupt Wake-up 0b - Wake-up disabled. 1b - Wake-up enabled. Valid from Deep-Sleep.
1 DMA	DMA Interrupt Wake-up DMA Operation in Deep-Sleep and Power-down are not supported. Leave set to 0.
0 WDT_BOD	WWDT and BOD Interrupt Wake-up 0b - Wake-up disabled. 1b - Wake-up enabled. Valid from Deep-Sleep and Power-down.

2.1.53 Start Logic 1 Wake-up Enable Register (STARTER1)

Offset

Register	Offset
STARTER1	684h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	GPIO	Reserved							Reserved		Reserved				WAKE_UP...	WAKE_UP...
Reset	0	u	u	u	u	u	u	u	u	u	u	u	u	u	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ANA_COMP	ISO7816	RFP_AGC	RFP_TMU	ZIGBEE_...	ZIGBEE_...	Reserved					HWVAD	DMIC	ADC_THC...	Reserved	ADC_SEQA
Reset	0	0	0	0	0	0	u	u	u	u	u	0	0	0	0	0

Fields

Field	Function
31 GPIO	GPIO Interrupt Wake-up 0b - Wake-up disabled. 1b - Wake-up enabled. Set this bit to allow GPIO or NTAG_INT to cause a wake-up in Deep-Sleep and Power-down mode.
30-24 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
23-22 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
21-18 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
17 WAKE_UP_TIMER1	Wake-up Timer1 Interrupt Wake-up 0b - Wake-up disabled. 1b - Wake-up enabled. Valid from Deep-Sleep and Power-down.
16 WAKE_UP_TIMER0	Wake-up Timer0 Interrupt Wake-up 0b - Wake-up disabled. 1b - Wake-up enabled. Valid from Deep-Sleep and Powerdown.
15 ANA_COMP	Analog Comparator Interrupt Wake-up 0b - Wake-up disabled. 1b - Wake-up enabled. Valid from Deep-Sleep and Power-down.
14 ISO7816	ISO7816 Smart Card interface Interrupt Wake-up 0b - Wake-up disabled. 1b - Wake-up enabled. Valid from Deep-Sleep.
13 RFP_AGC	Radio Control AGC (RFP AGC) Interrupt Wake-up 0b - Wake-up disabled. 1b - Wake-up enabled. Valid from Deep-Sleep.
12 RFP_TMU	Radio Controller Timing Controller (RFP TMU) Interrupt Wake-up 0b - Wake-up disabled. 1b - Wake-up enabled. Valid from Deep-Sleep.

Table continues on the next page...

Table continued from the previous page...

Field	Function
11 ZIGBEE_MODEM	Zigbee/Thread Modem Interrupt Wake-up 0b - Wake-up disabled. 1b - Wake-up enabled. Valid from Deep-Sleep.
10 ZIGBEE_MAC	Zigbee/Thread MAC Interrupt Wake-up 0b - Wake-up disabled. 1b - Wake-up enabled. Valid from Deep-Sleep.
9-5 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
4 HWVAD	Hardware Voice Activity Detector (HWVAD) Interrupt Wake-up 0b - Wake-up disabled. 1b - Wake-up enabled. Valid from Deep-Sleep.
3 DMIC	Digital Microphone (DMIC) Interrupt Wake-up 0b - Wake-up disabled. 1b - Wake-up enabled. Valid from Deep-Sleep.
2 ADC_THCMP_OVR	ADC threshold and error Interrupt Wake-up 0b - Wake-up disabled. 1b - Wake-up enabled. Valid from Deep-Sleep.
1 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
0 ADC_SEQA	ADC Sequence A Interrupt Wake-up 0b - Wake-up disabled. 1b - Wake-up enabled. Valid from Deep-Sleep.

2.1.54 Set STARTER0 Register (STARTERSET0)

Offset

Register	Offset
STARTERSET0	6A0h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	Reserved	NFCTAG_...	RTC_SET	I2C2_SET	PWM10_S...	PWM9_SET	PWM8_SET	PWM7_SET	PWM6_SET	PWM5_SET	PWM4_SET	PWM3_SET	PWM2_SET	PWM1_SET	PWM0_SET	SPI1_SET
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	SPI0_SET	I2C1_SET	I2C0_SET	USART1_...	USART0_...	TIMER1_...	TIMER0_...	SPIFI_S...	PINT3_S...	PINT2_S...	PINT1_S...	PINT0_S...	IRBLAST...	GINT_SET	DMA_SET	WDT_BOD...
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
30 NFCTAG_SET	NFCTAG Set Writing one to this bit sets the corresponding bit in the STARTER0 register
29 RTC_SET	RTC Set Writing one to this bit sets the corresponding bit in the STARTER0 register
28 I2C2_SET	I2C2 Set Writing one to this bit sets the corresponding bit in the STARTER0 register
27 PWM10_SET	PWM10 Set Writing one to this bit sets the corresponding bit in the STARTER0 register
26 PWM9_SET	PWM9 Set Writing one to this bit sets the corresponding bit in the STARTER0 register
25 PWM8_SET	PWM8 Set Writing one to this bit sets the corresponding bit in the STARTER0 register
24 PWM7_SET	PWM7 Set Writing one to this bit sets the corresponding bit in the STARTER0 register
23 PWM6_SET	PWM6 Set Writing one to this bit sets the corresponding bit in the STARTER0 register

Table continues on the next page...

Table continued from the previous page...

Field	Function
22 PWM5_SET	PWM5 Set Writing one to this bit sets the corresponding bit in the STARTER0 register
21 PWM4_SET	PWM4 Set Writing one to this bit sets the corresponding bit in the STARTER0 register
20 PWM3_SET	PWM3 Set Writing one to this bit sets the corresponding bit in the STARTER0 register
19 PWM2_SET	PWM2 Set Writing one to this bit sets the corresponding bit in the STARTER0 register
18 PWM1_SET	PWM1 Set Writing one to this bit sets the corresponding bit in the STARTER0 register
17 PWM0_SET	PWM0 Set Writing one to this bit sets the corresponding bit in the STARTER0 register
16 SPI1_SET	SPI1 Set Writing one to this bit sets the corresponding bit in the STARTER0 register
15 SPI0_SET	SPI0 Set Writing one to this bit sets the corresponding bit in the STARTER0 register
14 I2C1_SET	I2C1 Set Writing one to this bit sets the corresponding bit in the STARTER0 register
13 I2C0_SET	I2C0 Set Writing one to this bit sets the corresponding bit in the STARTER0 register
12 USART1_SET	USART1 Set Writing one to this bit sets the corresponding bit in the STARTER0 register
11 USART0_SET	USART0 Set Writing one to this bit sets the corresponding bit in the STARTER0 register
10 TIMER1_SET	TIMER1 Set Writing one to this bit sets the corresponding bit in the STARTER0 register
9 TIMER0_SET	TIMER0 Set Writing one to this bit sets the corresponding bit in the STARTER0 register

Table continues on the next page...

Table continued from the previous page...

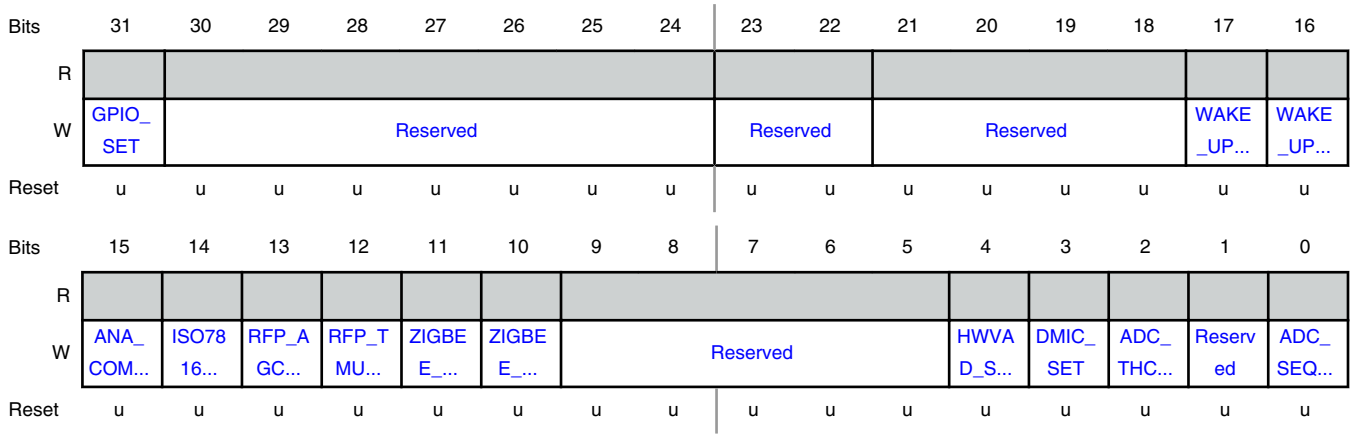
Field	Function
8 SPIFI_SET	SPIFI Set Writing one to this bit sets the corresponding bit in the STARTER0 register
7 PINT3_SET	PINT3 Set Writing one to this bit sets the corresponding bit in the STARTER0 register
6 PINT2_SET	PINT2 Set Writing one to this bit sets the corresponding bit in the STARTER0 register
5 PINT1_SET	PINT1 Set Writing one to this bit sets the corresponding bit in the STARTER0 register
4 PINT0_SET	PINT0 Set Writing one to this bit sets the corresponding bit in the STARTER0 register
3 IRBLASTER_SET	IRBLASTER Set Writing one to this bit sets the corresponding bit in the STARTER0 register
2 GINT_SET	GINT Set Writing one to this bit sets the corresponding bit in the STARTER0 register
1 DMA_SET	DMA Set Writing one to this bit sets the corresponding bit in the STARTER0 register
0 WDT_BOD_SET	WDT_BOD Set Writing one to this bit sets the corresponding bit in the STARTER0 register

2.1.55 Set STARTER1 Register (STARTERSET1)

Offset

Register	Offset
STARTERSET1	6A4h

Diagram



Fields

Field	Function
31 GPIO_SET	GPIO Set Writing one to this bit sets the corresponding bit in the STARTER1 register
30-24 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
23-22 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
21-18 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
17 WAKE_UP_TIMER1_SET	WAKE_UP_TIMER1 Set Writing one to this bit sets the corresponding bit in the STARTER1 register
16 WAKE_UP_TIMER0_SET	WAKE_UP_TIMER0 Set Writing one to this bit sets the corresponding bit in the STARTER1 register
15 ANA_COMP_SET	ANA_COMP Set Writing one to this bit sets the corresponding bit in the STARTER1 register
14 ISO7816_SET	ISO7816 Set Writing one to this bit sets the corresponding bit in the STARTER1 register

Table continues on the next page...

Table continued from the previous page...

Field	Function
13 RFP_AGC_SET	RFP_AGC Set Writing one to this bit sets the corresponding bit in the STARTER1 register
12 RFP_TMU_SET	RFP_TMU Set Writing one to this bit sets the corresponding bit in the STARTER1 register
11 ZIGBEE_MODEM_SET	ZIGBEE_MODEM Set Writing one to this bit sets the corresponding bit in the STARTER1 register
10 ZIGBEE_MAC_SET	ZIGBEE_MAC Set Writing one to this bit sets the corresponding bit in the STARTER1 register
9-5 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
4 HWVAD_SET	HWVAD Set Writing one to this bit sets the corresponding bit in the STARTER1 register
3 DMIC_SET	DMIC Set Writing one to this bit sets the corresponding bit in the STARTER1 register
2 ADC_THCMP_OVR_SET	ADC_THCMP_OVR Set Writing one to this bit sets the corresponding bit in the STARTER1 register
1 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
0 ADC_SEQA_SET	ADC_SEQA Set Writing one to this bit sets the corresponding bit in the STARTER1 register

2.1.56 Clear STARTER0 Bit Register (STARTERCLR0)

Offset

Register	Offset
STARTERCLR0	6C0h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	Reserved	NFCTAG_...	RTC_CLR	I2C2_CLR	PWM10_C...	PWM9_CLR	PWM8_CLR	PWM7_CLR	PWM6_CLR	PWM5_CLR	PWM4_CLR	PWM3_CLR	PWM2_CLR	PWM1_CLR	PWM0_CLR	SPI1_CLR
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	SPI0_CLR	I2C1_CLR	I2C0_CLR	USART1_...	USART0_...	TIMER1_...	TIMER0_...	SPIFI_C...	PINT3_C...	PINT2_C...	PINT1_C...	PINT0_C...	IRBLAST...	GINT_CLR	DMA_CLR	WDT_BOD...
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
30 NFCTAG_CLR	NFCTAG Clear Writing one to this bit clears the corresponding bit in the STARTER0 register
29 RTC_CLR	RTC Clear Writing one to this bit clears the corresponding bit in the STARTER0 register
28 I2C2_CLR	I2C2 Clear Writing one to this bit clears the corresponding bit in the STARTER0 register
27 PWM10_CLR	PWM10 Clear Writing one to this bit clears the corresponding bit in the STARTER0 register
26 PWM9_CLR	PWM9 Clear Writing one to this bit clears the corresponding bit in the STARTER0 register
25 PWM8_CLR	PWM8 Clear Writing one to this bit clears the corresponding bit in the STARTER0 register
24 PWM7_CLR	PWM7 Clear Writing one to this bit clears the corresponding bit in the STARTER0 register
23 PWM6_CLR	PWM6 Clear Writing one to this bit clears the corresponding bit in the STARTER0 register

Table continues on the next page...

Table continued from the previous page...

Field	Function
22 PWM5_CLR	PWM5 Clear Writing one to this bit clears the corresponding bit in the STARTER0 register
21 PWM4_CLR	PWM4 Clear Writing one to this bit clears the corresponding bit in the STARTER0 register
20 PWM3_CLR	PWM3 Clear Writing one to this bit clears the corresponding bit in the STARTER0 register
19 PWM2_CLR	PWM2 Clear Writing one to this bit clears the corresponding bit in the STARTER0 register
18 PWM1_CLR	PWM1 Clear Writing one to this bit clears the corresponding bit in the STARTER0 register
17 PWM0_CLR	PWM0 Clear Writing one to this bit clears the corresponding bit in the STARTER0 register
16 SPI1_CLR	SPI1 Clear Writing one to this bit clears the corresponding bit in the STARTER0 register
15 SPI0_CLR	SPI0 Clear Writing one to this bit clears the corresponding bit in the STARTER0 register
14 I2C1_CLR	I2C1 Clear Writing one to this bit clears the corresponding bit in the STARTER0 register
13 I2C0_CLR	I2C0 Clear Writing one to this bit clears the corresponding bit in the STARTER0 register
12 USART1_CLR	USART1 Clear Writing one to this bit clears the corresponding bit in the STARTER0 register
11 USART0_CLR	USART0 Clear Writing one to this bit clears the corresponding bit in the STARTER0 register
10 TIMER1_CLR	TIMER1 Clear Writing one to this bit clears the corresponding bit in the STARTER0 register
9 TIMER0_CLR	TIMER0 Clear Writing one to this bit clears the corresponding bit in the STARTER0 register

Table continues on the next page...

Table continued from the previous page...

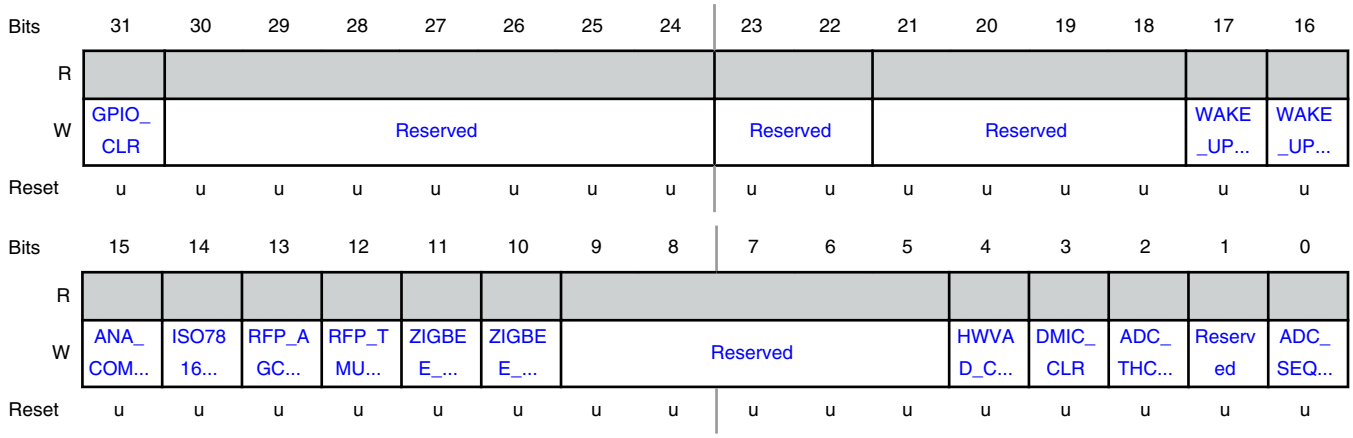
Field	Function
8 SPIFI_CLR	SPIFI Clear Writing one to this bit clears the corresponding bit in the STARTER0 register
7 PINT3_CLR	PINT3 Clear Writing one to this bit clears the corresponding bit in the STARTER0 register
6 PINT2_CLR	PINT2 Clear Writing one to this bit clears the corresponding bit in the STARTER0 register
5 PINT1_CLR	PINT1 Clear Writing one to this bit clears the corresponding bit in the STARTER0 register
4 PINT0_CLR	PINT0 Clear Writing one to this bit clears the corresponding bit in the STARTER0 register
3 IRBLASTER_CLR	IRBLASTER Clear Writing one to this bit clears the corresponding bit in the STARTER0 register
2 GINT_CLR	GINT Clear Writing one to this bit clears the corresponding bit in the STARTER0 register
1 DMA_CLR	DMA Clear Writing one to this bit clears the corresponding bit in the STARTER0 register
0 WDT_BOD_CLR	WDT_BOD Clear Writing one to this bit clears the corresponding bit in the STARTER0 register

2.1.57 Clear STARTER1 Bits Register (STARTERCLR1)

Offset

Register	Offset
STARTERCLR1	6C4h

Diagram



Fields

Field	Function
31 GPIO_CLR	GPIO Clear Writing one to this bit clears the corresponding bit in the STARTER1 register
30-24 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
23-22 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
21-18 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
17 WAKE_UP_TIMER1_CLR	WAKE_UP_TIMER1 Clear Writing one to this bit clears the corresponding bit in the STARTER1 register
16 WAKE_UP_TIMER0_CLR	WAKE_UP_TIMER0 Clear Writing one to this bit clears the corresponding bit in the STARTER1 register
15 ANA_COMP_CLR	ANA_COMP Clear Writing one to this bit clears the corresponding bit in the STARTER1 register
14 ISO7816_CLR	ISO7816 Clear Writing one to this bit clears the corresponding bit in the STARTER1 register

Table continues on the next page...

Table continued from the previous page...

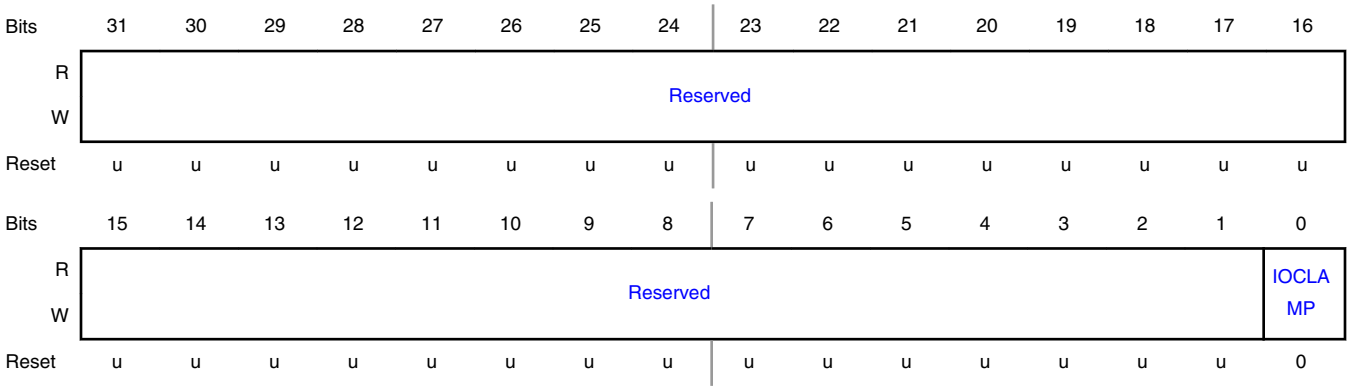
Field	Function
13 RFP_AGC_CLR	RFP_AGC Clear Writing one to this bit clears the corresponding bit in the STARTER1 register
12 RFP_TMU_CLR	RFP_TMU Clear Writing one to this bit clears the corresponding bit in the STARTER1 register
11 ZIGBEE_MODEM_CLR	ZIGBEE_MODEM Clear Writing one to this bit clears the corresponding bit in the STARTER1 register
10 ZIGBEE_MAC_CLR	ZIGBEE_MAC Clear Writing one to this bit clears the corresponding bit in the STARTER1 register
9-5 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
4 HWVAD_CLR	HWVAD Clear Writing one to this bit clears the corresponding bit in the STARTER1 register
3 DMIC_CLR	DMIC Clear Writing one to this bit clears the corresponding bit in the STARTER1 register
2 ADC_THCMP_OVR_CLR	ADC_THCMP_OVR Clear Writing one to this bit clears the corresponding bit in the STARTER1 register
1 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
0 ADC_SEQA_CLR	ADC_SEQA Clear Writing one to this bit clears the corresponding bit in the STARTER1 register

2.1.58 I/O Retention Control Register (RETENTIONCTRL)

Offset

Register	Offset
RETENTIONCTRL	708h

Diagram



Fields

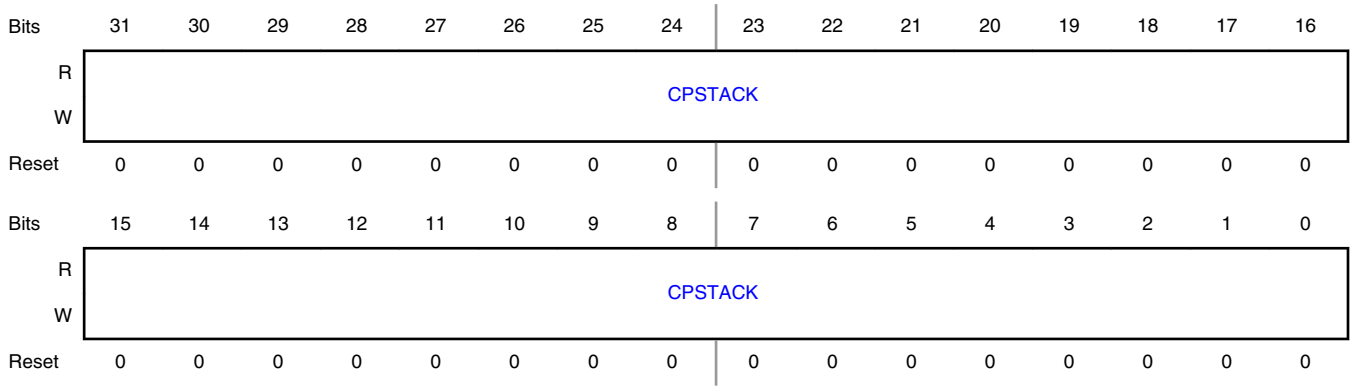
Field	Function
31-1 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
0 IOCLAMP	I/O Clamps Global control of activation of I/O clamps allows IOs to hold a value during a power mode cycle. Use enable before the power-down and disable after a wake-up. Note that each I/O clamp must also be enabled/ disabled individually in IOCON module and also that for an I2C IO cell it must be in GPIO mode for the clamping to work. 0b - I/O clamp is disabled. 1b - I/O clamp is enabled.

2.1.59 CPSTACK (CPSTACK)

Offset

Register	Offset
CPSTACK	808h

Diagram



Fields

Field	Function
31-0 CPSTACK	CPSTACK This field should only be modified by the APIs.

2.1.60 Analog Interrupt Control Rgister (ANACTRL_CTRL)

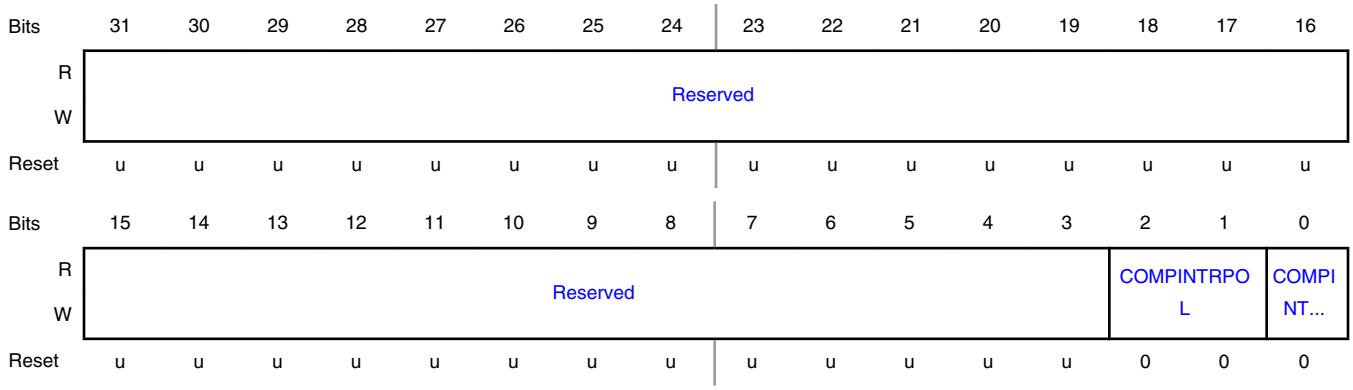
Offset

Register	Offset
ANACTRL_CTRL	A00h

Function

It requires AHBCLKCTRL0[ANA_INT_CTRL] to be set.

Diagram



Fields

Field	Function
31-3 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
2-1 COMPINTRPOL	Analog Comparator Interrupt Polarity 00b - When COMPINTRLVL = 0 (edge sensitive): rising edge; When COMPINTRLVL = 1 (level sensitive): Low level 01b - When COMPINTRLVL = 0 (edge sensitive): falling edge; When COMPINTRLVL = 1 (level sensitive): Low level 10b - When COMPINTRLVL = 0 (edge sensitive): both edges (rising and falling); When COMPINTRLVL = 1 (level sensitive): High level 11b - When COMPINTRLVL = 0 (edge sensitive): both edges (rising and falling); When COMPINTRLVL = 1 (level sensitive): High level
0 COMPINTRLVL	Analog Comparator Interrupt Type 0b - Analog Comparator interrupt is edge sensitive. 1b - Analog Comparator interrupt is level sensitive.

2.1.61 Analog Modules Outputs Current Values Register (ANACTRL_VAL)

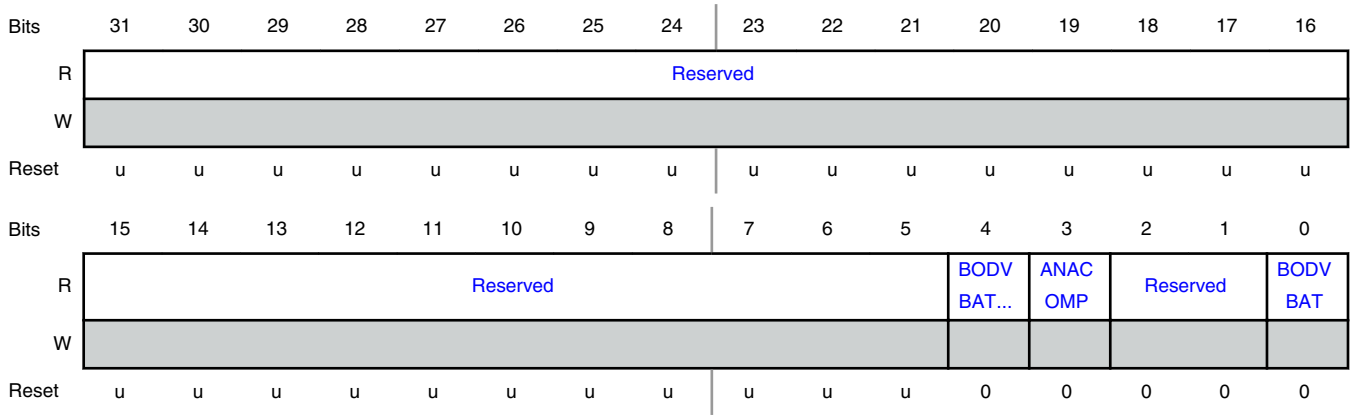
Offset

Register	Offset
ANACTRL_VAL	A04h

Function

Analog modules (BOD and Analog Comparator) outputs current values (BOD 'Power OK' and Analog comparator out). It requires AHBCLKCTRL0[ANA_INT_CTRL] to be set.

Diagram



Fields

Field	Function
31-5 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
4 BODVBATHIGH	Not(BOD VBAT). Inverse of BODVBAT. 0b - The voltage on VBAT is OK, ie. above the BOD threshold. 1b - The voltage on VBAT is not OK, ie. below the BOD threshold.
3 ANACOMP	Analog Comparator Status 0b - Comparator in 0 < in 1 1b - Comparator in 0 > in 1
2-1 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
0 BODVBAT	BOD VBAT Status 0b - Power not OK 1b - Power OK

2.1.62 Analog Status Register (ANACTRL_STAT)

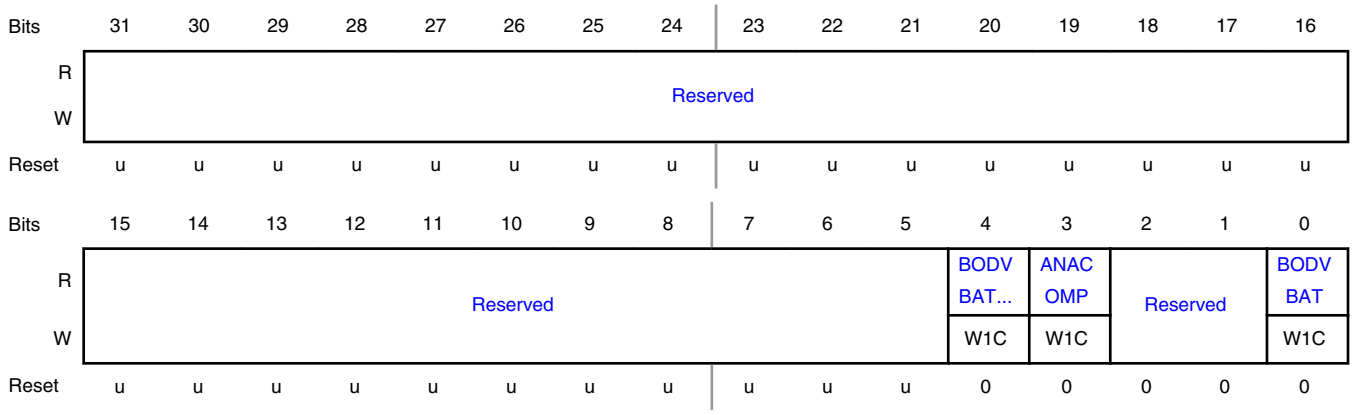
Offset

Register	Offset
ANACTRL_STAT	A08h

Function

Analog modules (BOD and Analog Comparator) interrupt status. It requires AHBCLKCTRL0[ANA_INT_CTRL] to be set.

Diagram



Fields

Field	Function
31-5 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
4 BODVBATHIGH	NOT(BOD VBAT) Interrupt Status 0b - No Interrupt pending. 1b - Interrupt pending. This will be set when VBAT increases and crosses the BOD VBAT threshold..
3 ANACOMP	Analog Comparator Interrupt Status 0b - No interrupt pending. 1b - Interrupt pending.
2-1 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
0 BODVBAT	BOD VBAT Interrupt Status 0b - No interrupt pending. 1b - Interrupt pending.

2.1.63 Analog Modules Interrupt Enable Read and Set Register (ANACTRL_INTENSET)

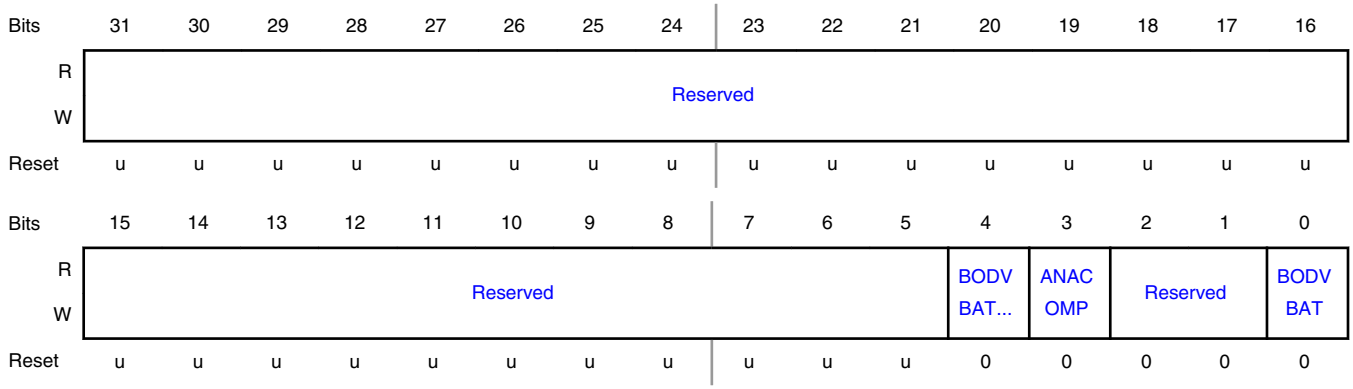
Offset

Register	Offset
ANACTRL_INTENSET	A0Ch

Function

Analog modules (BOD and Analog Comparator) Interrupt Enable Read and Set register. Read value indicates which interrupts are enabled. Writing ones sets the corresponding interrupt enable bits. Note, interrupt enable bits are cleared using ANACTRL_INTENCLR. It requires AHBCLKCTRL0[ANA_INT_CTRL] to be set to use this register.

Diagram



Fields

Field	Function
31-5 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
4 BODVBATHIGH	BODVBATHIGH Interrupt Enable Read and Set
3 ANACOMP	Analog Comparator Interrupt Enable Read and Set
2-1 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
0 BODVBAT	BOD VBAT Interrupt Enable Read and Set

2.1.64 Analog Modules Interrupt Enable Clear Register (ANACTRL_INTENCLR)

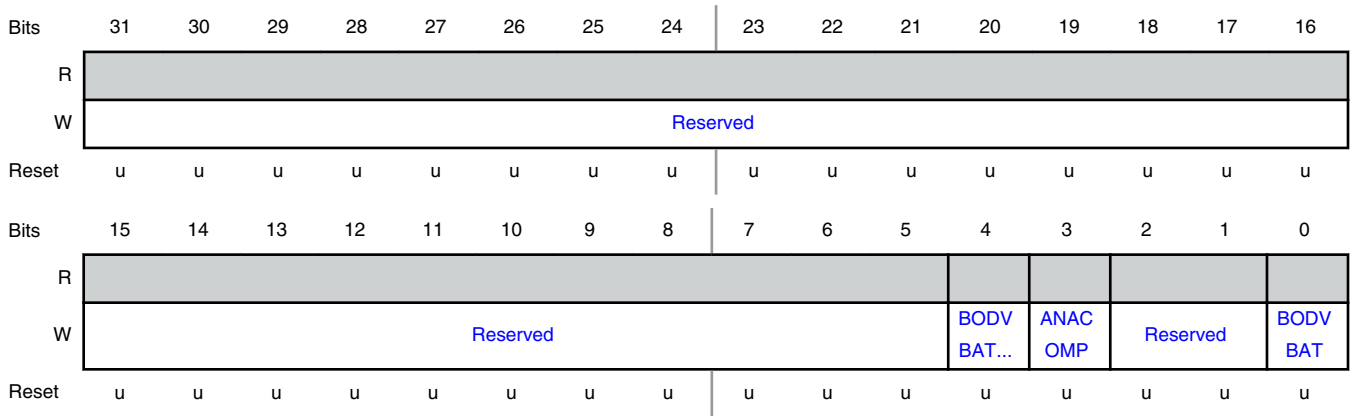
Offset

Register	Offset
ANACTRL_INTENCLR	A10h

Function

Analog modules (BOD and Analog Comparator) Interrupt Enable Clear register. Writing ones clears the corresponding interrupt enable bits. Note, interrupt enable bits are set in ANACTRL_INTENSET. It requires AHBCLKCTRL0.ANA_INT_CTRL to be set to use this register.

Diagram



Fields

Field	Function
31-5 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
4 BODVBATHIGH	BODVBATHIGH Interrupt Enable Clear

Table continues on the next page...

Table continued from the previous page...

Field	Function
3 ANACOMP	Analog Comparator Interrupt Enable Clear
2-1 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
0 BODVBAT	BOD VBAT Interrupt Enable Clear

2.1.65 Analog Modules Interrupt Status Register (ANACTRL_INTSTAT)

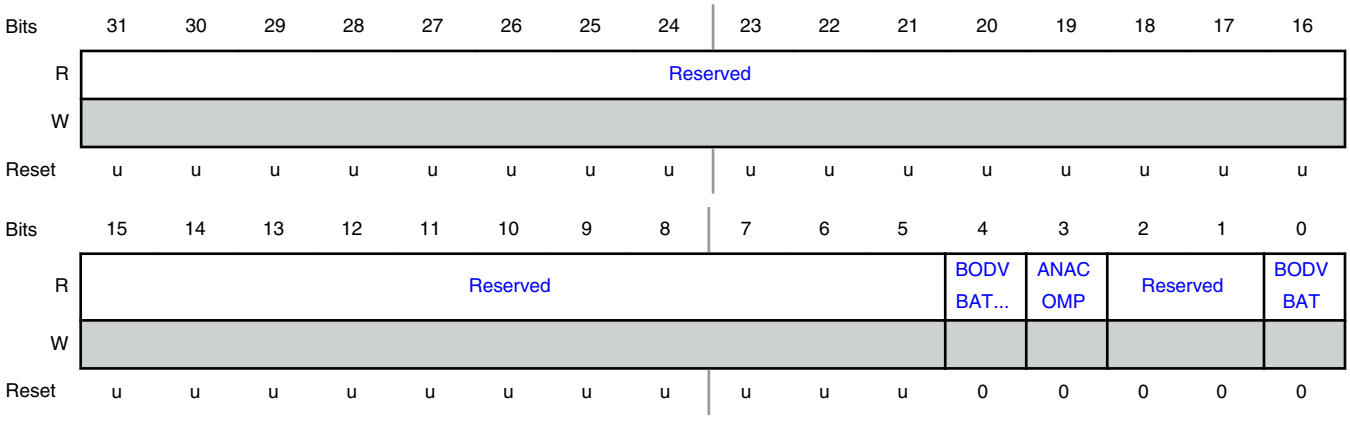
Offset

Register	Offset
ANACTRL_INTSTAT	A14h

Function

Analog modules (BOD and Analog Comparator) Interrupt Status register (masked with interrupt enable). It requires AHBCLKCTRL0[ANA_INT_CTRL] to be set to use this register. Interrupt status bit are cleared using ANACTRL_STAT.

Diagram



Fields

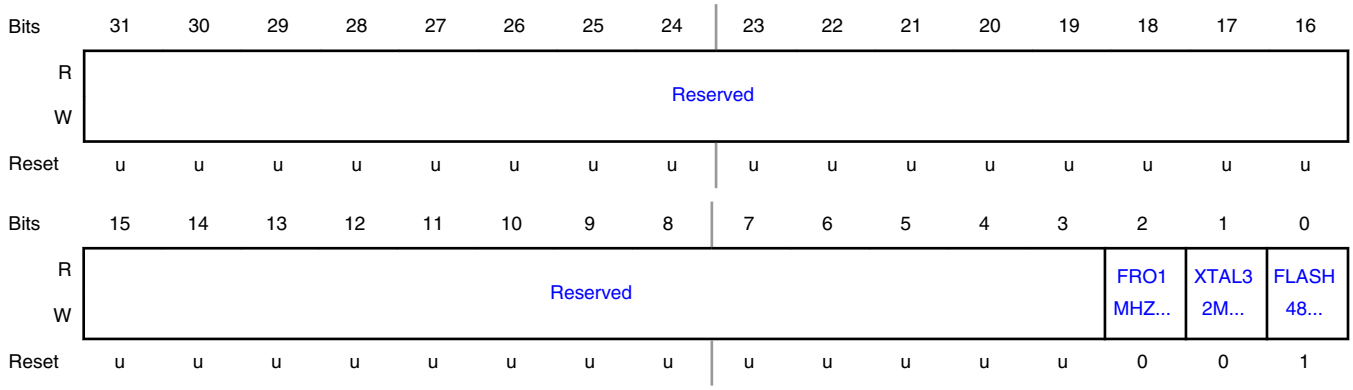
Field	Function
31-5 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
4 BODVBATHIGH	BODVBATHIGH Interrupt (after interrupt mask) Only set when ANACTRL_INTENSET[BODVBATHIGH] is enabled. 0b - No interrupt pending. 1b - Interrupt pending.
3 ANACOMP	Analog Comparator Interrupt (after interrupt mask) Only set when ANACTRL_INTENSET[ANACOMP] is enabled. 0b - No interrupt pending. 1b - Interrupt pending.
2-1 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
0 BODVBAT	BOD VBAT Interrupt (after interrupt mask) Only set when ANACTRL_INTENSET[BODVBAT] is enabled. 0b - No interrupt pending. 1b - Interrupt pending.

2.1.66 Various System Clock Control Register (CLOCK_CTRL)

Offset

Register	Offset
CLOCK_CTRL	A18h

Diagram



Fields

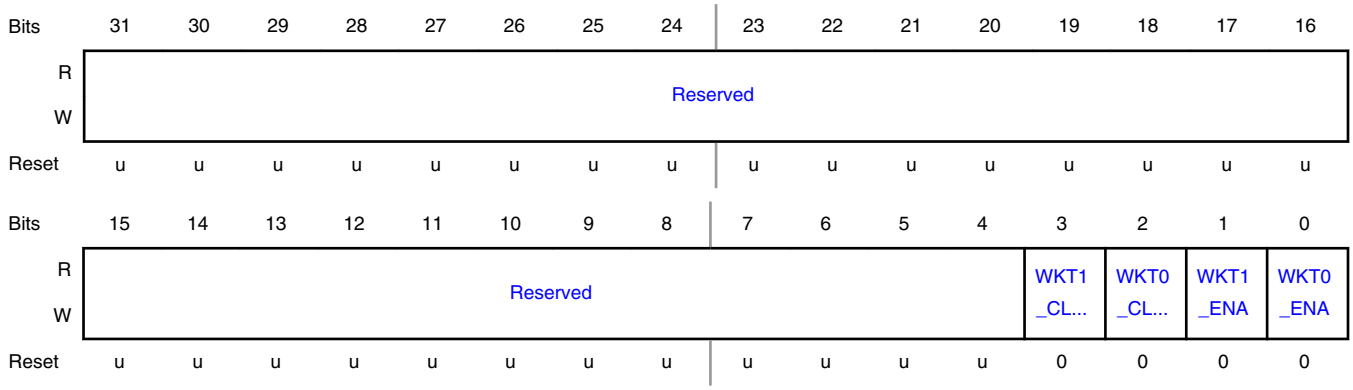
Field	Function
31-3 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
2 FRO1MHZ_FREQM_ENA	Enable FRO 1 MHz Clock for Frequency Measure Module 0b - Disabled. 1b - Enabled.
1 XTAL32MHZ_FREQM_ENA	Enable XTAL32MHz Clock for Frequency Measure Module 0b - Disabled. 1b - Enabled.
0 FLASH48MHZ_ENA	Enable Flash 48 MHz Clock 0b - Disabled. 1b - Enabled.

2.1.67 Wake-up Timers Control Register (WKT_CTRL)

Offset

Register	Offset
WKT_CTRL	A20h

Diagram



Fields

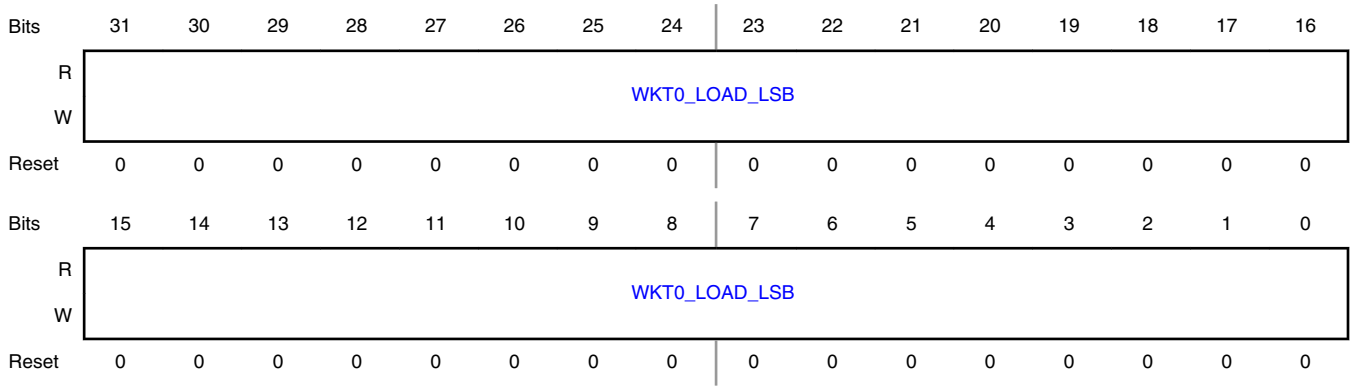
Field	Function
31-4 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
3 WKT1_CLK_ENA	Enable Wake-up Timer 1 Clock 0b - Disabled. 1b - Enabled.
2 WKT0_CLK_ENA	Enable Wake-up Timer 0 Clock 0b - Disabled. 1b - Enabled.
1 WKT1_ENA	Enable Wake-up Timer 1 0b - Disabled. 1b - Enabled.
0 WKT0_ENA	Enable Wake-up Timer 0 0b - Disabled. 1b - Enabled.

2.1.68 Wake-up Timer 0 Reload Value Least Significant Bits Resister (WKT_LOAD_WKT0_LSB)

Offset

Register	Offset
WKT_LOAD_WKT0_LSB	A24h

Diagram



Fields

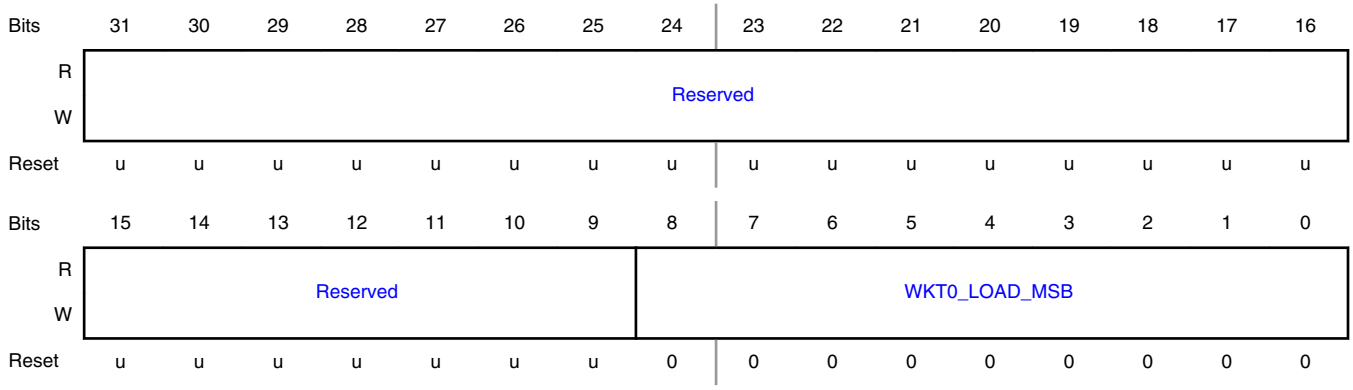
Field	Function
31-0 WKT0_LOAD_LSB	Wake-up Timer 0 Reload Value Least significant bits ([31:0]). Write when timer is not enabled

2.1.69 Wake-up Timer 0 Reload Value Most Significant Bits Register (WKT_LOAD_WKT0_MSB)

Offset

Register	Offset
WKT_LOAD_WKT0_MSB	A28h

Diagram



Fields

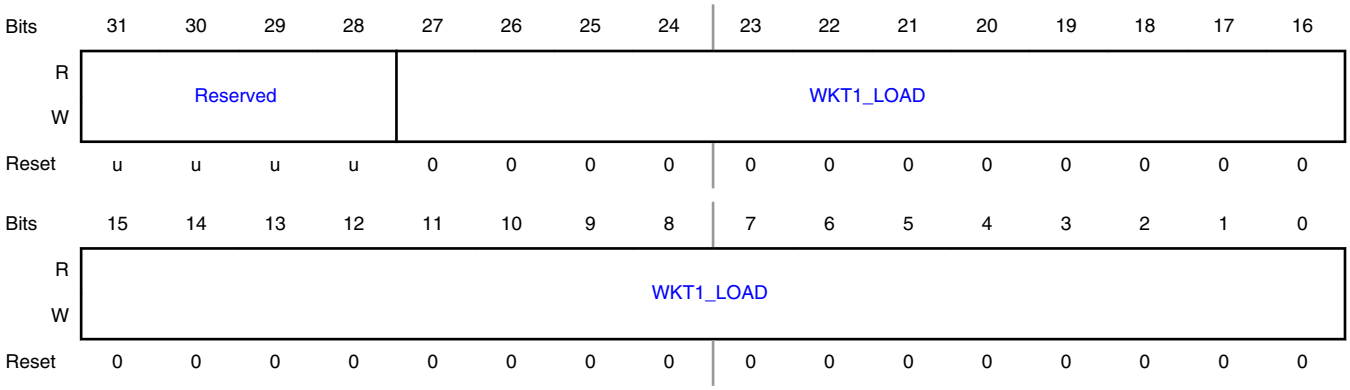
Field	Function
31-9 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
8-0 WKT0_LOAD_MSB	Wake-up Timer 0 Reload Value Most significant bits ([8:0]). Write when timer is not enabled

2.1.70 Wake-up Timer 1 Reload Value Register (WKT_LOAD_WKT1)

Offset

Register	Offset
WKT_LOAD_WKT1	A2Ch

Diagram



Fields

Field	Function
31-28 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
27-0 WKT1_LOAD	Wake-up Timer 1 Reload Value Write when timer is not enabled

2.1.71 Wake-up Timer 0 Current Value Least Significant Bits Register (WKT_VAL_WKT0_LSB)

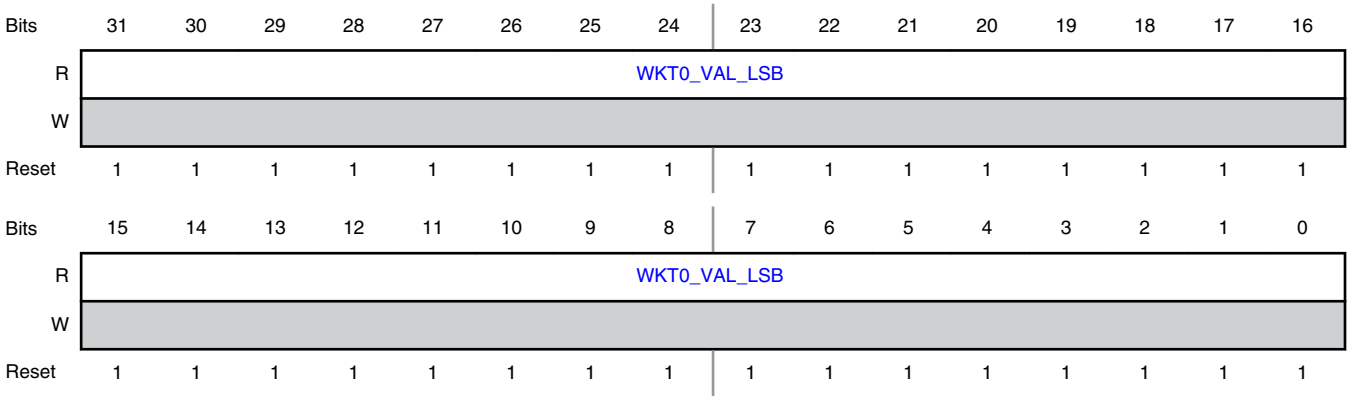
Offset

Register	Offset
WKT_VAL_WKT0_LSB	A30h

Function

Warning: reading is not reliable, read this register several times until you get a stable value.

Diagram



Fields

Field	Function
31-0 WKT0_VAL_LSB	Wake-up Timer 0 Value Least significant bits ([31:0]). Reread until stable value seen.

2.1.72 Wake-up Timer 0 Current Value Most Significant Bits Reister (WKT_VAL_WKT0_MSB)

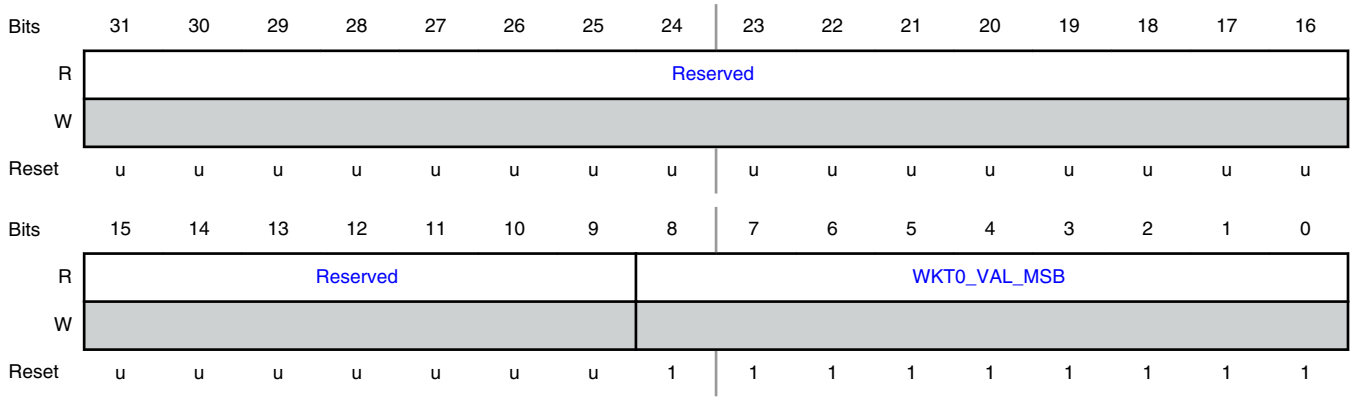
Offset

Register	Offset
WKT_VAL_WKT0_MSB	A34h

Function

Warning: reading is not reliable, read this register several times until you get a stable value.

Diagram



Fields

Field	Function
31-9 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
8-0 WKT0_VAL_MSB	Wake-up Timer 0 Value Most significant bits ([8:0]). Reread until stable value seen.

2.1.73 Wake-up Timer 1 Current Value Register (WKT_VAL_WKT1)

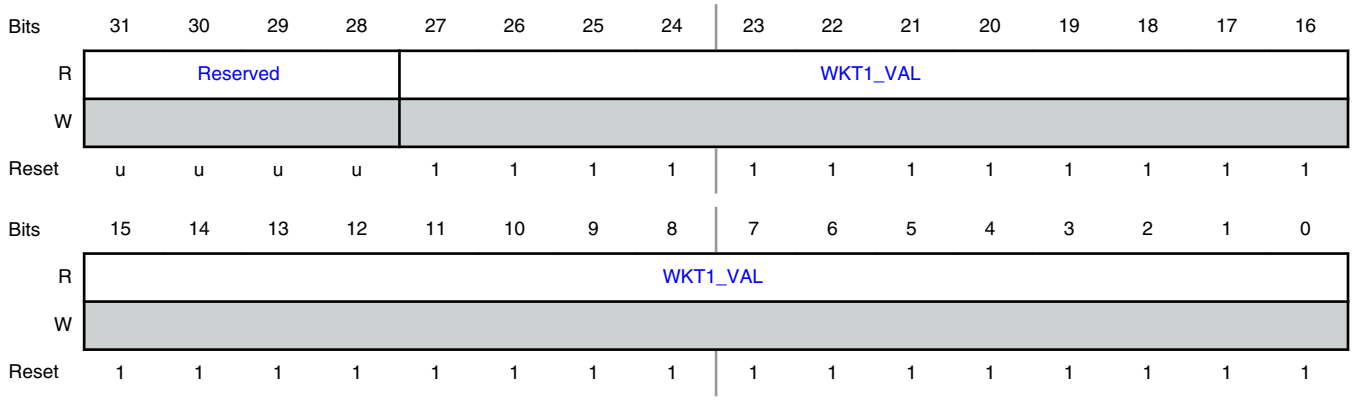
Offset

Register	Offset
WKT_VAL_WKT1	A38h

Function

Warning: reading is not reliable, read this register several times until you get a stable value.

Diagram



Fields

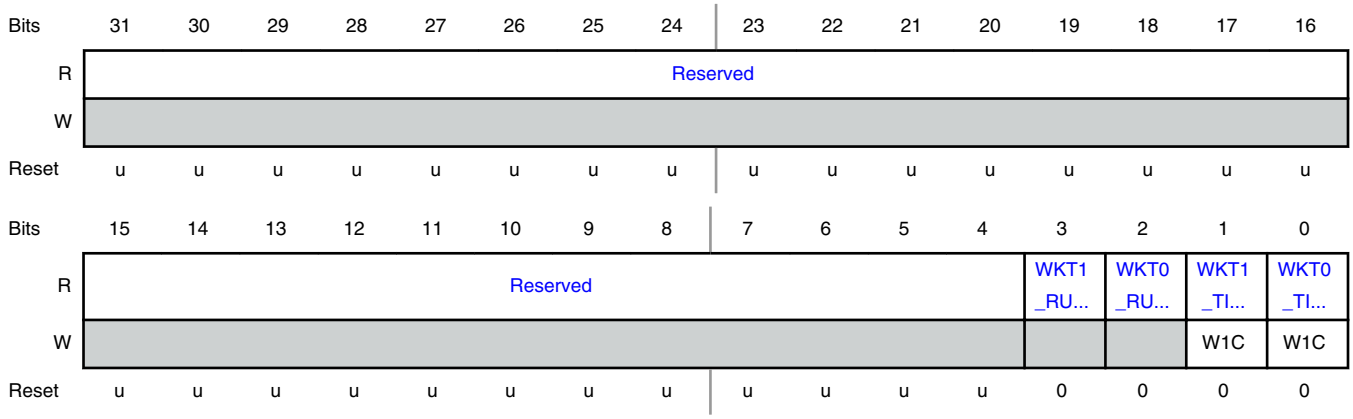
Field	Function
31-28 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
27-0 WKT1_VAL	Wake-up Timer 1 Value Reread until stable value seen.

2.1.74 Wake-up Timers Status Register (WKT_STAT)

Offset

Register	Offset
WKT_STAT	A3Ch

Diagram



Fields

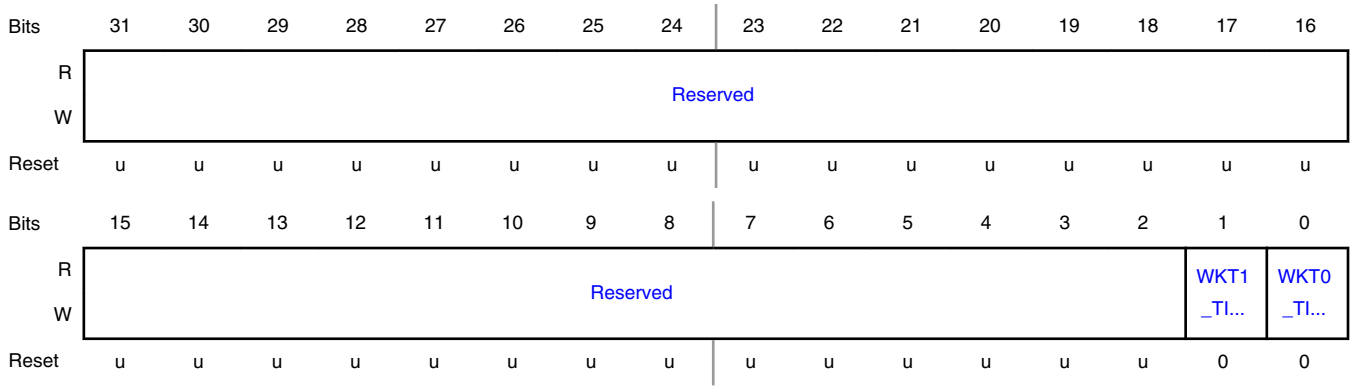
Field	Function
31-4 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
3 WKT1_RUNNING	Running Status of Wake-up Timer 1 0b - Not running. 1b - Running.
2 WKT0_RUNNING	Running Status of Wake-up Timer 0 0b - Not running. 1b - Running.
1 WKT1_TIMEOUT	Timeout Status of Wake-up Timer 1 0b - Timeout not reached. 1b - Timeout reached.
0 WKT0_TIMEOUT	Timeout Status of Wake-up Timer 0 0b - Timeout not reached. 1b - Timeout reached.

2.1.75 Interrupt Enable Read and Set Register (WKT_INTENSET)

Offset

Register	Offset
WKT_INTENSET	A40h

Diagram



Fields

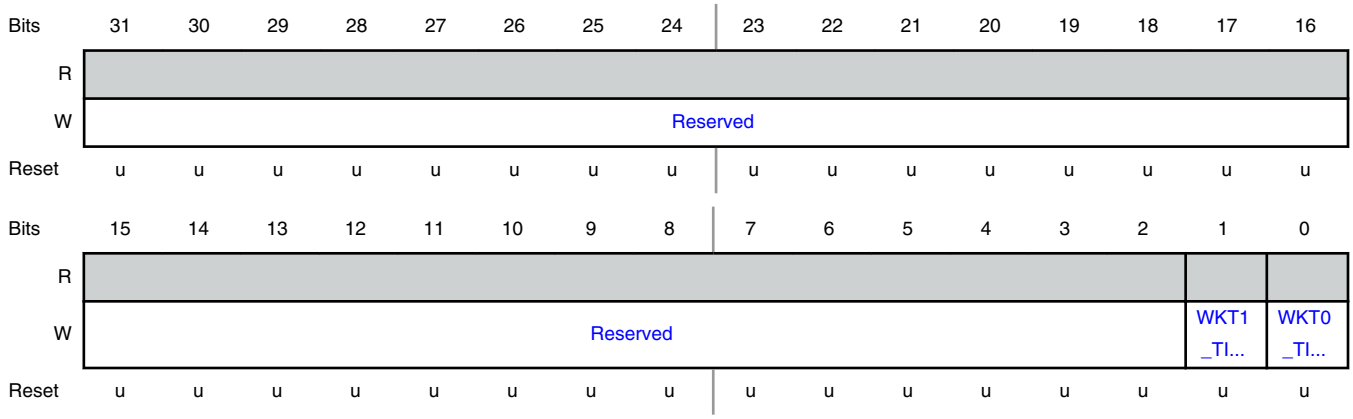
Field	Function
31-2 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
1 WKT1_TIMEOUT	Wake-up Timer 1 Timeout Interrupt Enable Read and Set Read value of '1' indicates that the interrupt is enabled. Set this bit to enable the interrupt. Use WKT_INTENCLR to disable the interrupt.
0 WKT0_TIMEOUT	Wake-up Timer 0 Timeout Interrupt Enable Read and Set Read value of '1' indicates that the interrupt is enabled. Set this bit to enable the interrupt. Use WKT_INTENCLR to disable the interrupt.

2.1.76 Interrupt Enable Clear Register (WKT_INTENCLR)

Offset

Register	Offset
WKT_INTENCLR	A44h

Diagram



Fields

Field	Function
31-2 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

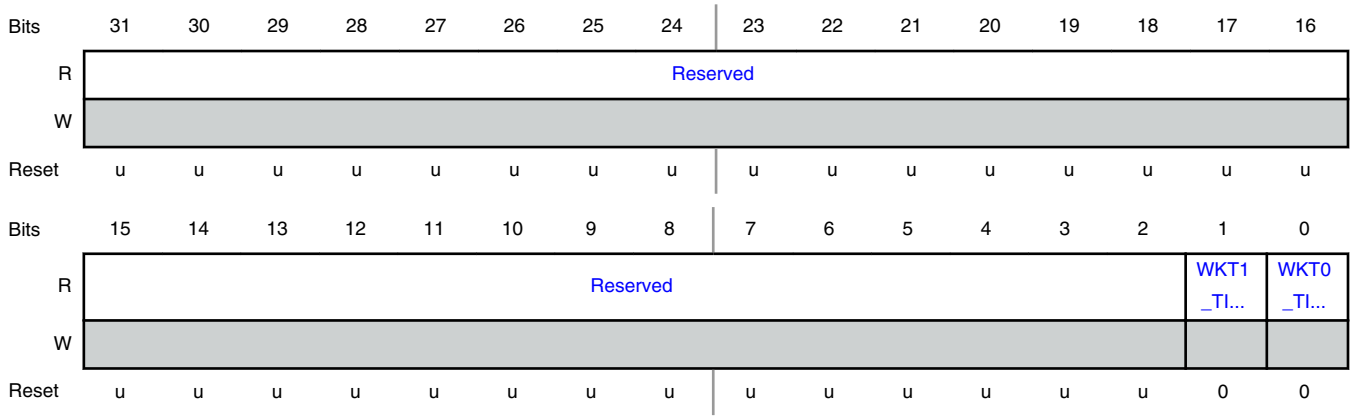
Field	Function
1 WKT1_TIMEOUT	Wake-up Timer 1 Timeout Interrupt Enable Clear Set this bit to disable the interrupt.
0 WKT0_TIMEOUT	Wake-up Timer 0 Timeout Interrupt Enable Clear Set this bit to disable the interrupt.

2.1.77 Interrupt Status Register (WKT_INTSTAT)

Offset

Register	Offset
WKT_INTSTAT	A48h

Diagram



Fields

Field	Function
31-2 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
1 WKT1_TIMEOUT	Wake-up Timer 1 Timeout Interrupt Only set when WKT1_TIMEOUT is enable in INTENSET 0b - No interrupt pending. 1b - Interrupt pending.
0 WKT0_TIMEOUT	Wake-up Timer 0 Timeout Interrupt Only set when WKT0_TIMEOUT is enable in INTENSET 0b - No interrupt pending. 1b - Interrupt pending.

2.1.78 GPIO_INT Synchonization First Stage Bypass Register (GPIOPSYNC)

Offset

Register	Offset
GPIOPSYNC	E08h

Diagram

Fields

Field	Function
31-1 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

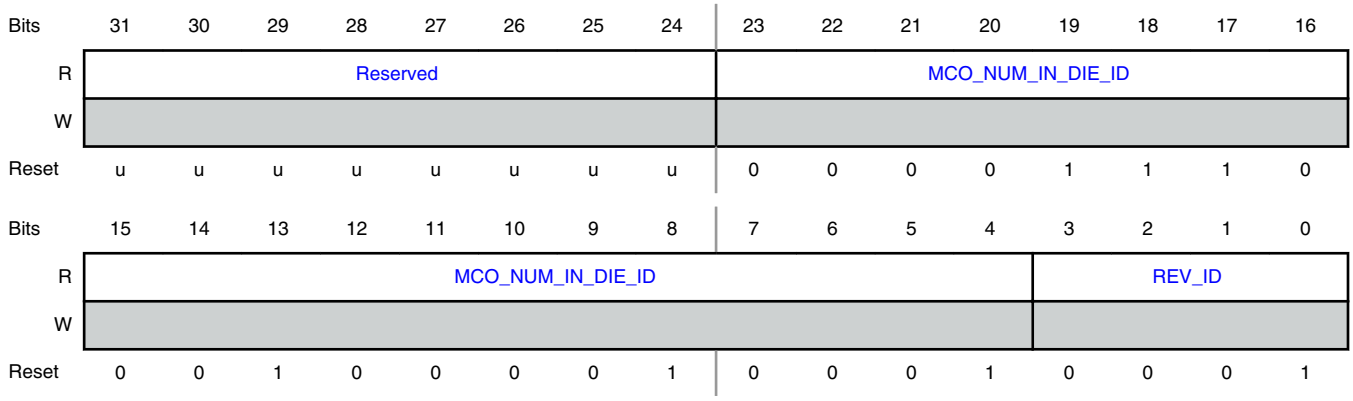
Field	Function
0 PSYNC	Enable Bypass of the first stage of synchonization inside GPIO_INT module

2.1.79 Chip Revision ID and Number Register (DIEID)

Offset

Register	Offset
DIEID	FB0h

Diagram



Fields

Field	Function
31-24 —	RESERVED Reserved. Always read as 0
23-4 MCO_NUM_IN_DIE_ID	Chip Number
3-0 REV_ID	Chip Revision ID

2.1.80 Test Access Security Code Register (CODESECURITY PROT)

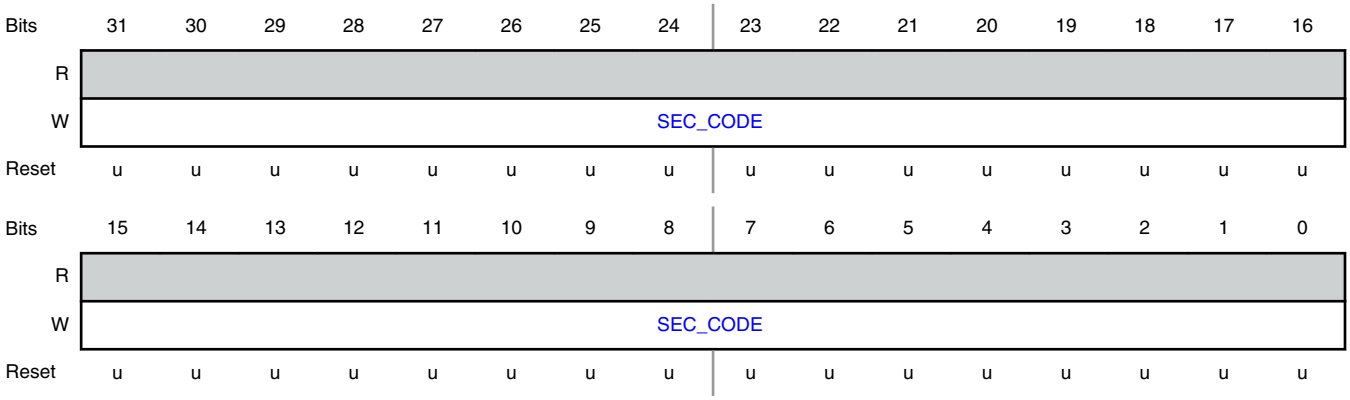
Offset

Register	Offset
CODESECURITYPROT	FF0h

Function

Security code allows test access via SWD/JTAG. Reset with POR, software reset or BOD

Diagram



Fields

Field	Function
31-0 SEC_CODE	Security Code Security code allows debug via SWD. Write once register, value 0x87654321 disables the access and therefore prevents any chance to enable it. Writing any other value enables the access and locks the mode. In some cases, the boot code will secure the device by writing to this register to disable SWD. This would prevent the application being able to re-enable this access.

Chapter 3
I/O Pin Configuration (IOCON)

3.1 IOCON register descriptions

3.1.1 IOCON memory map

IOCON base address: 4000_F000h

Offset	Register	Width (In bits)	Access	Reset value
0h	PIO0 Register (PIO0)	32	RW	See description
4h	PIO1 Register (PIO1)	32	RW	See description
8h	PIO2 Register (PIO2)	32	RW	See description
Ch	PIO3 Register (PIO3)	32	RW	See description
10h	PIO4 Register (PIO4)	32	RW	See description
14h	PIO5 Register (PIO5)	32	RW	See description
18h	PIO6 Register (PIO6)	32	RW	See description
1Ch	PIO7 Register (PIO7)	32	RW	See description
20h	PIO8 Register (PIO8)	32	RW	See description
24h	PIO9 Register (PIO9)	32	RW	See description
28h - 2Ch	I2C specific PIOa Register (PIO_I2C10 - PIO_I2C11)	32	RW	See description
30h	PIO12 Register (PIO12)	32	RW	See description
34h	PIO13 Register (PIO13)	32	RW	See description
38h	PIO14 Register (PIO14)	32	RW	See description

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
3Ch	PIO15 Register (PIO15)	32	RW	See description
40h	PIO16 Register (PIO16)	32	RW	See description
44h	PIO17 Register (PIO17)	32	RW	See description
48h	PIO18 Register (PIO18)	32	RW	See description
4Ch	PIO19 Register (PIO19)	32	RW	See description
50h	PIO20 Register (PIO20)	32	RW	See description
54h	PIO21 Register (PIO21)	32	RW	See description

3.1.2 PIOa Register (PIO0 - PIO21)

Offset

Register	Offset
PIO0	0h
PIO1	4h
PIO2	8h
PIO3	Ch
PIO4	10h
PIO5	14h
PIO6	18h
PIO7	1Ch
PIO8	20h
PIO9	24h
PIO12	30h
PIO13	34h
PIO14	38h
PIO15	3Ch

Table continues on the next page...

Table continued from the previous page...

Register	Offset
PIO16	40h
PIO17	44h
PIO18	48h
PIO19	4Ch
PIO20	50h
PIO21	54h

Function

Configuration array for PIO0 to PIO21. PIO[10] and PIO[11] use a different IO cell type to the other PIO pins and so there are some differences in the bit field descriptions of the PIO register for these IOs; see PIO_I2C. Reset values vary depending on whether the IO is configured with a pull-up or pull-down resistor as default. The value is also affected by the IO type. Bit fields are different for PIO[10] and PIO[11], see PIO_I2C decription. Reset value 0x180 for PIO 0,3,4,5,8,9,14,15,16,21. Reset value 0x198 for PIO 1,2,6,7,17,18,19,20. Reset value 0x188 for PIO 10,11. Reset value 0x182 for PIO 12, 13.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved				SSEL	OD	SLEW 1	FILTERO...	DIGIMODE	INVERT	SLEW 0	MODE		FUNC		
Reset	u	u	u	u	0	0	0	1	1	0	0	0	0	0	0	0

Fields

Field	Function
31-12 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
11 SSEL	IO Clamping Function This bit controls the IO clamping function, IO_CLAMP. Assert to freeze the IO. Also needs SYSCON_RETENTIONCTRL set as well. Useful in power down mode. This mode is held through power down cycle. Before releasing this mode on a wake-up, ensure the IO is set to the required direction and value by using GPIO_DIR and GPIO_PIN registers.

Table continues on the next page...

Table continued from the previous page...

Field	Function
10 OD	Control Open-drain Mode 0b - Normal. Normal push-pull output. 1b - Open-drain. Simulated open-drain output (high drive disabled).
9 SLEW1	Driver Slew Rate This bit is used in combination with SLEW0. The higher [SLEW1,SLEW0], the quicker the slew rate.
8 FILTEROFF	Control Input Glitch Filter 0b - Filter enabled. Noise pulses below approximately 1 ns are filtered out. 1b - Filter disabled. No input filtering is done.
7 DIGIMODE	Select Analog/Digital Mode When in analog mode, the receiver path in the IO cell is disabled. In this mode, it is essential that the digital function (e.g. GPIO) is not configured as an output. Otherwise it may conflict with analog stuff (loopback of digital on analog input). In other words, the digital output is not automatically disabled when the IO is in analog mode. As a consequence, it is not possible to disable the receiver path when the IO is used for digital output purpose. 0b - Analog mode. 1b - Digital mode.
6 INVERT	Input Polarity 0b - Disabled. Input function is not inverted. 1b - Enabled. Input function is inverted.
5 SLEW0	Driver Slew Rate This bit field is used in combination with SLEW1. The higher [SLEW1,SLEW0] the quicker the IO cell slew rate.
4-3 MODE	Select Function Mode Select function mode (on-chip pull-up/pull-down resistor control). For MFIO type ONLY . Note: When the register is related to a general purpose MFIO type pad (that is all PIOs except PIO10 & 11): Bit [3] (of the register) is connected to EPD (enable pull-down) input of the MFIO pad. Bit [4] (of the register) is connected to EPUN (enable pull-up NOT) input of MFIO pad. 00b - Pull-up resistor enabled. 01b - Repeater mode (bus keeper). 10b - Plain Input. 11b - Pull-down resistor enabled.

Table continues on the next page...

Table continued from the previous page...

Field	Function
2-0 FUNC	Select Digital Function Select digital function assigned to this pin. 000b - GPIO mode 001b-111b - See IO MUX.

3.1.3 I2C specific PIOa Register (PIO_I2C10 - PIO_I2C11)

Offset

Register	Offset
PIO_I2C10	28h
PIO_I2C11	2Ch

Function

Configuration array for I2C specific IO cells on PIO[10] and PIO[11]. Reset value 0x188 for PIO 10,11.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved			IO_CLAMP	Reserved	OD	FSEL	FILTERO...	DIGIMODE	INVERT	EHS	ECS	EGP	FUNC		
Reset	u	u	u	0	0	0	0	1	1	0	0	0	1	0	0	0

Fields

Field	Function
31-13 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
12 IO_CLAMP	IO_CLAMP Assert to freeze the IO. Also needs SYSCON_RETENTIONCTRL set as well. Useful in power down mode. This mode is held through power down cycle. Before releasing this mode on a wake-up, ensure the IO is set to the required direction and value by using GPIO_DIR and GPIO_PIN registers.
11 —	Reserved Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
10 OD	Control Open-drain Mode 0b - Normal. Normal push-pull output. 1b - Open-drain. Simulated open-drain output (high drive disabled).
9 FSEL	Control Input Glitch Filter 0b - Noise pulses below approximately 50 ns are filtered out. 1b - Noise pulses below approximately 10 ns are filtered out. If IO is in GPIO mode this control bit is irrelevant, a 3 ns filter is used
8 FILTEROFF	Control Input Glitch Filter 0b - Filter enabled. See FSEL for further information. 1b - Filter disabled. No input filtering is done.
7 DIGIMODE	Select Analog/Digital Mode When in analog mode, the receiver path in the IO cell is disabled. In this mode, it is essential that the digital function (e.g. GPIO) is not configured as an output. Otherwise it may conflict with analog stuff (loopback of digital on analog input). As a consequence, it is not possible to disable the receiver path when the IO is used for digital output purpose. 0b - Analog mode. 1b - Digital mode.
6 INVERT	Input Polarity 0b - Disabled. Input function is not inverted. 1b - Enabled. Input function is inverted.
5 EHS	Speed Selection When IO is in GPIO mode set 1 for high speed GPIO, 0 for low speed GPIO. For IIC mode, this bit has no effect and the IO is always in low speed..
4 ECS	Pull-up Current Source Enable Pull-up current source enable when set. When IO is in IIC mode (EGP=0) and ECS is low, the IO cell is an open drain cell.

Table continues on the next page...

Table continued from the previous page...

Field	Function
3 EGP	GPIO Mode of IO Cell 0b - IIC mode 1b - GPIO mode.
2-0 FUNC	Select Digital Function Select digital function assigned to this pin. 000b GPIO mode 001b-111b See IO MUX.

Chapter 4
Input Multiplexing (INPUTMUX)

4.1 INPUTMUX register descriptions

4.1.1 INPUTMUX memory map

INPUTMUX base address: 4000_E000h

Offset	Register	Width (In bits)	Access	Reset value
C0h - DCh	Pin Interrupt Select Register a (PINTSEL0 - PINTSEL7)	32	RW	See description
E0h - 128h	DMA Channel Trigger Select Register a (DMA_ITRIG_INMUX0 - DMA_ITRIG_INMUX18)	32	RW	See description
160h - 16Ch	DMA Output Trigger to DMA Trigger Mux Input Register a (DMA_OTRIG_INMUX0 - DMA_OTRIG_INMUX3)	32	RW	See description
180h	Frequency Measurement Reference Clock Selection Register (FREQMEAS_REF)	32	RW	See description
184h	Frequency Measurement Target Clock Selection Register (FREQMEAS_TARGET)	32	RW	See description

4.1.2 Pin Interrupt Select Register a (PINTSEL0 - PINTSEL7)

Offset

For a = 0 to 7:

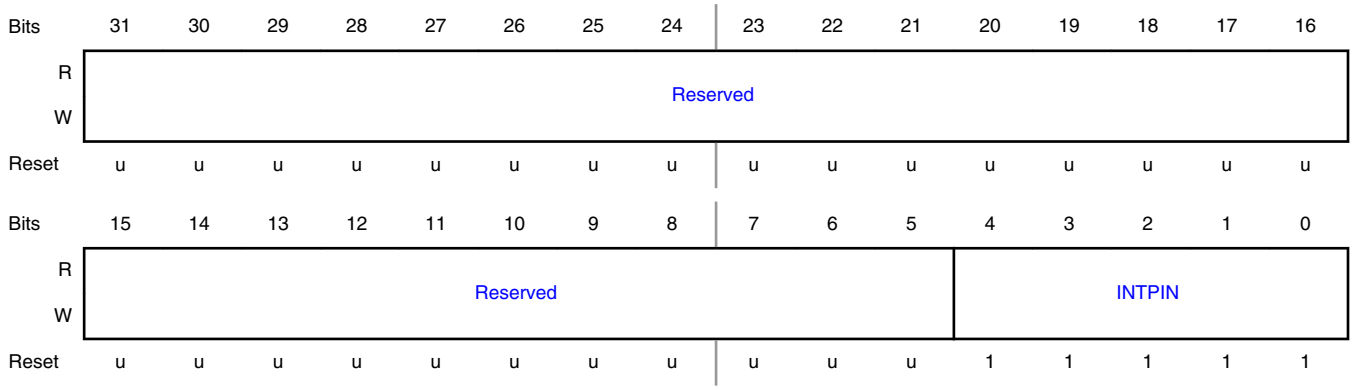
Register	Offset
PINTSELa	C0h + (a × 4h)

Function

Each of these registers selects one pin from the PIOs as the source of a pin interrupt or as the input to the pattern match engine. To select a pin for any of the 4 pin interrupts or 8 pattern match engine inputs, write the GPIO port pin number as 0 to 21 for pins PIO0 PIO21 to the INTPIN bits. For example, setting INTPIN to 0x5 in PINTSEL0 selects pin PIO5 for pin interrupt 0.

Each of the pin interrupts must be enabled in the NVIC before it becomes active.

Diagram



Fields

Field	Function
31-5 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
4-0 INTPIN	Pin Number Select Pin number select for pin interrupt or pattern match engine input.

4.1.3 DMA Channel Trigger Select Register a (DMA_ITRIG_INMUX0 - DMA_ITRIG_INMUX18)

Offset

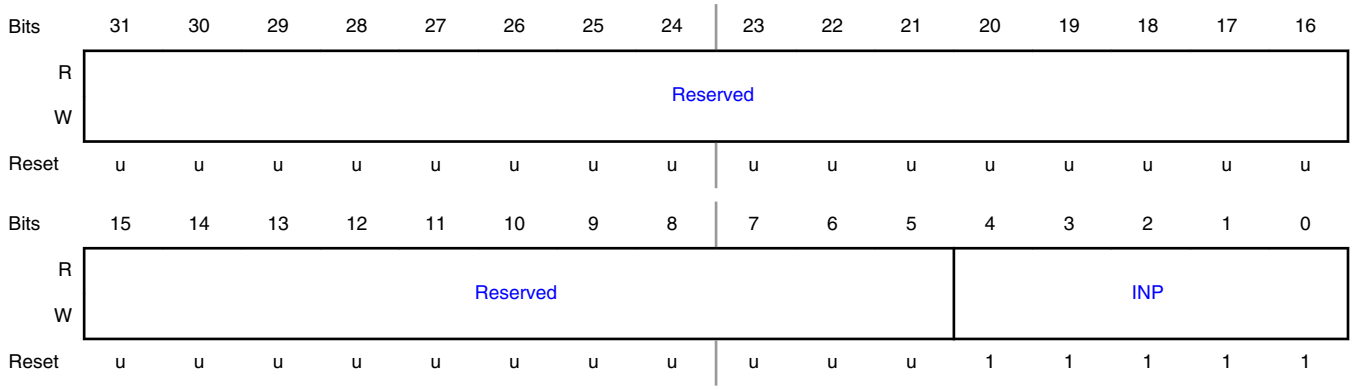
For a = 0 to 18:

Register	Offset
DMA_ITRIG_INMUXa	E0h + (a × 4h)

Function

With the DMA trigger input mux registers, one trigger input can be selected for each of the DMA channels from the potential internal sources. By default, none of the triggers are selected.

Diagram



Fields

Field	Function
31-5 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
4-0 INP	DMA Channel Trigger input Number This field configure the trigger input number (decimal value) for DMA channel n (n = 0 to 18). 00000b ADC0 Sequence A interrupt 00001b Reserved 00010b Timer CT32B0 Match 0 00011b Timer CT32B0 Match 1 00100b Timer CT32B1 Match 0 00101b Timer CT32B1 Match 1 00110b Pin interrupt 0 00111b Pin interrupt 1 01000b Pin interrupt 2 01001b Pin interrupt 3 01010b AES RX 01011b AES TX 01100b Hash RX 01101b Hash TX 01110b DMA output trigger mux 0 01111b DMA output trigger mux 1 10000b DMA output trigger mux 2 10001b DMA output trigger mux 3 10010b-11111b Reserved

4.1.4 DMA Output Trigger to DMA Trigger Mux Input Register a (DMA_OTRIG_INMUX0 - DMA_OTRIG_INMUX3)

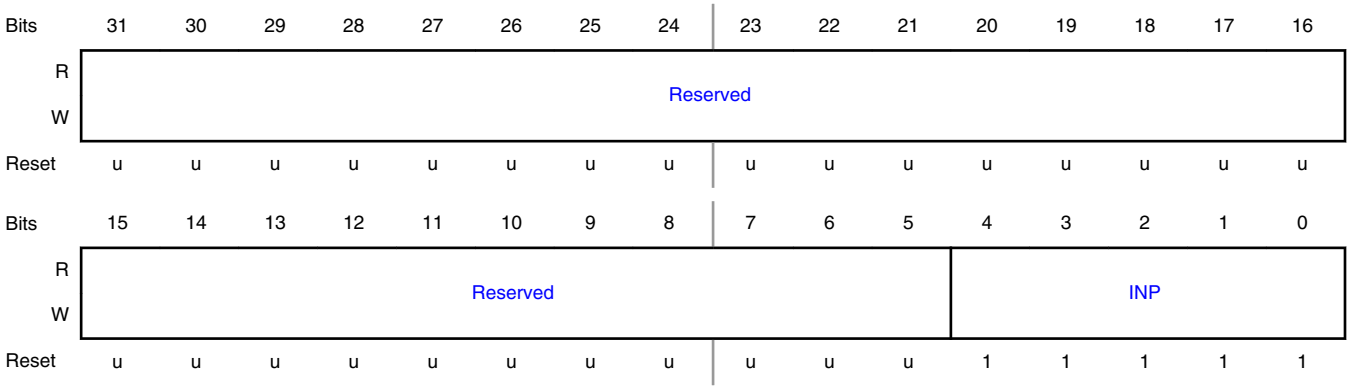
Offset

Register	Offset
DMA_OTRIG_INMUX0	160h
DMA_OTRIG_INMUX1	164h
DMA_OTRIG_INMUX2	168h
DMA_OTRIG_INMUX3	16Ch

Function

This register provides a multiplexer for inputs 14 to 17 of each DMA trigger input mux register DMA_ITRIG_INMUX. These inputs can be selected from among the trigger outputs generated by the each DMA channel. By default, none of the triggers are selected.

Diagram



Fields

Field	Function
31-5 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
4-0 INP	DMA Trigger Output for DMA Channel Select which DMA channel output will be used to generate the DMA trigger input, 'DMA Channel Input Mux n'. Values 0 to 18 are valid.

4.1.5 Frequency Measurement Reference Clock Selection Register (FREQMEAS_REF)

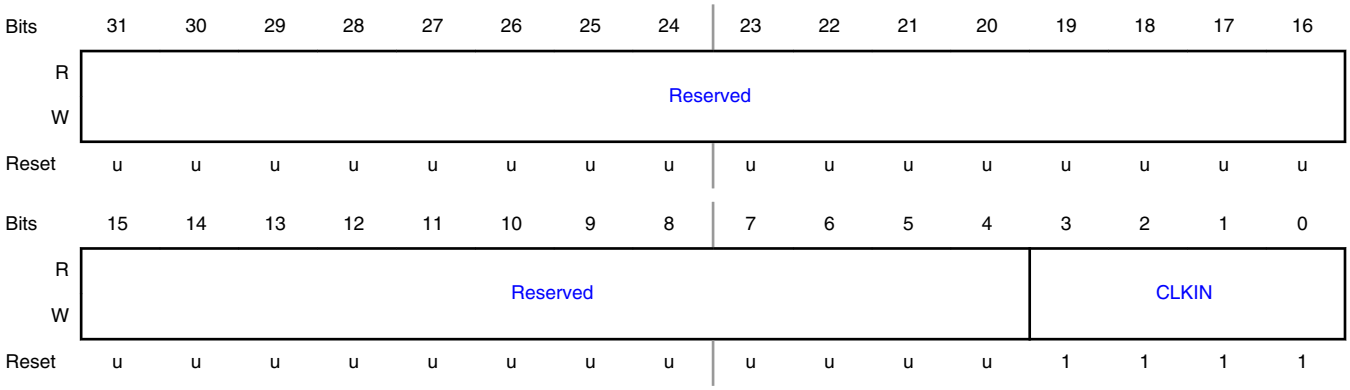
Offset

Register	Offset
FREQMEAS_REF	180h

Function

This register selects a clock for the reference clock of the frequency measure function. By default, no clock is selected.

Diagram



Fields

Field	Function
31-4 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
3-0 CLKIN	Frequency Measure Function Reference Clock Selection Numbe Clock source number (decimal value) for frequency measure function ref clock 0000b Reserved 0001b XTAL 32 MHz (must be enabled in clock_ctrl) 0010b FRO 1 MHz (must be enabled in clock_ctrl) 0011b 32 kHz oscillator (either FRO 32 KHz or XTAL 32 KHZ) 0100b Main clock (divided) 0101b PIO[4] (must be configured as GPIO) 0110b PIO[20] (must be configured as GPIO) 0111b PIO[16] (must be configured as GPIO) 1000b PIO[15] (must be configured as GPIO) 1001b-1111b Reserved

4.1.6 Frequency Measurement Target Clock Selection Register (FREQMEAS_TARGET)

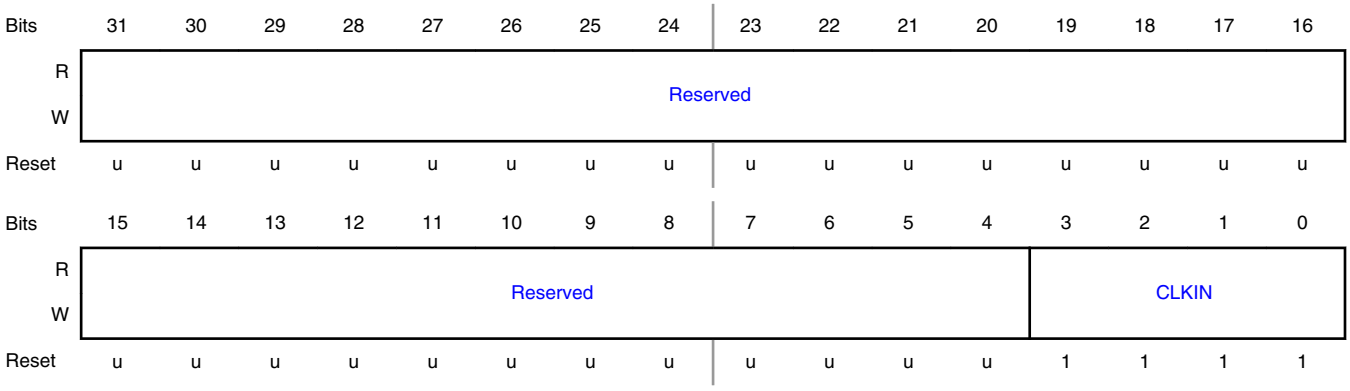
Offset

Register	Offset
FREQMEAS_TARGET	184h

Function

This register selects a clock for the target clock of the frequency measure function. By default, no clock is selected.

Diagram



Fields

Field	Function
31-4 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
3-0 CLKIN	Frequency Measurement Target Clock Source Number Clock source number (decimal value) for frequency measure function target clock 0000b Reserved 0001b XTAL 32 MHz (must be enabled in clock_ctrl) 0010b FRO 1 MHz (must be enabled in clock_ctrl) 0011b 32 kHz oscillator (either FRO 32 KHz or XTAL 32 KHZ) 0100b Main clock (divided) 0101b PIO[4] (must be configured as GPIO) 0110b PIO[20] (must be configured as GPIO) 0111b PIO[16] (must be configured as GPIO) 1000b PIO[15] (must be configured as GPIO) 1001b-1111b Reserved

Chapter 5
General Purpose I/O (GPIO)

5.1 GPIO register descriptions

5.1.1 GPIO memory map

GPIO base address: 4008_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h - 15h	Byte Pin Register a (B0 - B21)	8	RW	See description
1000h - 1054h	Word Pin Register a (W0 - W21)	32	RW	See description
2000h	Direction Register (DIR)	32	RW	See description
2080h	Mask Register (MASK)	32	RW	See description
2100h	Pin Register (PIN)	32	RW	See description
2180h	Masked Pin Register (MPIN)	32	RW	See description
2200h	Set Register (SET)	32	RW	See description
2280h	Clear Pin Register Bits Register (CLR)	32	WO	See description
2300h	Toggle Pin Register Bits Register (NOT)	32	WO	See description
2380h	Set Pin Direction Register (DIRSET)	32	WO	See description
2400h	Clear Pin Direction Register (DIRCLR)	32	WO	See description
2480h	Toggle Pin Direction Register (DIRNOT)	32	WO	See description

5.1.2 Byte Pin Register a (B0 - B21)

Offset

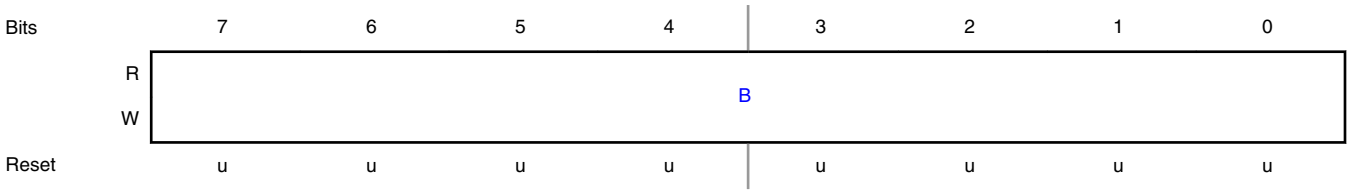
For a = 0 to 21:

Register	Offset
Ba	0h + (a × 1h)

Function

Each PIO pin has a byte register in this address range. Software typically reads and writes bytes to access individual pins, but can read or write halfwords to sense or set the state of two pins, and read or write words to sense or set the state of four pins. There is one register for each PIO.

Diagram



Fields

Field	Function
7-0 B	Byte Pin Read 0: pin PIOn is LOW. Read 0xFF: pin PIOn is HIGH. Only 0 or 0xFF can be read. Write 0: clear output bit. Write any value 0x01 to 0xFF: set output bit. Reset values reflects the state of pin given by the relevant bit of PIN reset value. The PIO output value will not affect the IO unless the IO is in GPIO mode.

5.1.3 Word Pin Register a (W0 - W21)

Offset

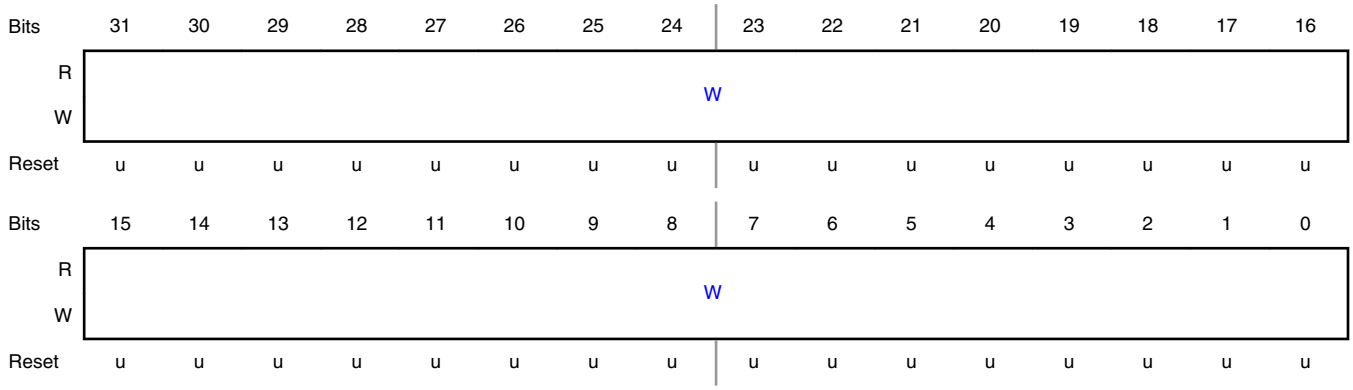
For a = 0 to 21:

Register	Offset
Wa	1000h + (a × 4h)

Function

There is one register for each PIO. The PIO output value will not affect the IO unless the IO is in GPIO mode.

Diagram



Fields

Field	Function
31-0 W	Word Pin Read 0: pin PIOn is LOW. Read 0xFFFFFFFF: pin PIOn is HIGH. Only 0 or 0xFFFF FFFF can be read. Write 0: clear output bit. Write any value 0x00000001 to 0xFFFFFFFF: set output bit. Reset values reflects the state of pin given by the relevant bit of PIN reset value.

5.1.4 Direction Register (DIR)

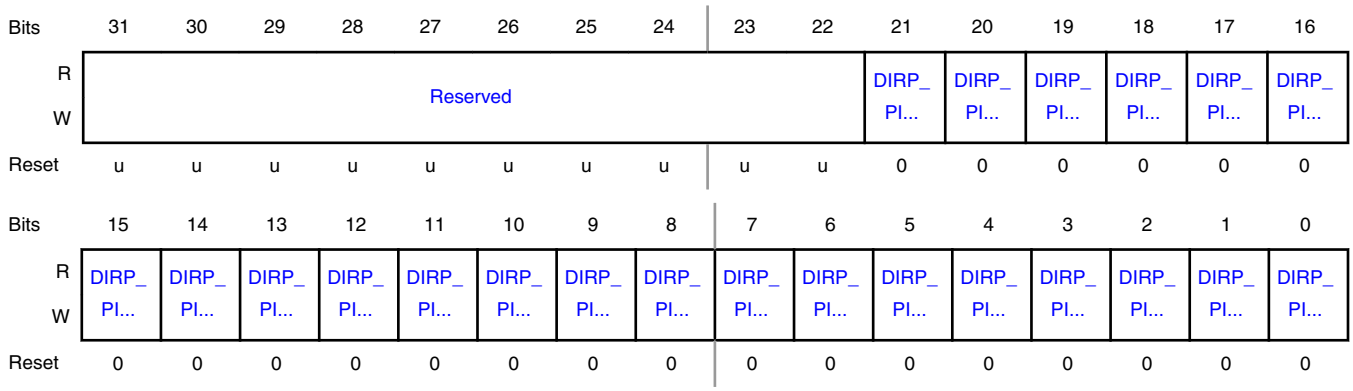
Offset

Register	Offset
DIR	2000h

Function

Configure the direction of GPIO function. If the PIO is configured for GPIO mode then this direction will control the PIO direction.

Diagram



Fields

Field	Function
31-22 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
21-0 DIRP_PIOn	Selects pin direction for pin PIOn . 0b - Input. 1b - Output.

5.1.5 Mask Register (MASK)

Offset

Register	Offset
MASK	2080h

Function
This register affects writing and reading the MPIN register. Zeros in these register fields enable reading and writing; ones disable writing and result in zeros in corresponding positions when reading.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved										MASK P_P...	MASK P_P...	MASK P_P...	MASK P_P...	MASK P_P...	MASK P_P...
Reset	u	u	u	u	u	u	u	u	u	u	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MASK P_P...	MASK P_P...	MASK P_P...	MASK P_P...	MASK P_P...	MASK P_P...	MASK P_P...	MASK P_P...	MASK P_P...	MASK P_P...	MASK P_P...	MASK P_P...	MASK P_P...	MASK P_P...	MASK P_P...	MASK P_P...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-22 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
21-0 MASKP_PIOn	MPIN[MPORT_PIOn] Active Control Controls if MPORT_PIOn is active in MPIN register. 0b - Mask bit is clear/mask not active, the PIO will be active when using the MPIN register. 1b - Mask bit is set/mask is active, the PIO will not be active when using the MPIN register.

5.1.6 Pin Register (PIN)

Offset

Register	Offset
PIN	2100h

Function

Reading these registers returns the current state of the pins read, regardless of direction, masking, or alternate functions, except that pins configured as analog I/O always read as 0s. Writing these registers loads the output bits of the pins written to, regardless of the Mask register; provided the PIO is configured in its GPIO function.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved										PORT_PI...	PORT_PI...	PORT_PI...	PORT_PI...	PORT_PI...	PORT_PI...
Reset	u	u	u	u	u	u	u	u	u	u	1	0	0	0	0	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PORT_PI...	PORT_PI...	PORT_PI...	PORT_PI...	PORT_PI...	PORT_PI...	PORT_PI...	PORT_PI...	PORT_PI...	PORT_PI...	PORT_PI...	PORT_PI...	PORT_PI...	PORT_PI...	PORT_PI...	PORT_PI...
Reset	1	1	1	1	1	1	1	1	0	0	1	1	1	0	0	1

Fields

Field	Function
31-22 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
21-0 PORT_PIOn	Reads Pinn States or Loads OutpuT 0b - Read: pin is low; write: clear output bit. 1b - Read: pin is high; write: set output bit.

5.1.7 Masked Pin Register (MPIN)

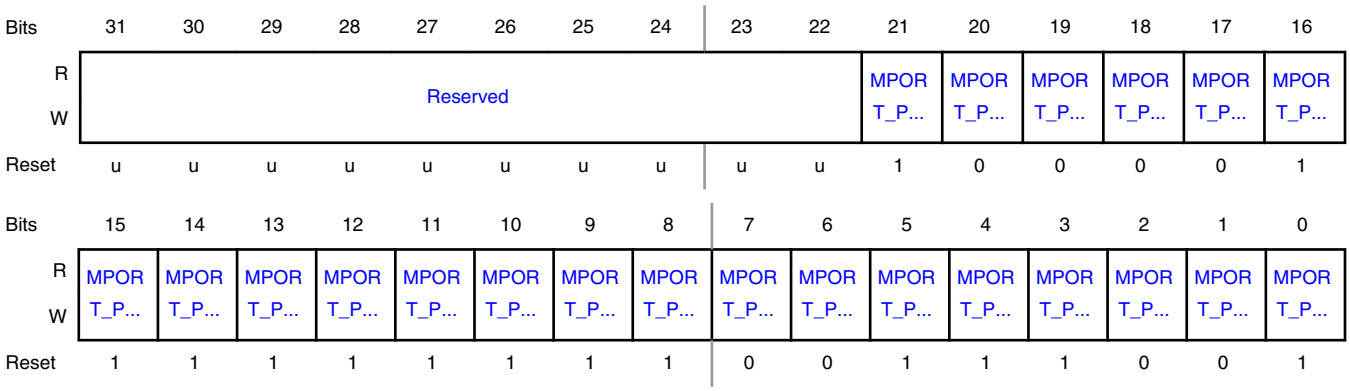
Offset

Register	Offset
MPIN	2180h

Function

This register is similar to the PIN register, except that the value read is masked by ANDing with the inverted contents of the corresponding MASK register, and writing to one of these register bits only affects output register bits that are enabled by zeros in the corresponding MASK register.

Diagram



Fields

Field	Function
31-22 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
21-0 MPORT_PIOn	Masked Pinn 0b - Read: pin is LOW and/or the corresponding bit in the MASK register is 1; write: clear output bit if the corresponding bit in the MASK register is 0. 1b - Read: pin is HIGH and the corresponding bit in the MASK register is 0; write: set output bit if the corresponding bit in the MASK register is 0.

5.1.8 Set Register (SET)

Offset

Register	Offset
SET	2200h

Function

Output bits can be set by writing ones to these register bits, regardless of MASK register bits. Reading from these register bits returns the port's output bits, regardless of pin directions.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved										SETP_PI...	SETP_PI...	SETP_PI...	SETP_PI...	SETP_PI...	SETP_PI...
W																
Reset	u	u	u	u	u	u	u	u	u	u	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SETP_PI...	SETP_PI...	SETP_PI...	SETP_PI...	SETP_PI...	SETP_PI...	SETP_PI...	SETP_PI...	SETP_PI...	SETP_PI...	SETP_PI...	SETP_PI...	SETP_PI...	SETP_PI...	SETP_PI...	SETP_PI...
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-22 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
21-0 SETP_PIOn	PIOn Read or Set Output Bit 0b - Read: output bit: write: no operation. 1b - Read: output bit; write: set output bit.

5.1.9 Clear Pin Register Bits Register (CLR)

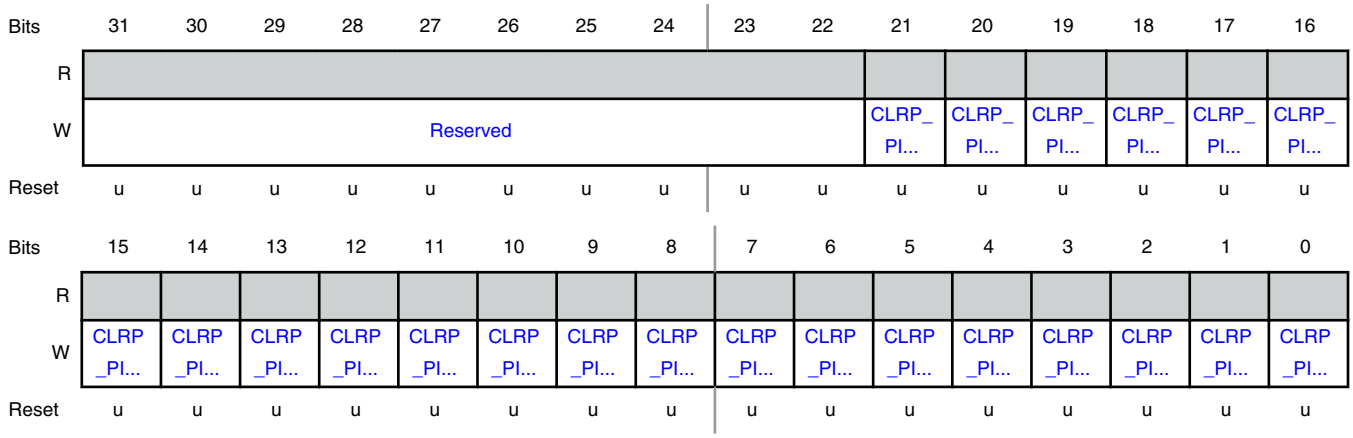
Offset

Register	Offset
CLR	2280h

Function

Output bits can be cleared by writing ones to these write-only register bits, regardless of MASK register bits.

Diagram



Fields

Field	Function
31-22 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
21-0 CLRP_PIOn	Clear PIOn Output Bits 0b - No operation. 1b - Clear output bit.

5.1.10 Toggle Pin Register Bits Register (NOT)

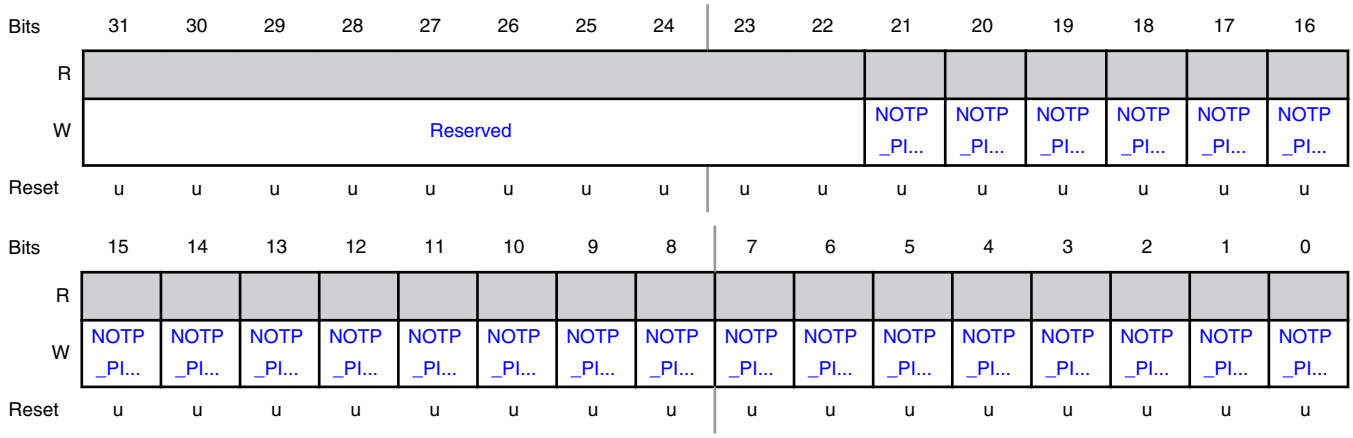
Offset

Register	Offset
NOT	2300h

Function

Output bits can be toggled/ inverted/ complemented by writing ones to these write-only register, regardless of MASK register.

Diagram



Fields

Field	Function
31-22 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
21-0 NOTP_PIOn	PIOn Toggle Output Bits 0b - No operation. 1b - Toggle output bit.

5.1.11 Set Pin Direction Register (DIRSET)

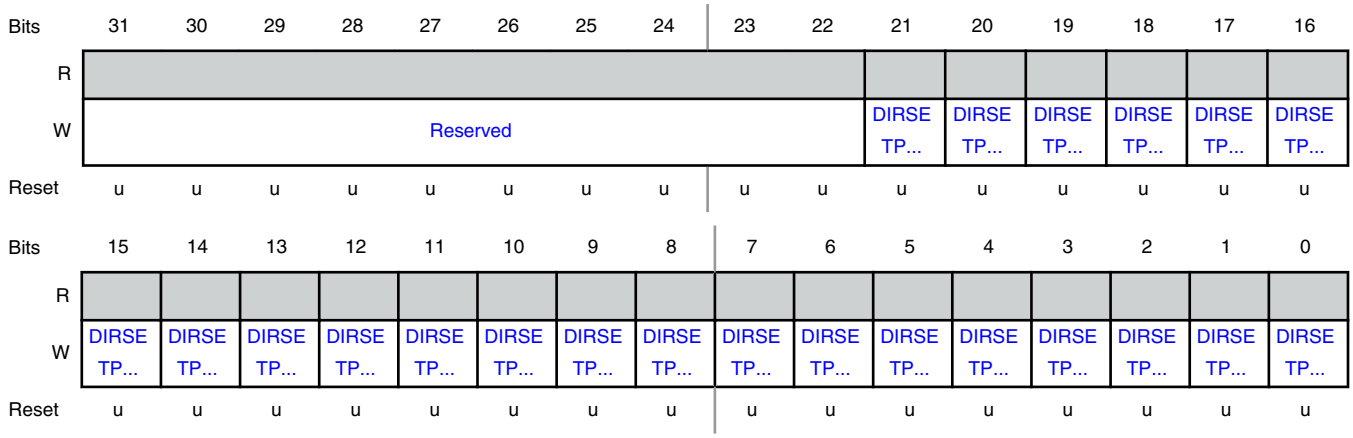
Offset

Register	Offset
DIRSET	2380h

Function

Direction bits can be set by writing ones to these register bits.

Diagram



Fields

Field	Function
31-22 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
21-0 DIRSETP_PIOn	Set PIOn Direction 0b - No operation. 1b - Set direction bit.

5.1.12 Clear Pin Direction Register (DIRCLR)

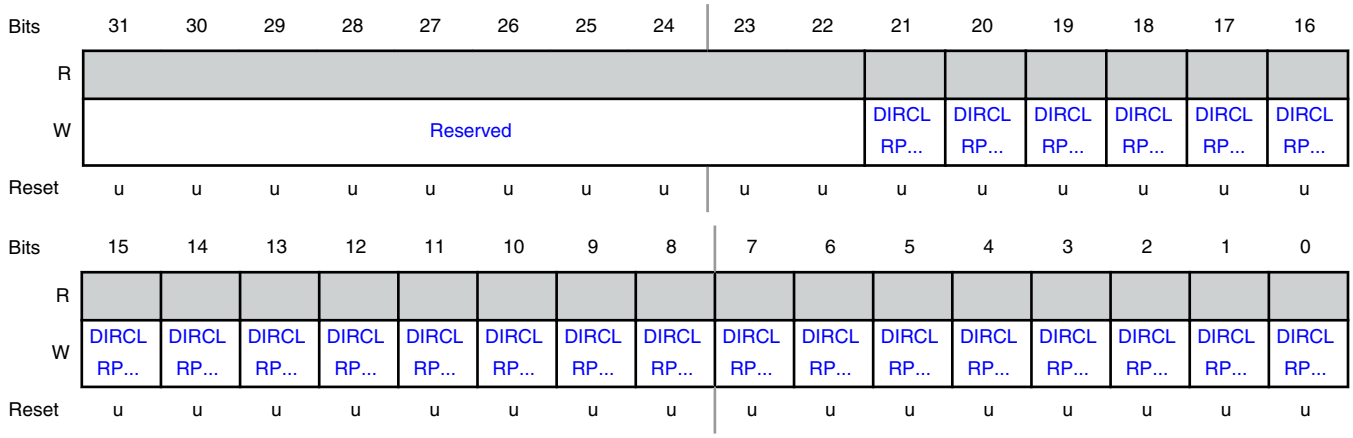
Offset

Register	Offset
DIRCLR	2400h

Function

Direction bits can be cleared by writing ones to these write-only register bits.

Diagram



Fields

Field	Function
31-22 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
21-0 DIRCLRP_PIOn	Clear PIOn Direction 0b - No operation. 1b - Clear direction bit

5.1.13 Toggle Pin Direction Register (DIRNOT)

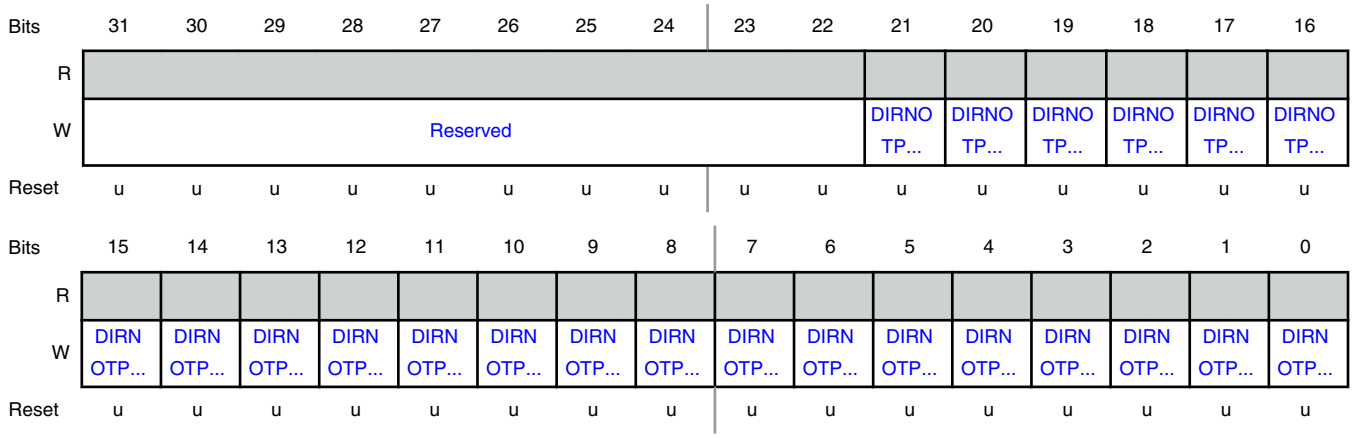
Offset

Register	Offset
DIRNOT	2480h

Function

Direction bits can be set by writing ones to these write-only register bits.

Diagram



Fields

Field	Function
31-22 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
21-0 DIRNOTP_PIOn	Toggle PIOn Direction 0b - No operation. 1b - Toggle direction bit.

Chapter 6
Pin interrupt and Pattern Match (PINT)

6.1 PINT register descriptions

6.1.1 PINT memory map

PINT base address: 4001_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Pin Interrupt Mode Register (ISEL)	32	RW	See description
4h	Pin Interrupt Level or Rising Edge Interrupt Enable Register (IENR)	32	RW	See description
8h	Pin Interrupt Level or Rising Edge Interrupt Set Register (SIENR)	32	WO	See description
Ch	Pin Interrupt Level (Rising Edge Interrupt) Clear Register (CIENR)	32	WO	See description
10h	Pin Interrupt Active Level or Falling Edge Interrupt Enable Register (IENF)	32	RW	See description
14h	Pin Interrupt Active Level or Falling Edge Interrupt Set Register (SIENF)	32	WO	See description
18h	Pin Interrupt Active Level or Falling Edge Interrupt Clear Register (CIENF)	32	WO	See description
1Ch	Pin Interrupt Rising Edge Register (RISE)	32	RW	See description
20h	Pin Interrupt Falling Edge Register (FALL)	32	RW	See description
24h	Pin Interrupt Status Register (IST)	32	RW	See description
28h	Pattern Match Interrupt Control Register (PMCTRL)	32	RW	See description
2Ch	Pattern Match Interrupt Bit-slice Source Register (PMSRC)	32	RW	See description
30h	Pattern Match Interrupt Bit Slice Configuration Register (PMCFG)	32	RW	See description

6.1.2 Pin Interrupt Mode Register (ISEL)

Offset

Register	Offset
ISEL	0h

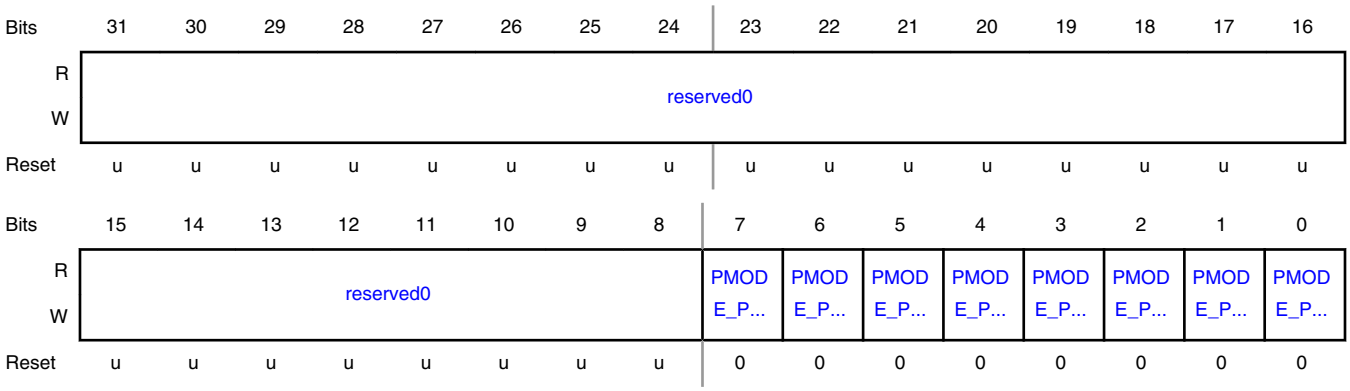
Function

For each of the 8 pin interrupts selected in the INPUTMUX_PINTSELn registers, one bit in the ISEL register determines whether the interrupt is edge or level sensitive.

NOTE

Only interrupts 0 to 3 are supported to processor.

Diagram



Fields

Field	Function
31-8 reserved0	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
7-0 PMODE_PINn	Pin Interrupt n Interrupt Mode Select Selects the interrupt mode for pin interrupt n (selected in PINTSELn). 0b - Edge sensitive. 1b - Level sensitive.

6.1.3 Pin Interrupt Level or Rising Edge Interrupt Enable Register (IENR)

Offset

Register	Offset
IENR	4h

Function

For each of the 8 pin interrupts selected in the INPUTMUX_PINTSELn registers, one bit in the IENR register enables the interrupt depending on the pin interrupt mode configured in the ISEL register:

- If the pin interrupt mode is edge sensitive (ISEL[PMODE_PINn] = 0), the rising edge interrupt is enabled.
- If the pin interrupt mode is level sensitive (ISEL[PMODE_PINn] = 1), the level interrupt is enabled.

The IENF register configures the active level (HIGH or LOW) for this interrupt.

NOTE

Only interrupts 0 to 3 are supported to the processor.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	reserved0															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	reserved0								ENRL_PI...	ENRL_PI...	ENRL_PI...	ENRL_PI...	ENRL_PI...	ENRL_PI...	ENRL_PI...	ENRL_PI...
Reset	u	u	u	u	u	u	u	u	0	0	0	0	0	0	0	0

Fields

Field	Function
31-8 reserved0	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
7-0 ENRL_PINn	Pin Interrupt n Rising Edge or Level Interrupt Enable Enables the rising edge or level interrupt for pin interrupt n (selected in INPUTMUX_PINTSELn). 0b - Disable rising edge or level interrupt. 1b - Enable rising edge or level interrupt.

6.1.4 Pin Interrupt Level or Rising Edge Interrupt Set Register (SIENR)

Offset

Register	Offset
SIENR	8h

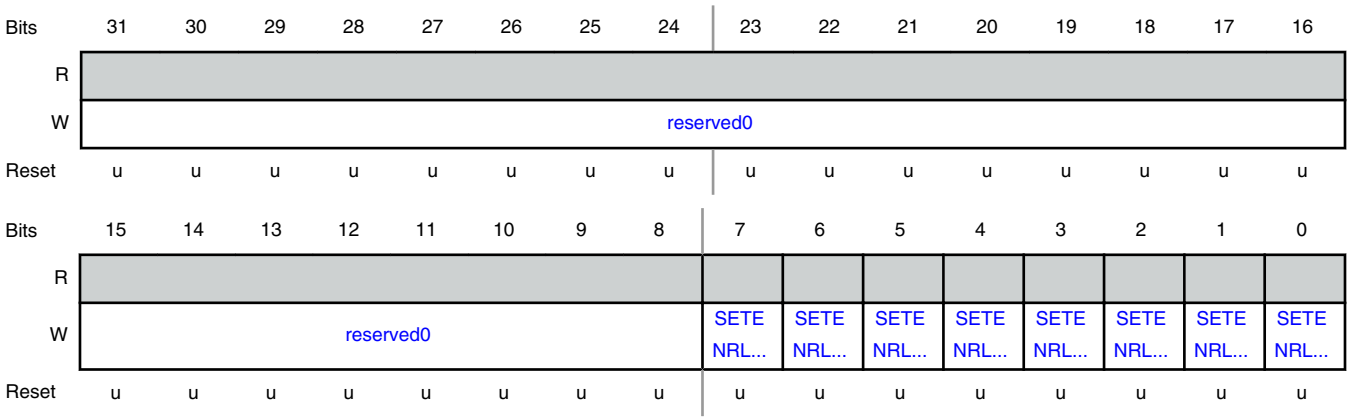
Function

For each of the 8 pin interrupts selected in the INPUTMUX_PINTSELn registers, one bit in the SIENR register sets the corresponding bit in the IENR register depending on the pin interrupt mode configured in the ISEL register. See IENR description.

NOTE

Only interrupts 0 to 3 are supported to processor.

Diagram



Fields

Field	Function
31-8 reserved0	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
7-0 SETENRL_PINn	ENRL_PINn Set 1 written to this address set IENR[ENRL_PINn], thus enabling interrupts. 0b - No operation. 1b - Enable rising edge or level interrupt.

6.1.5 Pin Interrupt Level (Rising Edge Interrupt) Clear Register (CIENR)

Offset

Register	Offset
CIENR	Ch

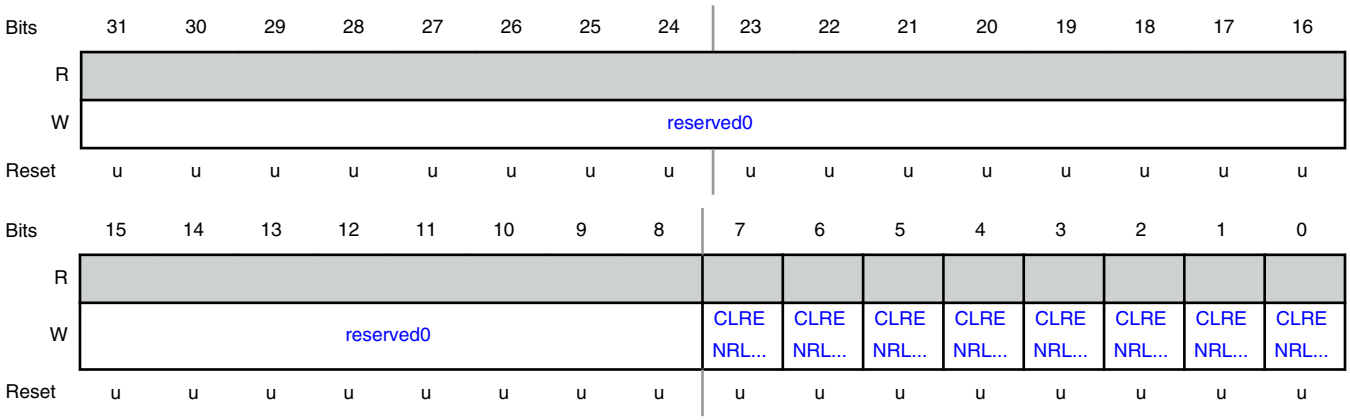
Function

For each of the 8 pin interrupts selected in the INPUTMUX_PINTSELn registers, one bit in the CIENR register clears the corresponding bit in the IENR register depending on the pin interrupt mode configured in the ISEL register. See IENR description.

NOTE

Only interrupts 0 to 3 are supported to processor.

Diagram



Fields

Field	Function
31-8 reserved0	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
7-0 CLRENRL_PINn	ENRL_PINn Clear 1 written to this address clear IENR[ENRL_PINn], thus disabling the interrupts. 0b - No operation. 1b - Disable rising edge or level interrupt.

6.1.6 Pin Interrupt Active Level or Falling Edge Interrupt Enable Register (IENF)

Offset

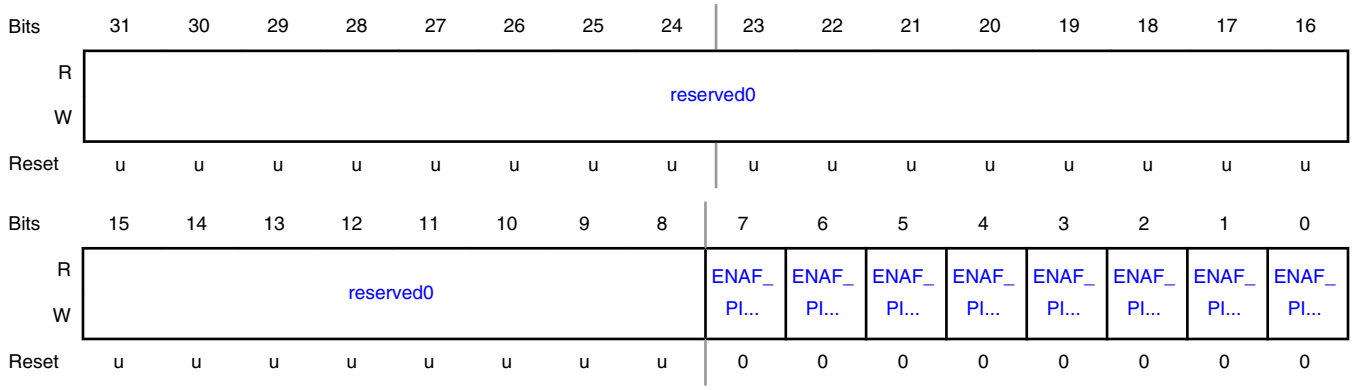
Register	Offset
IENF	10h

Function

For each of the 8 pin interrupts selected in the INPUTMUX_PINTSELn registers, one bit in the IENF register enables the falling edge interrupt or configures the level sensitivity depending on the pin interrupt mode configured in the ISEL register:

- If the pin interrupt mode is edge sensitive (ISEL[PMODE_PINn] = 0), the falling edge interrupt is enabled.
- If the pin interrupt mode is level sensitive (ISEL[PMODE_PINn] = 1), the active level of the level interrupt (HIGH or LOW) is configured.

Diagram



Fields

Field	Function
31-8 reserved0	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
7-0 ENAF_PINn	Pin Interrupt n Fall Edge Enable and Active Level Interrupt Configure Enables the falling edge or configures the active level interrupt for Pin Interrupt n (selected in PINTSELn). 0b - Disable falling edge interrupt or set active interrupt level LOW. 1b - Enable falling edge interrupt enabled or set active interrupt level HIGH.

6.1.7 Pin Interrupt Active Level or Falling Edge Interrupt Set Register (SIENF)

Offset

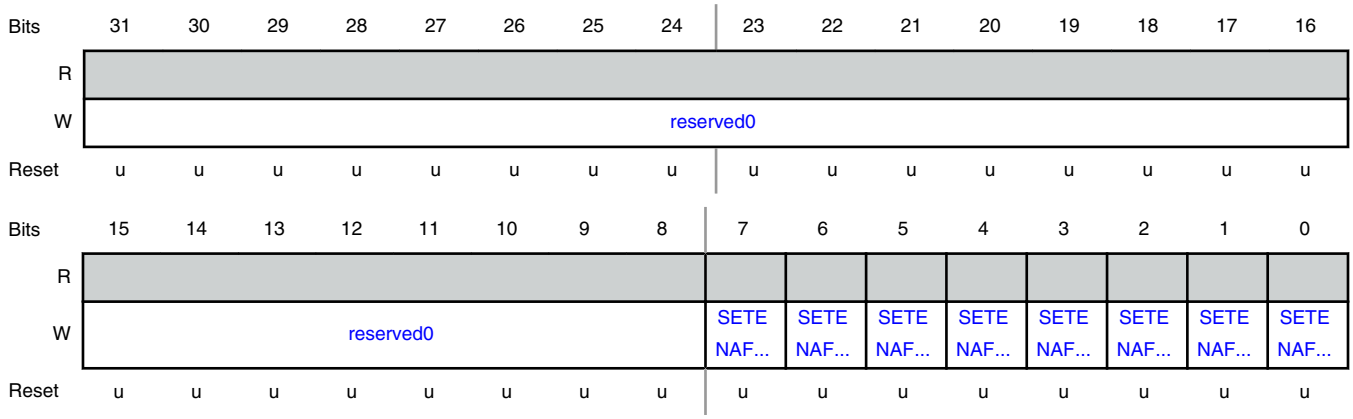
Register	Offset
SIENF	14h

Function

For each of the 8 pin interrupts selected in the INPUTMUX_PINTSELn registers, one bit in the SIENF register sets the corresponding bit in the IENF register depending on the pin interrupt mode configured in the ISEL register:

- If the pin interrupt mode is edge sensitive (ISEL[PMODE_PINn] = 0), the falling edge interrupt is set.
- If the pin interrupt mode is level sensitive (ISEL[PMODE_PINn] = 1), the HIGH-active interrupt is selected.

Diagram



Fields

Field	Function
31-8 reserved0	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
7-0 SETENAF_PINn	ENAF_PINn Bit Set Ones written to this address set ENAF_PINn in the IENF, thus enabling interrupts. 0b - No operation. 1b - Select HIGH-active interrupt or enable falling edge interrupt.

6.1.8 Pin Interrupt Active Level or Falling Edge Interrupt Clear Register (CIENF)

Offset

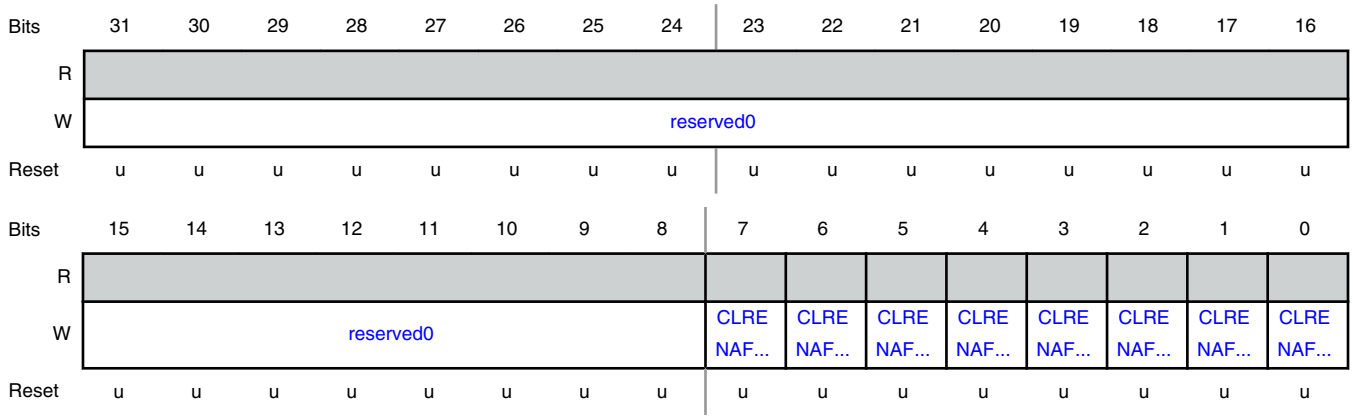
Register	Offset
CIENF	18h

Function

For each of the 8 pin interrupts selected in the INPUTMUX_PINTSELn registers, one bit in the CIENF register sets the corresponding bit in the IENF register depending on the pin interrupt mode configured in the ISEL register:

- If the pin interrupt mode is edge sensitive (ISEL[PMODE_PINn] = 0), the falling edge interrupt is cleared.
- If the pin interrupt mode is level sensitive (ISEL[PMODE_PINn] = 1), the LOW-active interrupt is selected.

Diagram



Fields

Field	Function
31-8 reserved0	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
7-0 CLRENAF_PINn	ENAF_PINn Bit Clear Ones written to this address clears ENAF_PINn in the IENF, thus disabling interrupts. 0b - No operation. 1b - LOW-active interrupt selected or falling edge interrupt disabled.

6.1.9 Pin Interrupt Rising Edge Register (RISE)

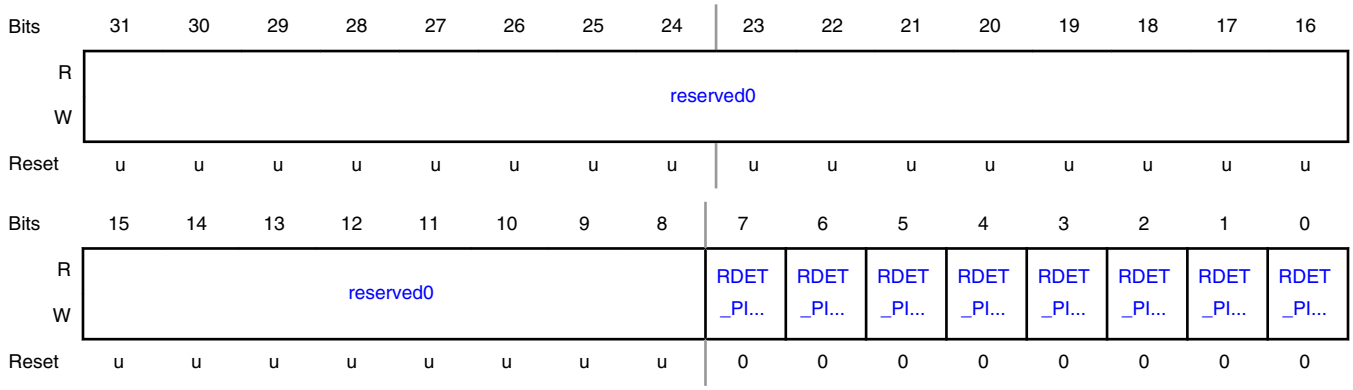
Offset

Register	Offset
RISE	1Ch

Function

This register contains bit fields for pin interrupts selected in the INPUTMUX_PINTSELn registers on which a rising edge has been detected. Writing ones to this register clears rising edge detection. If a rising edge detect status (RDET_PINn) bit is set and the corresponding interrupt enable is set then an interrupt will be generated. All edges are detected for all pins selected by the PINTSELn registers, regardless of whether they are interrupt-enabled.

Diagram



Fields

Field	Function
31-8 reserved0	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
7-0 RDET_PINn	Rising Edge Detect Status Bit n detects the rising edge of the pin selected in PINTSELn. Read 0: No rising edge has been detected on this pin since Reset or the last time a one was written to this bit. Write 0: no operation. Read 1: a rising edge has been detected since Reset or the last time a one was written to this bit. Write 1: clear rising edge detection status for this pin.

6.1.10 Pin Interrupt Falling Edge Register (FALL)

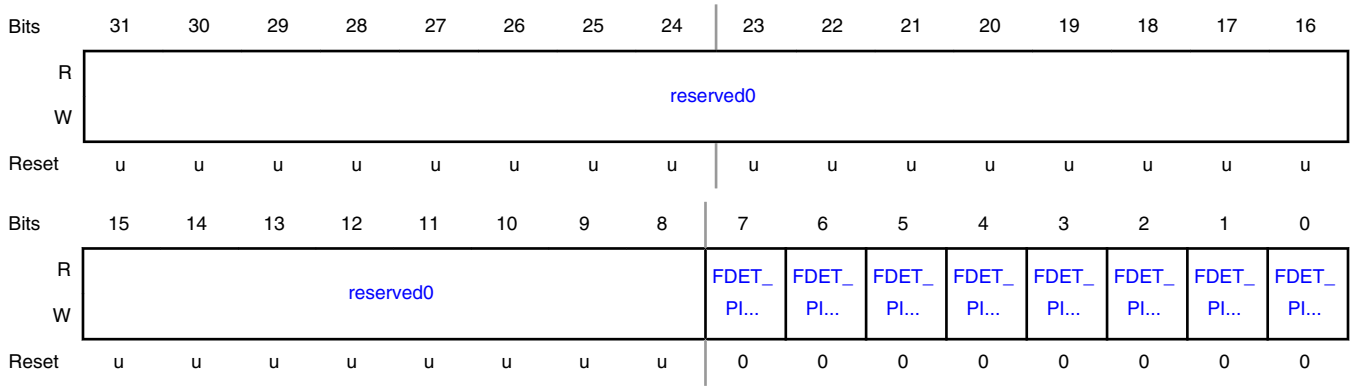
Offset

Register	Offset
FALL	20h

Function

This register contains bit fields for pin interrupts selected in the INPUTMUX_PINTSELn registers on which a falling edge has been detected. Writing ones to this register clears falling edge detection. If a falling edge detect status (FDET_PINn) bit is set and the corresponding interrupt enable is set then an interrupt will be generated. All edges are detected for all pins selected by the INPUTMUX_PINTSELn registers, regardless of whether they are interrupt-enabled.

Diagram



Fields

Field	Function
31-8 reserved0	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
7-0 FDET_PINn	Falling Edge Detect Status Bit n detects the falling edge of the pin selected in PINTSELn. Read 0: No falling edge has been detected on this pin since Reset or the last time a one was written to this bit. Write 0: no operation. Read 1: a falling edge has been detected since Reset or the last time a one was written to this bit. Write 1: clear falling edge detection status for this pin.

6.1.11 Pin Interrupt Status Register (IST)

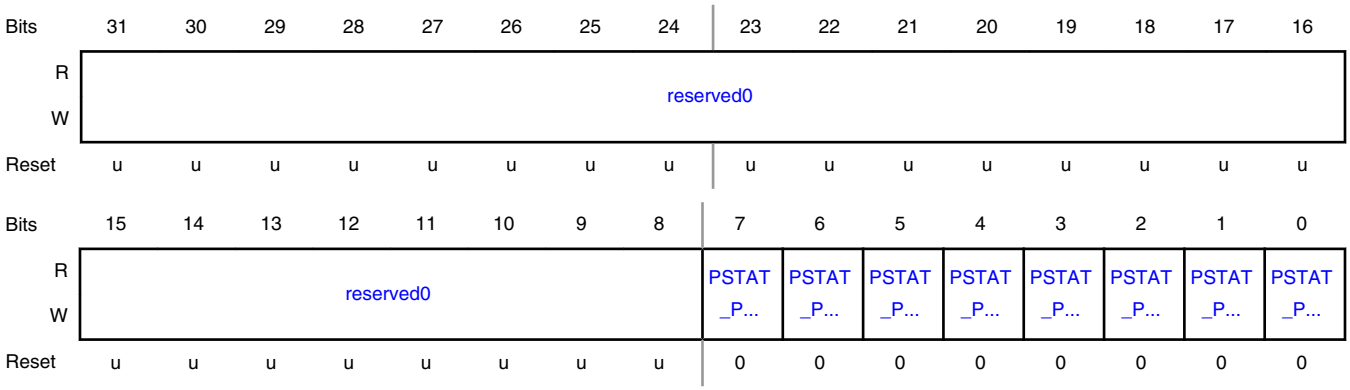
Offset

Register	Offset
IST	24h

Function

Reading this register returns ones for pin interrupts that are currently requesting an interrupt. For pins identified as edge sensitive in the ISEL register, writing ones to this register clears both rising- and falling-edge detection for the pin. For level-sensitive pins, writing ones inverts the corresponding bit in the IENF register, thus switching the active level on the pin.

Diagram



Fields

Field	Function
31-8 reserved0	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
7-0 PSTAT_PINn	Pin interrupt status. Pin interrupt status. Bit n returns the status, clears the edge interrupt, or inverts the active level of the pin 0 (selected in PINTSELn). Read 0: interrupt is not being requested for this interrupt pin. Write 0: no operation. Read 1: interrupt is being requested for this interrupt pin. Write 1 (edge-sensitive): clear rising- and falling-edge detection for this pin. Write 1 (level-sensitive): switch the active level for this pin (in the IENF register).

6.1.12 Pattern Match Interrupt Control Register (PMCTRL)

Offset

Register	Offset
PMCTRL	28h

Function

This register contains one bit to select pattern-match interrupt generation (as opposed to pin interrupts which share the same interrupt request lines), and another to enable the RXEV output to the CPU. This register also allows the current state of any pattern matches to be read.

If the pattern match feature is not used (either for interrupt generation or for RXEV assertion) bits SEL_PMATCH and ENA_RXEV of this register should be left at 0 to conserve power.

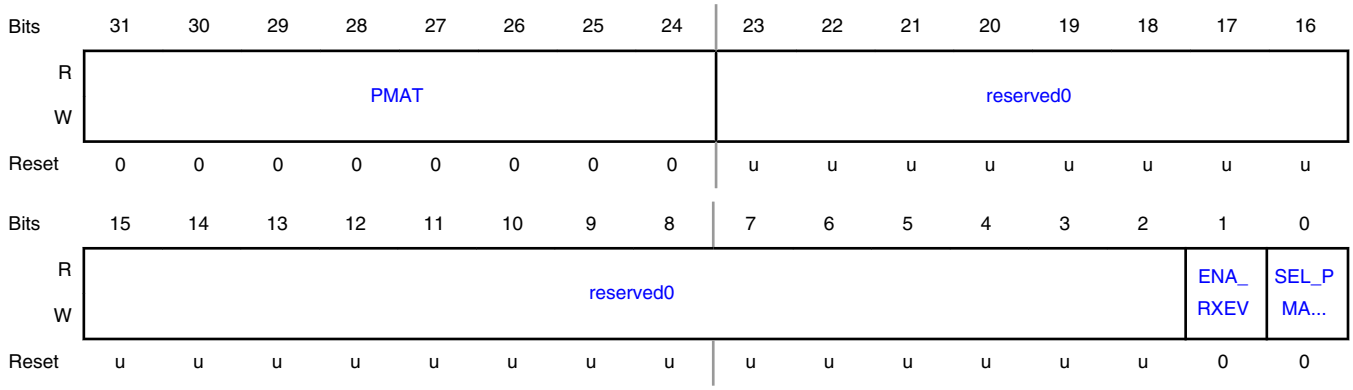
NOTE

Set up the pattern-match configuration in the PMSRC and PMCFG registers before writing to this register to enable (or re-enable) the pattern-match functionality. This eliminates the possibility of spurious interrupts as the feature is being enabled.

NOTE

The pattern match feature requires clocks in order to operate, and can thus not generate an interrupt or wake up the device during reduced power modes below sleep mode.

Diagram



Fields

Field	Function
31-24 PMAT	Pattern Matches State Display This field displays the current state of pattern matches. A 1 in any bit of this field indicates that the corresponding product term is matched by the current state of the appropriate inputs.

Table continues on the next page...

Table continued from the previous page...

Field	Function
23-2 reserved0	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
1 ENA_RXEV	RXEV Output Enable Enables the RXEV output to the CPU when the specified boolean expression evaluates to true. 0b - Disabled. RXEV output to the CPU is disabled. 1b - Enabled. RXEV output to the CPU is enabled.
0 SEL_PMATCH	8 Pin Interrupts Controll Specifies whether the 8 pin interrupts are controlled by the pin interrupt function or by the pattern match function. 0b - Pin interrupt. Interrupts are driven in response to the standard pin interrupt function. 1b - Pattern match. Interrupts are driven in response to pattern matches.

6.1.13 Pattern Match Interrupt Bit-slice Source Register (PMSRC)

Offset

Register	Offset
PMSRC	2Ch

Function

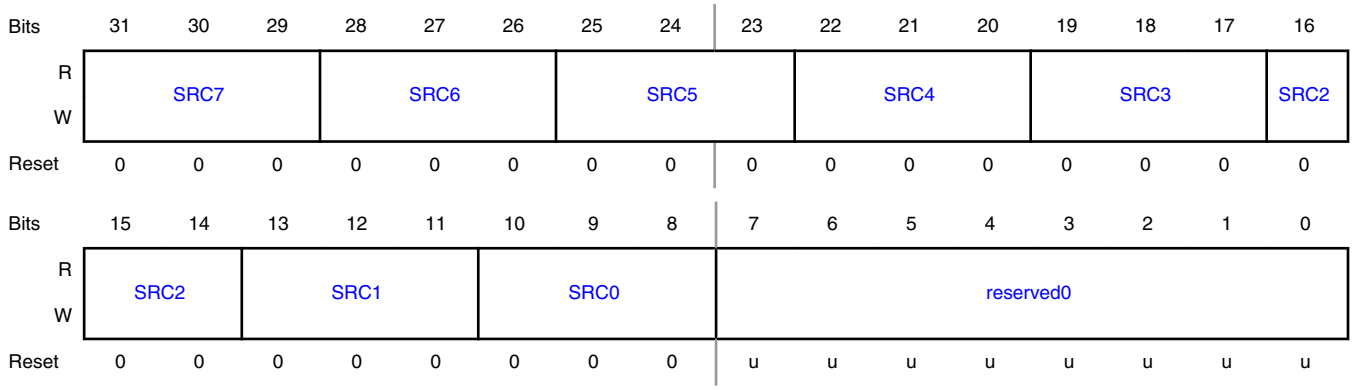
This register specifies the input source for each of the eight pattern match bit slices.

Each of the possible eight inputs is selected in the INPUTMUX_PINTSELn registers. Input 0 corresponds to the pin selected in the PINTSEL0 register, input 1 corresponds to the pin selected in the PINTSEL1 register, and so forth.

NOTE

Writing any value to either the PMCFG register or the PMSRC register, or disabling the pattern-match feature (by clearing both the SEL_PMATCH and ENA_RXEV bits in the PMCTRL register to zeros) will erase all edge-detect history.

Diagram



Fields

Field	Function
31-29: SRC7 28-26: SRC6 25-23: SRC5 22-20: SRC4 19-17: SRC3 16-14: SRC2 13-11: SRC1 10-8: SRC0	Slice n Input Source Select This field selects the input source for bit slice 0 (SRC0) to bit slice 7 (SRC7). Value X selects the pin selected in the INPUTMUX_PINTSELn register as the source to this bit slice. For example, 3 selects the pin selected in INPUTMUX_PINTSEL3 regsiter.
7-0 reserved0	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

6.1.14 Pattern Match Interrupt Bit Slice Configuration Register (PMCFG)

Offset

Register	Offset
PMCFG	30h

Function

The bit-slice configuration register configures the detect logic and contains bits to select from among eight alternative conditions for each bit slice that cause that bit slice to contribute to a pattern match. The seven LSBs of this register specify which bit-slices are the end-points of product terms in the boolean expression (i.e. where OR terms are to be inserted in the expression).

Two types of edge detection on each input are possible:

- Sticky: A rising edge, a falling edge, or a rising or falling edge that is detected at any time after the edge-detection mechanism has been cleared. The input qualifies as detected (the detect logic output remains HIGH) until the pattern match engine detect logic is cleared again.
- Non-sticky: Every time an edge (rising or falling) is detected, the detect logic output for this pin goes HIGH. This bit is cleared after one clock cycle, and the edge detect logic can detect another edge.

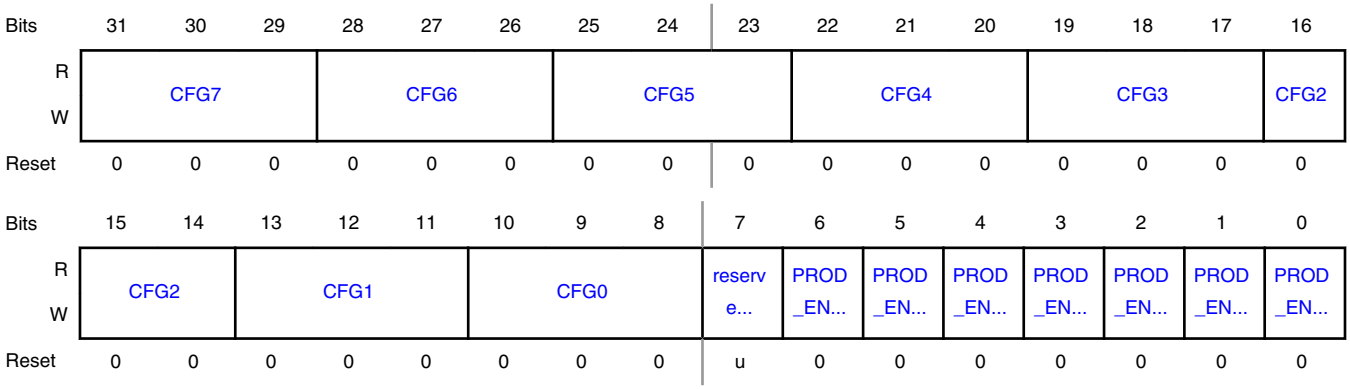
NOTE

To clear the pattern match engine detect logic, write any value to either the PMCFG register or the PMSRC register, or disable the pattern-match feature (by clearing both the SEL_PMATCH and ENA_RXEV bits in the PMCTRL register to zeros). This will erase all edge-detect history.

To select whether a slice marks the final component in a minterm of the boolean expression, write a 1 in the corresponding PROD_ENDPTSn bit. Setting a term as the final component has two effects:

1. The interrupt request associated with this bit slice will be asserted whenever a match to that product term is detected.
2. The next bit slice will start a new, independent product term in the boolean expression (i.e. an OR will be inserted in the boolean expression following the element controlled by this bit slice).

Diagram



Fields

Field	Function
31-29: CFG7 28-26: CFG6 25-23: CFG5 22-20: CFG4 19-17: CFG3 16-14: CFG2 13-11: CFG1 10-8: CFG0	Slice n Match Contribution Condition Specifies the match contribution condition for bit slice n. 000b - Constant HIGH. This bit slice always contributes to a product term match. 001b - Sticky rising edge. Match occurs if a rising edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to. 010b - Sticky falling edge. Match occurs if a falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to. 011b - Sticky rising or falling edge. Match occurs if either a rising or falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to. 100b - High level. Match (for this bit slice) occurs when there is a high level on the input specified for this bit slice in the PMSRC register. 101b - Low level. Match occurs when there is a low level on the specified input. 110b - Constant 0. This bit slice never contributes to a match (should be used to disable any unused bit slices). 111b - Event. Non-sticky rising or falling edge. Match occurs on an event - i.e. when either a rising or falling edge is first detected on the specified input (this is a non-sticky version of value 0x3). This bit is cleared after one clock cycle.
7 reserved0	RESERVED Reserved. Bit slice 7 is automatically considered a product end point.
6-0 PROD_ENDPTSn	Slice n Determines whether slice n is an endpoint. 0b - No effect. Slice n is not an endpoint. 1b - Endpoint. Slice n is the endpoint of a product term (minterm). Interrupt PINT0 in the NVIC is raised if the minterm evaluates as true.

Chapter 7
Group GPIO Input Interrupt (GINT)

7.1 GINT register descriptions

7.1.1 GINT memory map

GINT base address: 4001_1000h

Offset	Register	Width (In bits)	Access	Reset value
0h	GPIO Grouped Interrupt Control Register (CTRL)	32	RW	See description
20h	GPIO Grouped Interrupt Polarity Register (PORT_POL0)	32	RW	See description
40h	GPIO Grouped Interrupt Port Enable Register (PORT_ENA0)	32	RW	See description

7.1.2 GPIO Grouped Interrupt Control Register (CTRL)

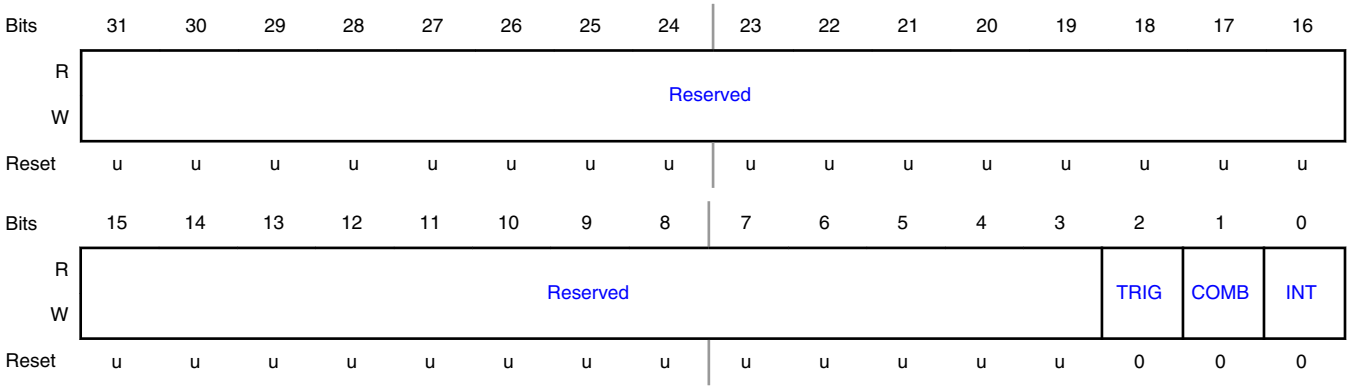
Offset

Register	Offset
CTRL	0h

Function

Grouped Interrupt (GINT) Interrupt status and configuration settings for the interrupt generation.

Diagram



Fields

Field	Function
31-3 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
2 TRIG	Group Interrupt Trigger 0b - Edge Triggered. 1b - Level Triggered.
1 COMB	Combine Enabled Inputs for Group Interrupt 0b - Or, OR functionality: A grouped interrupt is generated when any one of the enabled inputs is active (based on its programmed polarity) 1b - And, AND functionality: An interrupt is generated when all enabled bits are active (based on their programmed polarity)
0 INT	Group Interrupt Status This bit is cleared by writing a one to it. Writing zero has no effect. 0b - No request. No interrupt request is pending. 1b - Request active. Interrupt request is active.

7.1.3 GPIO Grouped Interrupt Polarity Register (PORT_POL0)

Offset

Register	Offset
PORT_POL0	20h

Function

Configure the pin polarity of each PIO signal into the group interrupt function. If a bit is low then the corresponding PIO has an active low contribution into the group interrupt. If a bit is high then the corresponding PIO has an active high contribution.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved										POL_PIO...	POL_PIO...	POL_PIO...	POL_PIO...	POL_PIO...	POL_PIO...
W																
Reset	u	u	u	u	u	u	u	u	u	u	1	1	1	1	1	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	POL_PIO...	POL_PIO...	POL_PIO...	POL_PIO...	POL_PIO...	POL_PIO...	POL_PIO9	POL_PIO8	POL_PIO7	POL_PIO6	POL_PIO5	POL_PIO4	POL_PIO3	POL_PIO2	POL_PIO1	POL_PIO0
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fields

Field	Function
31-22 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
21-0 POL_PIOn	Configure pin polarity of pin PIOn.

7.1.4 GPIO Grouped Interrupt Port Enable Register (PORT_ENA0)

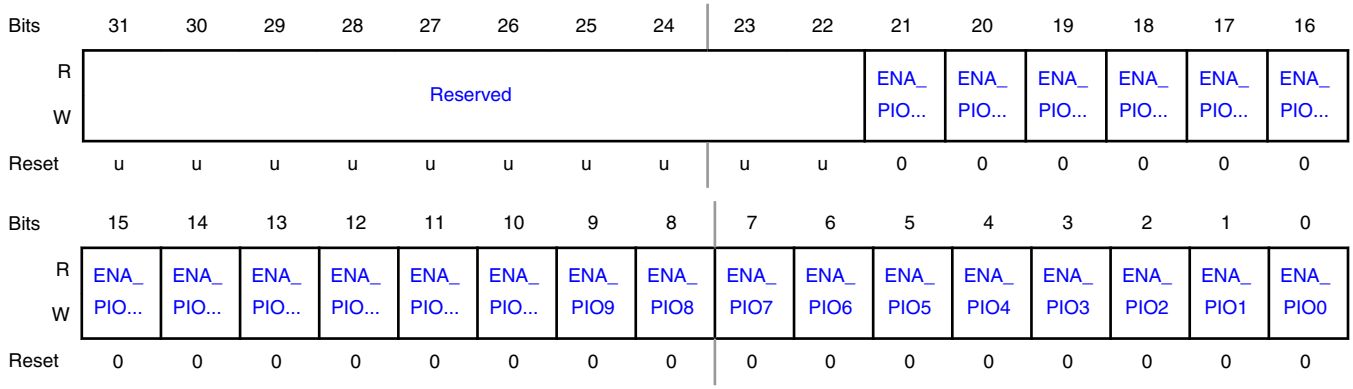
Offset

Register	Offset
PORT_ENA0	40h

Function

When a bit is set, the corresponding PIO is enabled for the group interrupt function.

Diagram



Fields

Field	Function
31-22 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
21-0 ENA_PIOn	Enable pin PIOn for group interrupt

Chapter 8
Direct Memory Access (DMA)

8.1 DMA register descriptions

8.1.1 DMA memory map

DMA base address: 4008_5000h

Offset	Register	Width (In bits)	Access	Reset value
0h	DMA Control Register (CTRL)	32	RW	See description
4h	Interrupt Status Register (INTSTAT)	32	RO	See description
8h	SRAM Base Address Register (SRAMBASE)	32	RW	See description
20h	DMA Channel Enable Set Register (ENABLESET0)	32	RW	See description
28h	DMA Channels Enable Clear Register (ENABLECLR0)	32	WO	See description
30h	DMA channel Active Status Register (ACTIVE0)	32	RO	See description
38h	DMA Channel Busy status Register (BUSY0)	32	RO	See description
40h	DMA Channel Error Interrupt Status Register (ERRINT0)	32	RW	See description
48h	DMA Channel Interrupt Enable Read and Set Register (INTENSET0)	32	RW	See description
50h	DMA Channel Interrupt Enable Clear Register (INTENCLR0)	32	WO	See description
58h	DMA Channel Interrupt A Status Register (INTA0)	32	RW	See description
60h	DMA Channel Interrupt B Status (INTB0)	32	RW	See description
68h	DMA Channel Set ValidPending Control (SETVALID0)	32	WO	See description
70h	DMA Channel Set Trigger Control Register (SETTRIG0)	32	WO	See description

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
78h	DMA Channel Abort Control Register (ABORT0)	32	WO	See description
400h	Configuration register for DMA channel 0 (CFG0)	32	RW	See description
404h	DMA Channel 0 Control and Status Register (CTLSTAT0)	32	RO	See description
408h	DMA Channel 0 Transfer Configuration Register (XFERCFG0)	32	RW	See description
410h	Configuration register for DMA channel 1 (CFG1)	32	RW	See description
414h	DMA Channel 1 Control and Status Register (CTLSTAT1)	32	RO	See description
418h	DMA Channel 1 Transfer Configuration Register (XFERCFG1)	32	RW	See description
420h	Configuration register for DMA channel 2 (CFG2)	32	RW	See description
424h	DMA Channel 2 Control and Status Register (CTLSTAT2)	32	RO	See description
428h	DMA Channel 2 Transfer Configuration Register (XFERCFG2)	32	RW	See description
430h	Configuration register for DMA channel 3 (CFG3)	32	RW	See description
434h	DMA Channel 3 Control and Status Register (CTLSTAT3)	32	RO	See description
438h	DMA Channel 3 Transfer Configuration Register (XFERCFG3)	32	RW	See description
440h	Configuration register for DMA channel 4 (CFG4)	32	RW	See description
444h	DMA Channel 4 Control and Status Register (CTLSTAT4)	32	RO	See description
448h	DMA Channel 4 Transfer Configuration Register (XFERCFG4)	32	RW	See description
450h	Configuration register for DMA channel 5 (CFG5)	32	RW	See description
454h	DMA Channel 5 Control and Status Register (CTLSTAT5)	32	RO	See description

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
458h	DMA Channel 5 Transfer Configuration Register (XFERCFG5)	32	RW	See description
460h	Configuration register for DMA channel 6 (CFG6)	32	RW	See description
464h	DMA Channel 6 Control and Status Register (CTLSTAT6)	32	RO	See description
468h	DMA Channel 6 Transfer Configuration Register (XFERCFG6)	32	RW	See description
470h	Configuration register for DMA channel 7 (CFG7)	32	RW	See description
474h	DMA Channel 7 Control and Status Register (CTLSTAT7)	32	RO	See description
478h	DMA Channel 7 Transfer Configuration Register (XFERCFG7)	32	RW	See description
480h	Configuration register for DMA channel 8 (CFG8)	32	RW	See description
484h	DMA Channel 8 Control and Status Register (CTLSTAT8)	32	RO	See description
488h	DMA Channel 8 Transfer Configuration Register (XFERCFG8)	32	RW	See description
490h	Configuration register for DMA channel 9 (CFG9)	32	RW	See description
494h	DMA Channel 9 Control and Status Register (CTLSTAT9)	32	RO	See description
498h	DMA Channel 9 Transfer Configuration Register (XFERCFG9)	32	RW	See description
4A0h	Configuration register for DMA channel 10 (CFG10)	32	RW	See description
4A4h	DMA Channel 10 Control and Status Register (CTLSTAT10)	32	RO	See description
4A8h	DMA Channel 10 Transfer Configuration Register (XFERCFG10)	32	RW	See description
4B0h	Configuration register for DMA channel 11 (CFG11)	32	RW	See description
4B4h	DMA Channel 11 Control and Status Register (CTLSTAT11)	32	RO	See description

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
4B8h	DMA Channel 11 Transfer Configuration Register (XFERCFG11)	32	RW	See description
4C0h	Configuration register for DMA channel 12 (CFG12)	32	RW	See description
4C4h	DMA Channel 12 Control and Status Register (CTLSTAT12)	32	RO	See description
4C8h	DMA Channel 12 Transfer Configuration Register (XFERCFG12)	32	RW	See description
4D0h	Configuration register for DMA channel 13 (CFG13)	32	RW	See description
4D4h	DMA Channel 13 Control and Status Register (CTLSTAT13)	32	RO	See description
4D8h	DMA Channel 13 Transfer Configuration Register (XFERCFG13)	32	RW	See description
4E0h	Configuration register for DMA channel 14 (CFG14)	32	RW	See description
4E4h	DMA Channel 14 Control and Status Register (CTLSTAT14)	32	RO	See description
4E8h	DMA Channel 14 Transfer Configuration Register (XFERCFG14)	32	RW	See description
4F0h	Configuration register for DMA channel 15 (CFG15)	32	RW	See description
4F4h	DMA Channel 15 Control and Status Register (CTLSTAT15)	32	RO	See description
4F8h	DMA Channel 15 Transfer Configuration Register (XFERCFG15)	32	RW	See description
500h	Configuration register for DMA channel 16 (CFG16)	32	RW	See description
504h	DMA Channel 16 Control and Status Register (CTLSTAT16)	32	RO	See description
508h	DMA Channel 16 Transfer Configuration Register (XFERCFG16)	32	RW	See description
510h	Configuration register for DMA channel 17 (CFG17)	32	RW	See description
514h	DMA Channel 17 Control and Status Register (CTLSTAT17)	32	RO	See description

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
518h	DMA Channel 17 Transfer Configuration Register (XFERCFG17)	32	RW	See description
520h	Configuration register for DMA channel 18 (CFG18)	32	RW	See description
524h	DMA Channel 18 Control and Status Register (CTLSTAT18)	32	RO	See description
528h	DMA Channel 18 Transfer Configuration Register (XFERCFG18)	32	RW	See description

8.1.2 DMA Control Register (CTRL)

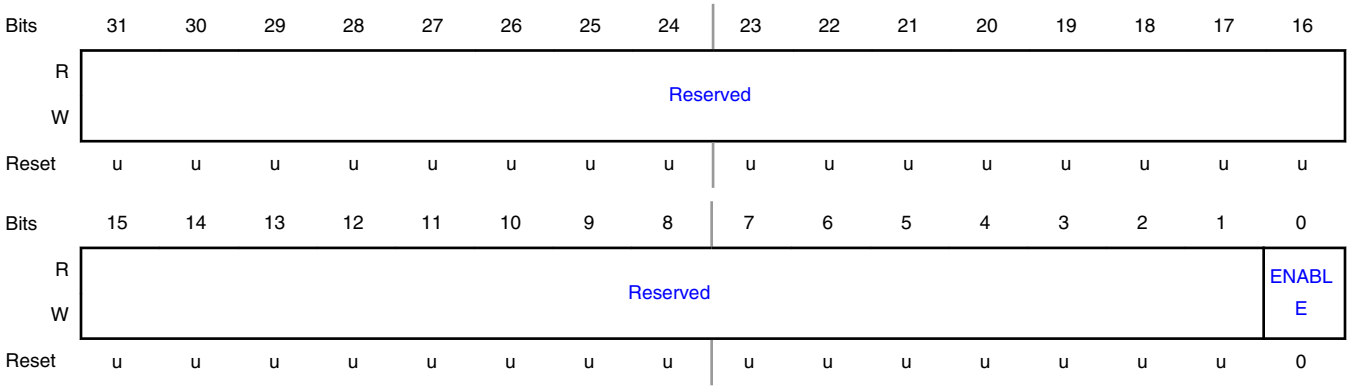
Offset

Register	Offset
CTRL	0h

Function

The CTRL register contains global the control bit for a enabling the DMA controller.

Diagram



Fields

Field	Function
31-1 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
0 ENABLE	DMA Controller Master Enable 0b - Disabled. The DMA controller is disabled. This clears any triggers that were asserted at the point when disabled, but does not prevent re-triggering when the DMA controller is re-enabled. 1b - Enabled. The DMA controller is enabled.

8.1.3 Interrupt Status Register (INTSTAT)

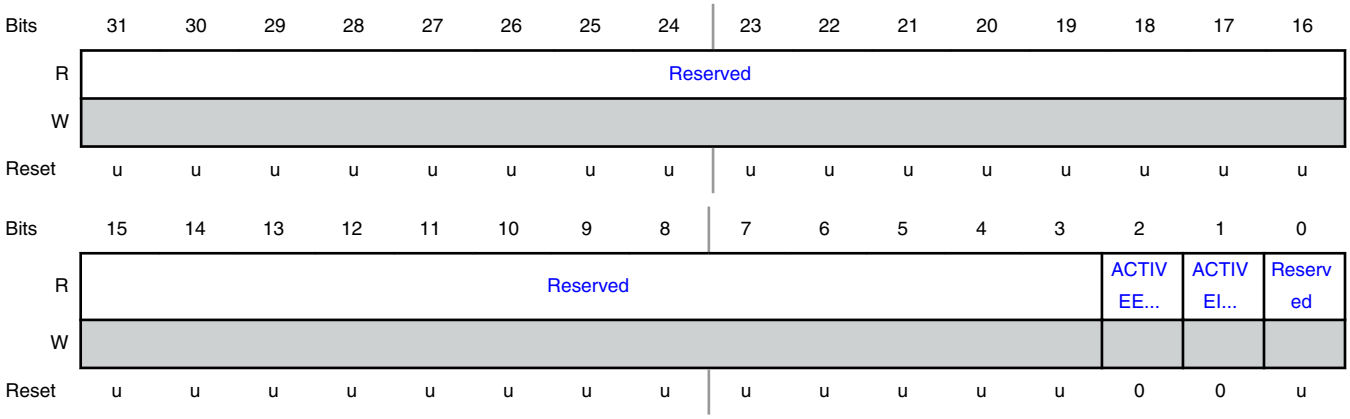
Offset

Register	Offset
INTSTAT	4h

Function

The Read-Only INTSTAT register provides an overview of DMA status. This allows quick determination of whether any enabled interrupts are pending. Details of which channels are involved are found in the interrupt type specific registers.

Diagram



Fields

Field	Function
31-3 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
2 ACTIVEERRINT	Error Interrupts Pending. Summarizes whether any error interrupts are pending. 0b - Not pending. No error interrupts are pending. 1b - Pending. At least one error interrupt is pending.
1 ACTIVEINT	Enabled Interrupts Pending Summarizes whether any enabled interrupts (other than error interrupts) are pending. 0b - Not pending. No enabled interrupts are pending. 1b - Pending. At least one enabled interrupt is pending.
0 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

8.1.4 SRAM Base Address Register (SRAMBASE)

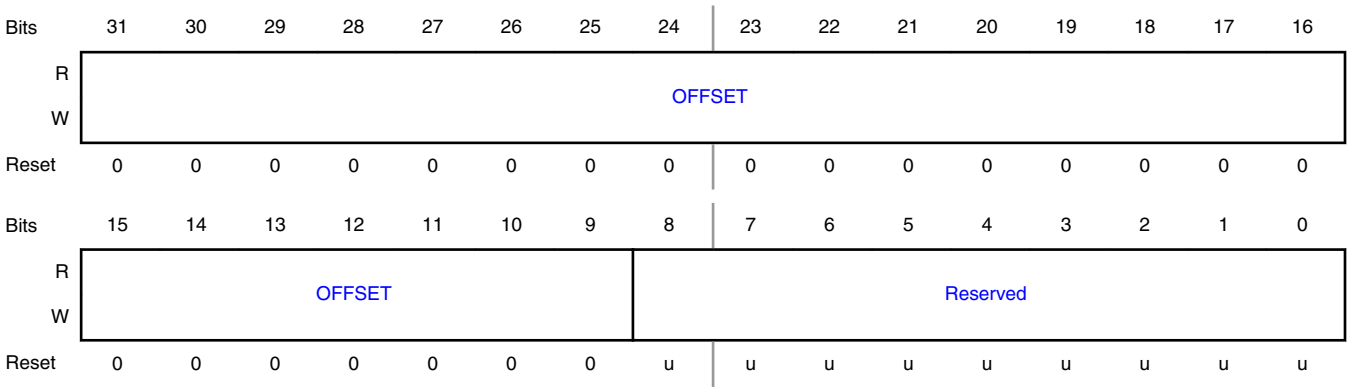
Offset

Register	Offset
SRAMBASE	8h

Function

The SRAMBASE register must be configured with an address (preferably in on-chip SRAM) where DMA descriptors will be stored. Software must set up the descriptors for those DMA channels that will be used in the application.

Diagram



Fields

Field	Function
31-9 OFFSET	DMA Descriptor Table Beginning Address Bits 31:9 Address bits 31:9 of the beginning of the DMA descriptor table. The table must begin on a 512 byte boundary. The SRAMBASE register must be configured with an address (preferably in on-chip SRAM) where DMA descriptors will be stored.
8-0 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

8.1.5 DMA Channel Enable Set Register (ENABLESET0)

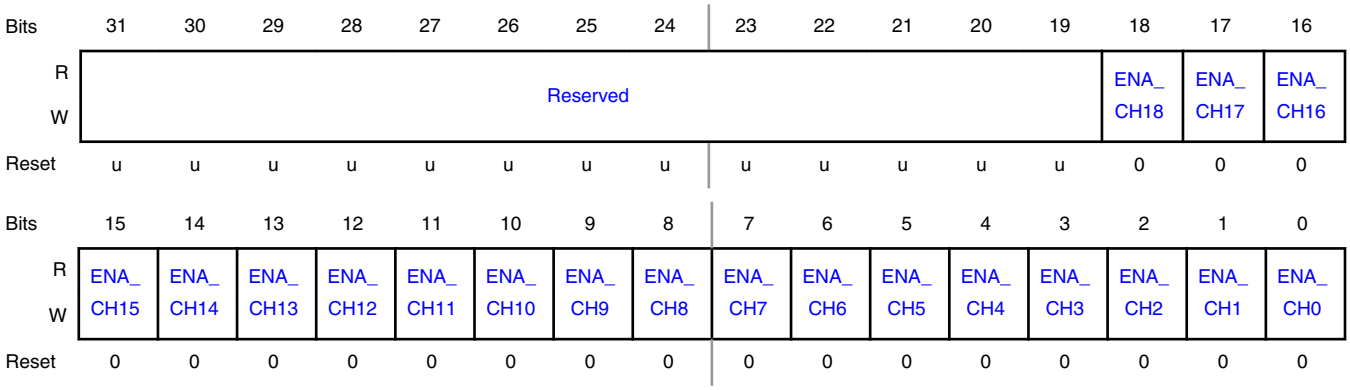
Offset

Register	Offset
ENABLESET0	20h

Function

The ENABLESET0 register determines whether each DMA channel is enabled or disabled. Disabling a DMA channel does not reset the channel in any way. A channel can be paused and restarted by clearing, then setting the Enable bit for that channel. Reading ENABLESET0 provides the current state of all of the DMA channels represented by that register. Writing a 1 to a bit position in ENABLESET0 that corresponds to an implemented DMA channel sets the bit, enabling the related DMA channel. Writing a 0 to any bit has no effect. Enables are cleared by writing to ENABLECLR0.

Diagram



Fields

Field	Function
31-19 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
18-0 ENA_CHn	Enable and Set function for DMA channel n When writing bit n, it is possible to enable the channel. 0b - No effect 1b - Enable the DMA channel When reading bit n, it shows the state of DMA channel n. Disabled. Enabled.

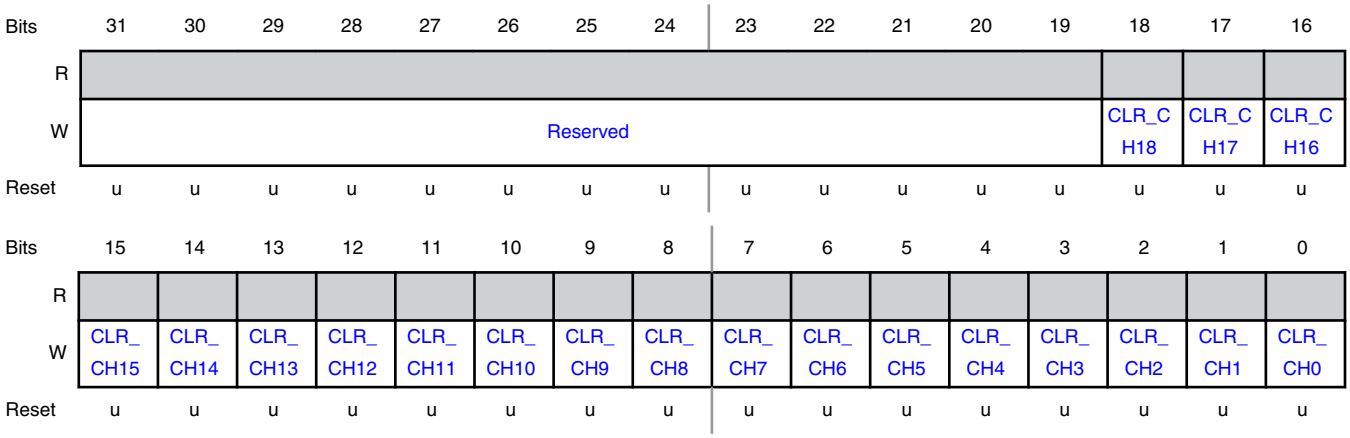
8.1.6 DMA Channels Enable Clear Register (ENABLECLR0)

Offset

Register	Offset
ENABLECLR0	28h

Function
The ENABLECLR0 register is used to clear the enable of one or more DMA channels. This register is write-only.

Diagram



Fields

Field	Function
31-19 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
18-0 CLR_CHn	ENA_CHn Clear Writing ones to this register clears the ENA_CHn bit in ENABLESET0.

8.1.7 DMA channel Active Status Register (ACTIVE0)

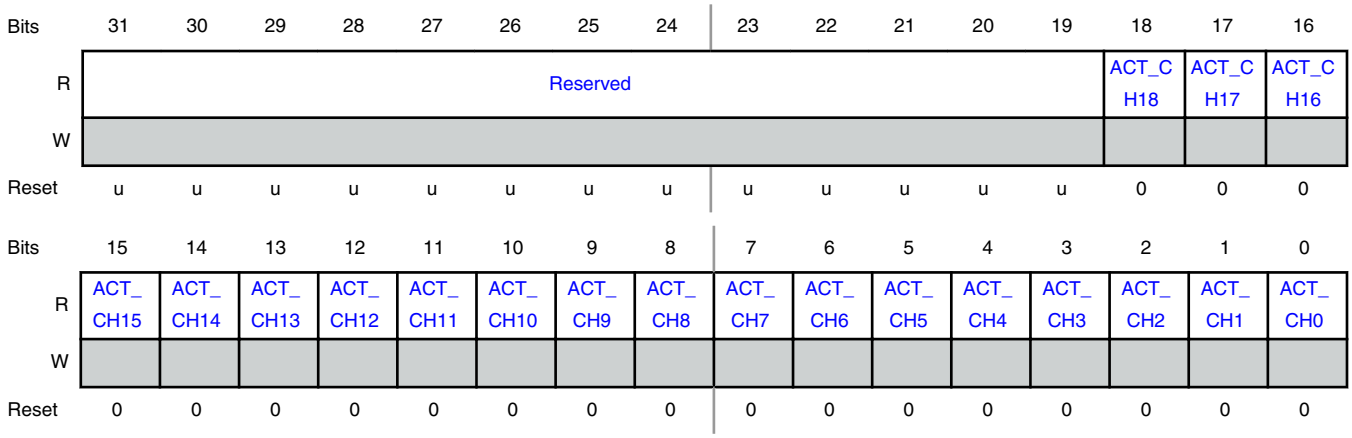
Offset

Register	Offset
ACTIVE0	30h

Function

The ACTIVE0 register indicates which DMA channels are active at the point when the read occurs. The register is read-only. A DMA channel is considered active when a DMA operation has been started but not yet fully completed. The Active status will persist from a DMA operation being started, until the pipeline is empty after end of the last descriptor (when there is no reload). An active channel may be aborted by software by setting the appropriate bit in one of the ABORT register.

Diagram



Fields

Field	Function
31-19 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
18-0 ACT_CHn	Active flag for DMA channel n Bit n corresponds to DMA channel n. 0b - Not active. 1b - Active.

8.1.8 DMA Channel Busy status Register (BUSY0)

Offset

Register	Offset
BUSY0	38h

Function

The BUSY0 register indicates which DMA channels are busy at the point when the read occurs. This registers is read-only. A DMA channel is considered busy when there is any operation related to that channel in the DMA controller's internal pipeline. This information can be used after a DMA channel is disabled by software (but still active), allowing confirmation that there are no remaining operations in progress for that channel.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved													BSY_CH18	BSY_CH17	BSY_CH16
W																
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BSY_CH15	BSY_CH14	BSY_CH13	BSY_CH12	BSY_CH11	BSY_CH10	BSY_CH9	BSY_CH8	BSY_CH7	BSY_CH6	BSY_CH5	BSY_CH4	BSY_CH3	BSY_CH2	BSY_CH1	BSY_CH0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-19 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
18-0 BSY_CHn	Busy Flag for DMA Channel n 0b - Not busy. 1b - Busy.

8.1.9 DMA Channel Error Interrupt Status Register (ERRINT0)

Offset

Register	Offset
ERRINT0	40h

Function

The ERRINT0 register contains flags for each DMA channel's Error Interrupt. Any pending interrupt flag in the register will be reflected on the DMA interrupt output.

Reading the register provides the current state of all DMA channel error interrupts. Writing a 1 to a bit position in ERRINT0 that corresponds to an implemented DMA channel clears the bit, removing the interrupt for the related DMA channel. Writing a 0 to any bit has no effect.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved													ERR_CH18	ERR_CH17	ERR_CH16
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ERR_CH15	ERR_CH14	ERR_CH13	ERR_CH12	ERR_CH11	ERR_CH10	ERR_CH9	ERR_CH8	ERR_CH7	ERR_CH6	ERR_CH5	ERR_CH4	ERR_CH3	ERR_CH2	ERR_CH1	ERR_CH0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-19 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
18-0 ERR_CHn	Error Interrupt Flag for DMA Channel n 0b - Error interrupt is not active. 1b - Error interrupt is active.

8.1.10 DMA Channel Interrupt Enable Read and Set Register (INTENSET0)

Offset

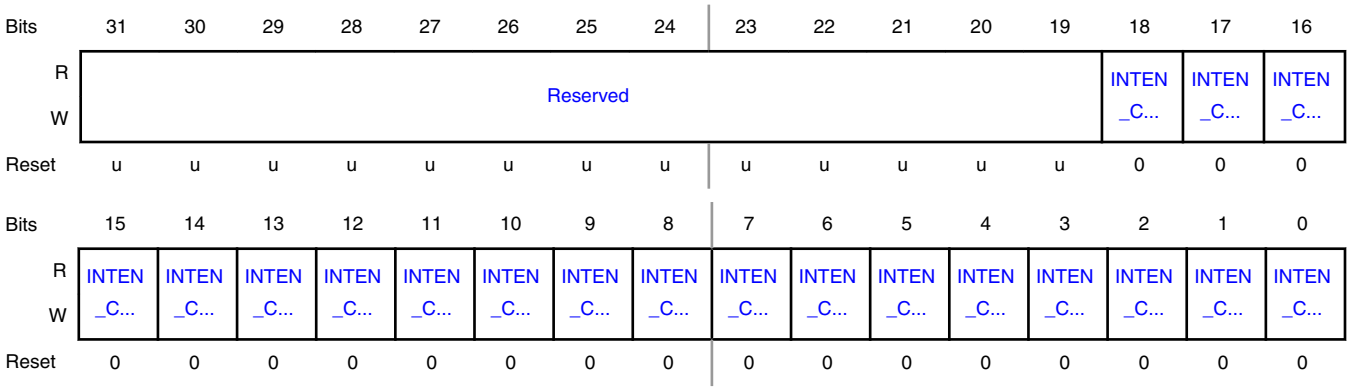
Register	Offset
INTENSET0	48h

Function

The INTENSET0 register controls whether the individual Interrupts for DMA channels contribute to the DMA interrupt output.

Reading the register provides the current state of all DMA channel interrupt enables. Writing a 1 to a bit position in INTENSET0 that corresponds to an implemented DMA channel sets the bit, enabling the interrupt for the related DMA channel. Writing a 0 to any bit has no effect. Interrupt enables are cleared by writing to INTENCLR0.

Diagram



Fields

Field	Function
31-19 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
18-0 INTEN_CHn	Interrupt Enable Read and Set for DMA Channel n 0b - Interrupt for DMA channel is disabled. 1b - Interrupt for DMA channel is enabled.

8.1.11 DMA Channel Interrupt Enable Clear Register (INTENCLR0)

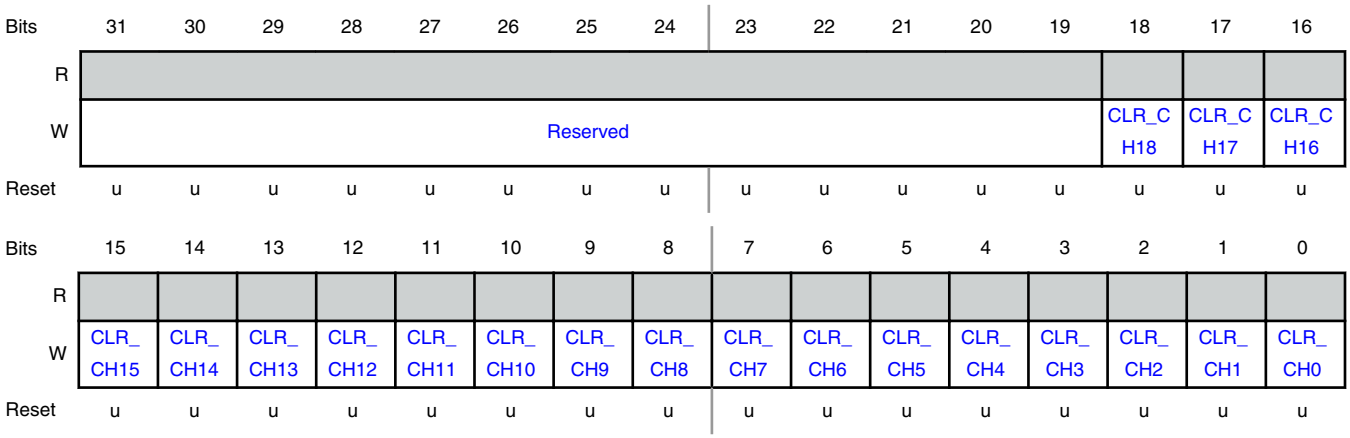
Offset

Register	Offset
INTENCLR0	50h

Function

The INTENCLR0 register is used to clear interrupt enable bits in INTENSET0. The register is write-only.

Diagram



Fields

Field	Function
31-19 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
18-0 CLR_CHn	INTEN_CHn Clear Writing ones to this register clears INTEN_CHn bits in the INTENSET0.

8.1.12 DMA Channel Interrupt A Status Register (INTA0)

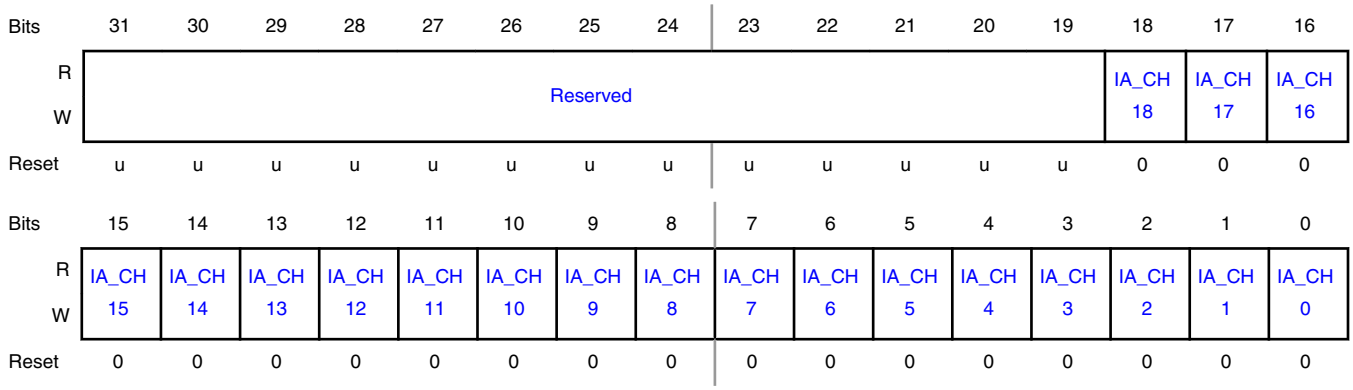
Offset

Register	Offset
INTA0	58h

Function

The INTA0 register contains the interrupt A status for each DMA channel. The status will be set when the XFERCFGn[SETINTA] bit is 1 in the transfer configuration for a channel, when the descriptor becomes exhausted. Writing a 1 to a bit in this register clears the related INTA flag. Writing 0 has no effect. Any interrupt pending status in the registers will be reflected on the DMA interrupt output if it is enabled in the related INTENSET register.

Diagram



Fields

Field	Function
31-19 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
18-0 IA_CHn	Interrupt A Status for DMA Channel n 0b - The DMA channel interrupt A is not active. 1b - The DMA channel interrupt A is active.

8.1.13 DMA Channel Interrupt B Status (INTB0)

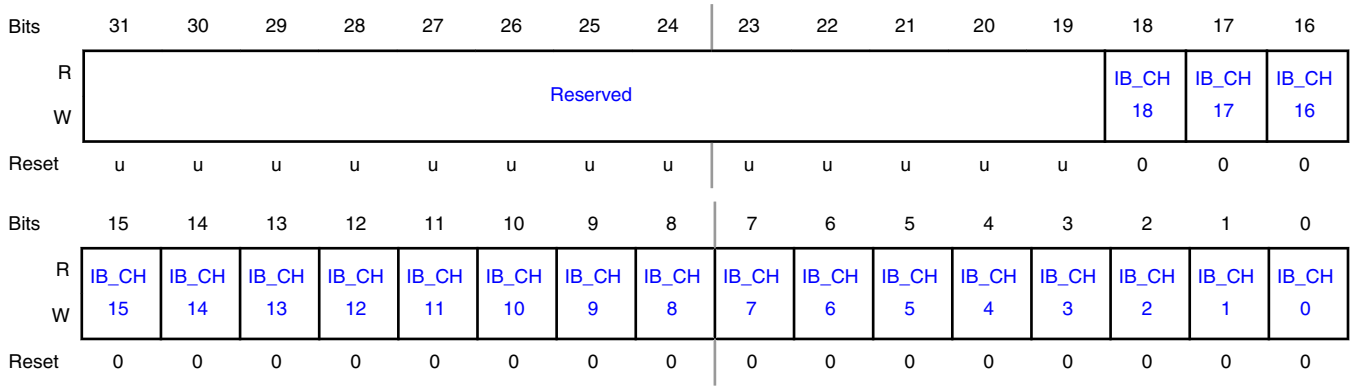
Offset

Register	Offset
INTB0	60h

Function

The INTB0 register contains the interrupt B status for each DMA channel. The status will be set when the XFERCFGn[SETINTB] bit is 1 in the transfer configuration for a channel, when the descriptor becomes exhausted. Writing a 1 to a bit in the register clears the related INTB flag. Writing 0 has no effect. Any interrupt pending status in this register will be reflected on the DMA interrupt output if it is enabled in the INTENSET register.

Diagram



Fields

Field	Function
31-19 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
18-0 IB_CHn	Interrupt B Status for DMA Channel n 0b - The DMA channel interrupt B is not active. 1b - The DMA channel interrupt B is active.

8.1.14 DMA Channel Set ValidPending Control (SETVALID0)

Offset

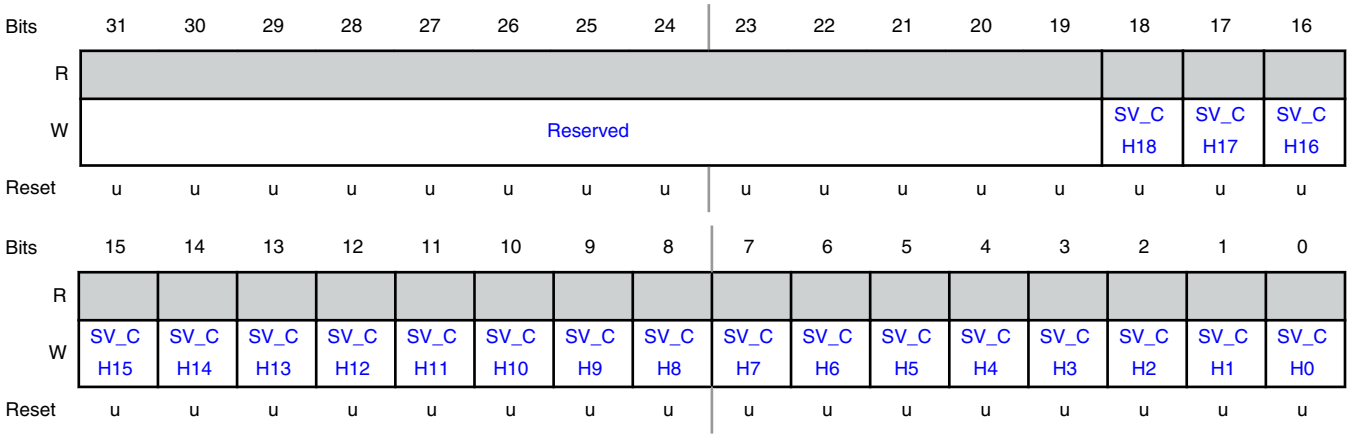
Register	Offset
SETVALID0	68h

Function

The SETVALID0 register allows setting the CTLSTATn[VALIDPENDING] bit for one or more DMA channels. This register is write-only. The XFERCFGn[CFGVALID] and SETVALID0[SV_CHn] bits allow more direct DMA block timing control by software. Each Channel Descriptor, in a sequence of descriptors, can be validated by either the setting of the XFERCFGn[CFGVALID] bit or by setting the channel's SETVALID0 register bit. Normally, the XFERCFGn[CFGVALID] bit is set. This tells the DMA that the Channel Descriptor is active and can be executed. The DMA will continue sequencing through descriptor blocks whose XFERCFGn[CFGVALID] bit are set without further software intervention. Leaving a XFERCFGn[CFGVALID] bit set to 0 allows the DMA sequence to pause at the Descriptor until software triggers the continuation. If, during DMA transmission, a Channel Descriptor is found with XFERCFGn[CFGVALID] set to 0, the DMA checks for a previously buffered SETVALID0 register bit setting for the channel. If found, the DMA will set the descriptor valid,

clear the SV_CHn setting, and resume processing the descriptor. Otherwise, the DMA pauses until the channels SETVALID0 register bit is set.

Diagram



Fields

Field	Function
31-19 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
18-0 SV_CHn	SETVALID Control for DMA Channel n 0b - No effect. 1b - Sets the VALIDPENDING control bit for DMA channel n

8.1.15 DMA Channel Set Trigger Control Register (SETTRIG0)

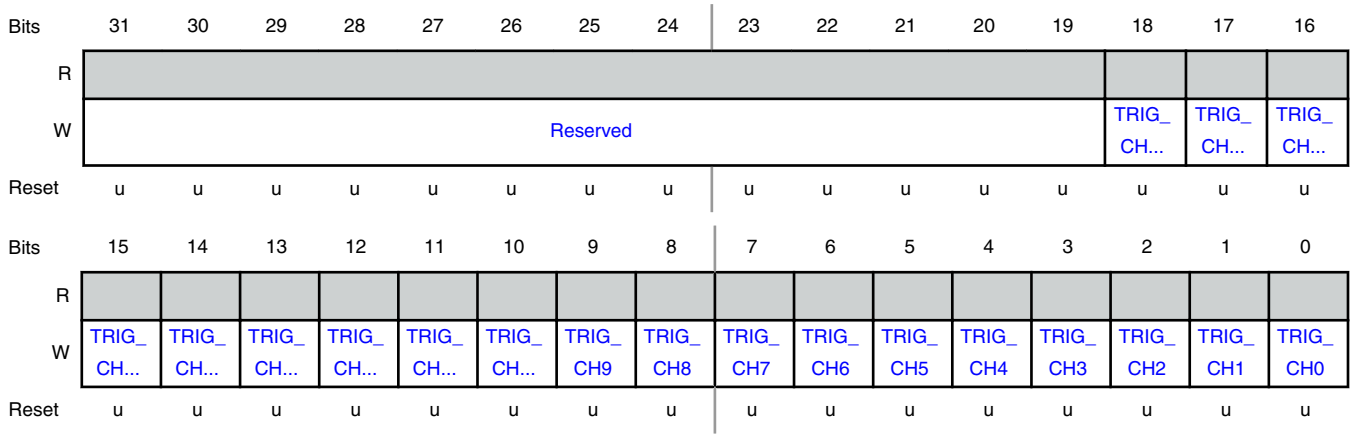
Offset

Register	Offset
SETTRIG0	70h

Function

The SETTRIG0 register allows setting the CTLSTATn[TRIG] bit in the register for one or more DMA channel. This register is write-only.

Diagram



Fields

Field	Function
31-19 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
18-0 TRIG_CHn	Set Trigger Control bit for DMA Channel n 0b - No effect. 1b - Sets the TRIG bit for DMA channel n

8.1.16 DMA Channel Abort Control Register (ABORT0)

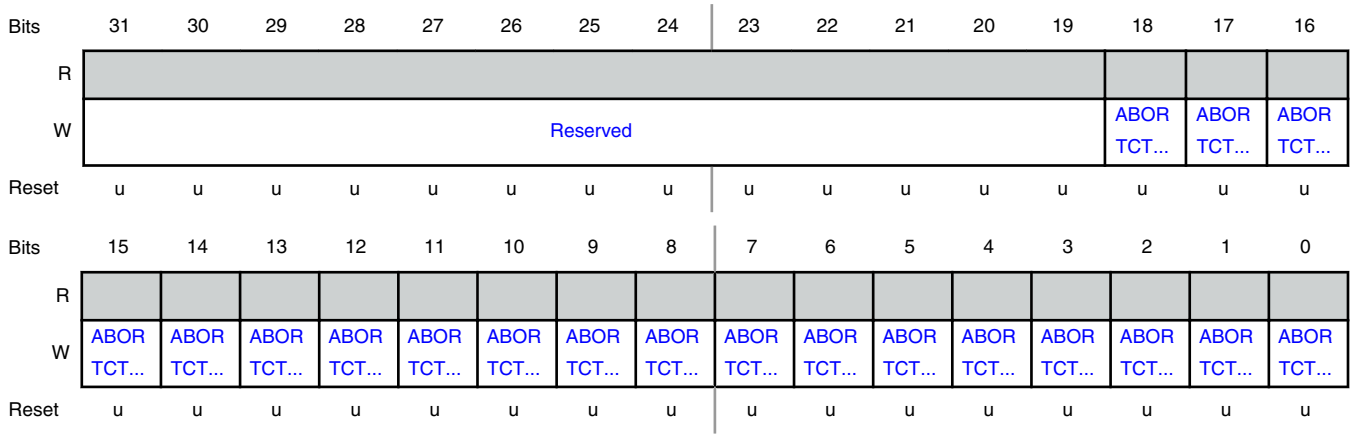
Offset

Register	Offset
ABORT0	78h

Function

The ABORT0 register allows aborting operation of a DMA channel if needed. To abort a selected channel, the channel should first be disabled by clearing the corresponding Enable bit by writing a 1 to the proper bit in the ENABLECLR0 register. Then wait until the channel is no longer busy by checking the corresponding bit in BUSY0 register. Finally, write a 1 to the proper bit of ABORT0. This prevents the channel from restarting an incomplete operation when it is enabled again. This register is write-only.

Diagram



Fields

Field	Function
31-19 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
18-0 ABORTCTRL_CHn	Abort control for DMA channel n 0b - No effect. 1b - Aborts DMA operations on channel n.

8.1.17 Configuration register for DMA channel a (CFG0 - CFG18)

Offset

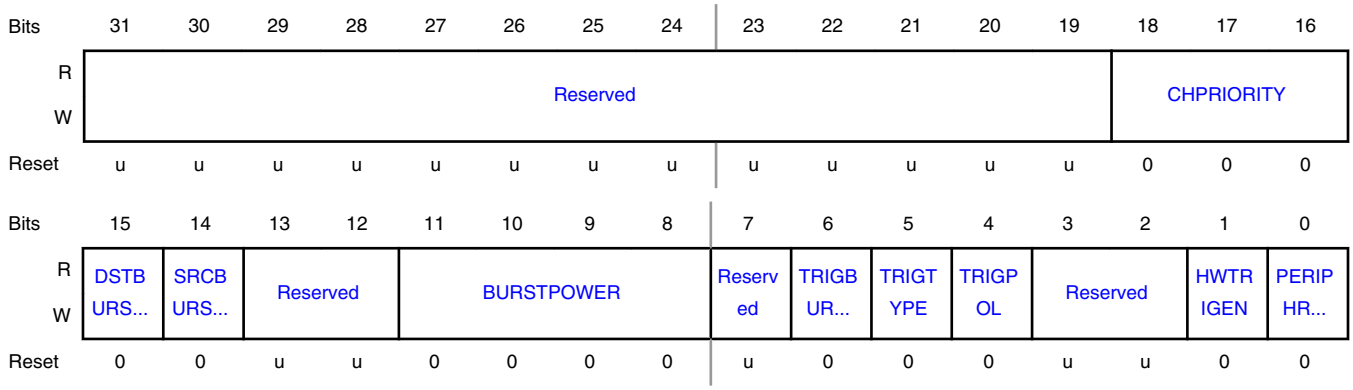
For a = 0 to 18:

Register	Offset
CFGa	400h + (a × 10h)

Function

The CFGa register contains various configuration options for DMA channel a.

Diagram



Fields

Field	Function
31-19 —	RESERVED Reserved. The value read from a reserved bit is not defined.
18-16 CHPRIORITY	Priority of this channel when multiple DMA requests are pending. Priority of this channel when multiple DMA requests are pending. Eight priority levels are supported: 000b = highest priority. 111b = lowest priority.
15 DSTBURSTWRAP	Destination Burst Wrap When enabled, the destination data address for the DMA is wrapped , meaning that the destination address range for each burst will be the same. As an example, this could be used to write several sequential registers to a peripheral for each DMA burst, writing the same registers again for each burst. 0b - Disabled. Destination burst wrapping is not enabled for this DMA channel. 1b - Enabled. Destination burst wrapping is enabled for this DMA channel.
14 SRCBURSTWRAP	Source Burst Wrap When enabled, the source data address for the DMA is wrapped , meaning that the source address range for each burst will be the same. As an example, this could be used to read several sequential registers from a peripheral for each DMA burst, reading the same registers again for each burst. 0b - Disabled. Source burst wrapping is not enabled for this DMA channel. 1b - Enabled. Source burst wrapping is enabled for this DMA channel.
13-12 —	RESERVED Reserved. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
11-8 BURSTPOWER	Burst Power Burst Power is used in two ways. It always selects the address wrap size when SRCBURSTWRAP and/or DSTBURSTWRAP modes are selected (see descriptions elsewhere in this register). When the TRIGBURST field elsewhere in this register = 1, Burst Power selects how many transfers are performed for each DMA trigger. This can be used, for example, with peripherals that contain a FIFO that can initiate a DMA operation when the FIFO reaches a certain level. The total transfer length as defined in the XFERCOUNT bits in the XFERCFG register must be an even multiple of the burst size. 0000b - Burst size = 1 (2^0). 0001b - Burst size = 2 (2^1). 0010b - Burst size = 4 (2^2). ..... 1010b - Burst size = 1024 (2^{10}). This corresponds to the maximum supported transfer count. others: Not supported.
7 —	RESERVED Reserved. The value read from a reserved bit is not defined.
6 TRIGBURST	Trigger Burst Select whether hardware triggers cause a single or burst transfer. 0b - Single transfer. Hardware trigger causes a single transfer. 1b - Burst transfer. When the trigger for this channel is set to edge triggered, a hardware trigger causes a burst transfer, as defined by BURSTPOWER. When the trigger for this channel is set to level triggered, a hardware trigger causes transfers to continue as long as the trigger is asserted, unless the transfer is complete.
5 TRIGTYPE	Trigger Type Select hardware trigger as edge triggered or level triggered. 0b - Edge. Hardware trigger is edge triggered. Transfers will be initiated and completed, as specified for a single trigger. 1b - Level. Hardware trigger is level triggered. Note that when level triggering without burst (BURSTPOWER = 0) is selected, only hardware triggers should be used on that channel. Transfers continue as long as the trigger level is asserted. Once the trigger is de-asserted, the transfer will be paused until the trigger is, again, asserted. However, the transfer will not be paused until any remaining transfers within the current BURSTPOWER length are completed.
4 TRIGPOL	Trigger Polarity Select the polarity of a hardware trigger for this channel. 0b - Active low - falling edge. Hardware trigger is active low or falling edge triggered, based on TRIGTYPE. 1b - Active high - rising edge. Hardware trigger is active high or rising edge triggered, based on TRIGTYPE.

Table continues on the next page...

Table continued from the previous page...

Field	Function
3-2 —	RESERVED Reserved. The value read from a reserved bit is not defined.
1 HWTRIGEN	Channel Hardware Triggering Enable 0b - Disabled. Hardware triggering is not used. 1b - Enabled. Use hardware triggering.
0 PERIPHREQEN	Peripheral Request Enable If a DMA channel is used to perform a memory-to-memory move, any peripheral DMA request associated with that channel can be disabled to prevent any interaction between the peripheral and the DMA controller. 0b - Disabled. Peripheral DMA requests are disabled. 1b - Enabled. Peripheral DMA requests are enabled.

8.1.18 DMA Channel a Control and Status Register (CTLSTAT0 - CTLSTAT18)

Offset

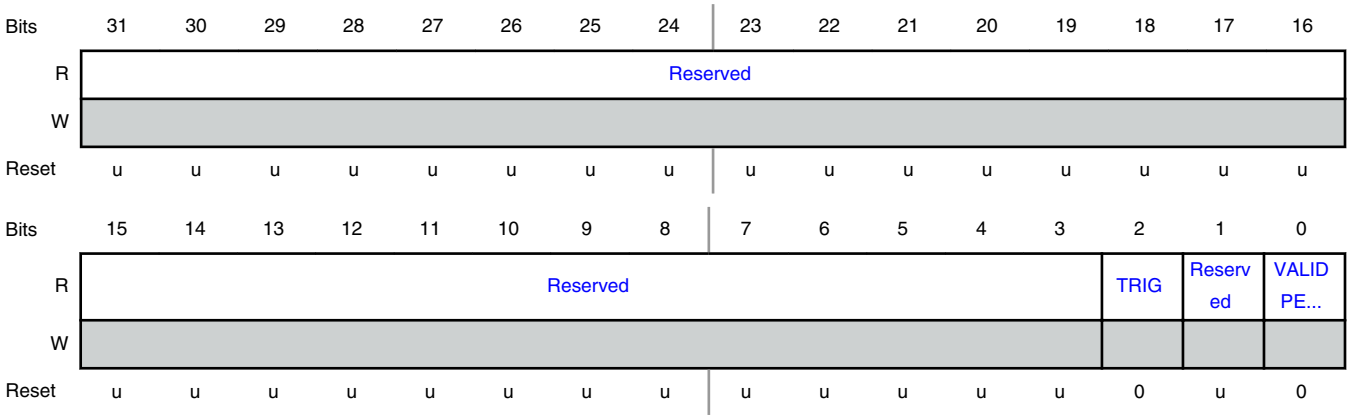
For a = 0 to 18:

Register	Offset
CTLSTATa	404h + (a × 10h)

Function

The CTLSTATa register provides status flags specific to DMA channel a. These registers are read-only.

Diagram



Fields

Field	Function
31-3 —	RESERVED Reserved. The value read from a reserved bit is not defined.
2 TRIG	Trigger Flag Indicates that the trigger for this channel is currently set. This bit is cleared at the end of an entire transfer or upon reload when CLRTRIG = 1. 0b - Not triggered. The trigger for this DMA channel is not set. DMA operations will not be carried out. 1b - Triggered. The trigger for this DMA channel is set. DMA operations will be carried out.
1 —	RESERVED Reserved. The value read from a reserved bit is not defined.
0 VALIDPENDING	Valid Pending Flag This bit is set when a 1 is written to the corresponding bit in the related SETVALID register when CFGVALID = 1 for the same channel. 0b - No effect. No effect on DMA operation. 1b - Valid pending.

8.1.19 DMA Channel 0 Transfer Configuration Register (XFER CFG0)

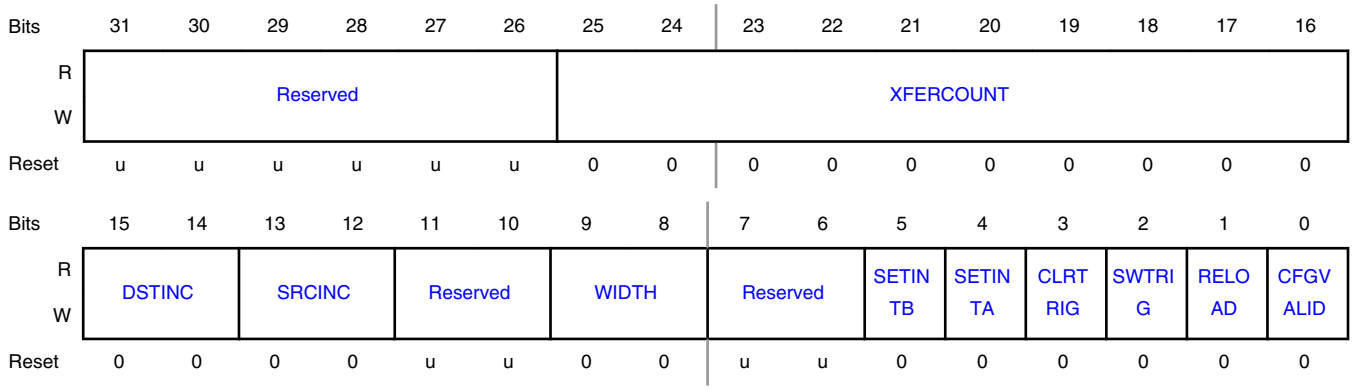
Offset

Register	Offset
XFERCFG0	408h

Function

The XFERCFG0 register contains transfer related configuration information for DMA channel 0. Using the Reload bit, this register can optionally be automatically reloaded when the current settings are exhausted (the full transfer count has been completed), allowing linked transfers with more than one descriptor to be performed.

Diagram



Fields

Field	Function
31-26 —	RESERVED Reserved. The value read from a reserved bit is not defined.
25-16 XFERCOUNT	Transfer Count Total number of transfers to be performed, minus 1 encoded. The number of bytes transferred is: (XFERCOUNT + 1) x data width (as defined by the WIDTH field). **NOTE** The DMA controller uses this bit field during transfer to count down. Hence, it cannot be used by software to read back the size of the transfer, for instance, in an interrupt handler. 0x0 - a total of 1 transfer will be performed. 0x1 - a total of 2 transfers will be performed. ..... 0x3FF - a total of 1,024 transfers will be performed.
15-14 DSTINC	Destination Address Incremented for DMA Transfer Determines whether the destination address is incremented for each DMA transfer. 00b - No increment. The destination address is not incremented for each transfer. This is the usual case when the destination is a peripheral device. 01b - 1 x width. The destination address is incremented by the amount specified by Width for each transfer. This is the usual case when the destination is memory. 10b - 2 x width. The destination address is incremented by 2 times the amount specified by Width for each transfer. 11b - 4 x width. The destination address is incremented by 4 times the amount specified by Width for each transfer.

Table continues on the next page...

Table continued from the previous page...

Field	Function
13-12 SRCINC	Source Address Incremented for DMA Transfer Determines whether the source address is incremented for each DMA transfer. 00b - No increment. The source address is not incremented for each transfer. This is the usual case when the source is a peripheral device. 01b - 1 x width. The source address is incremented by the amount specified by Width for each transfer. This is the usual case when the source is memory. 10b - 2 x width. The source address is incremented by 2 times the amount specified by Width for each transfer. 11b - 4 x width. The source address is incremented by 4 times the amount specified by Width for each transfer.
11-10 —	RESERVED Reserved. The value read from a reserved bit is not defined.
9-8 WIDTH	Transfer Width for DMA Channel Transfer width used for this DMA channel. 0x0 0x1 0x2 . 0x3 00b - 8-bit. 8-bit transfers are performed (8-bit source reads and destination writes). 01b - 16-bit. 16-bit transfers are performed (16-bit source reads and destination writes). 10b - 32-bit. 32-bit transfers are performed (32-bit source reads and destination writes). 11b - Reserved. Reserved setting, do not use.
7-6 —	RESERVED Reserved. The value read from a reserved bit is not defined.
5 SETINTB	Set Interrupt Flag B for Channel Set Interrupt flag B for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. 0b - No effect. 1b - Set. The INTB flag for this channel will be set when the current descriptor is exhausted.
4 SETINTA	Set Interrupt Flag A for Channel Set Interrupt flag A for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. 0b - No effect. 1b - Set. The INTA flag for this channel will be set when the current descriptor is exhausted.

Table continues on the next page...

Table continued from the previous page...

Field	Function
3 CLRTRIG	Clear Trigger 0b - Not cleared. The trigger is not cleared when this descriptor is exhausted. If there is a reload, the next descriptor will be started. 1b - Cleared. The trigger is cleared when this descriptor is exhausted.
2 SWTRIG	Software Trigger 0b - Not set. When written by software, the trigger for this channel is not set. A new trigger, as defined by the HWTRIGEN, TRIGPOL, and TRIGTYPE will be needed to start the channel. 1b - Set. When written by software, the trigger for this channel is set immediately. This feature should not be used with level triggering when TRIGBURST = 0.
1 RELOAD	Control Structure Reload Indicates whether the channel s control structure will be reloaded when the current descriptor is exhausted. Reloading allows ping-pong and linked transfers. 0b - Disabled. Do not reload the channels control structure when the current descriptor is exhausted. 1b - Enabled. Reload the channels control structure when the current descriptor is exhausted.
0 CFGVALID	Configuration Valid Flag This bit indicates whether the current channel descriptor is valid and can potentially be acted upon, if all other activation criteria are fulfilled. 0b - Not valid. The channel descriptor is not considered valid until validated by an associated SETVALID0 setting. 1b - Valid. The current channel descriptor is considered valid.

8.1.20 DMA Channel 1 Transfer Configuration Register (XFER CFG1)

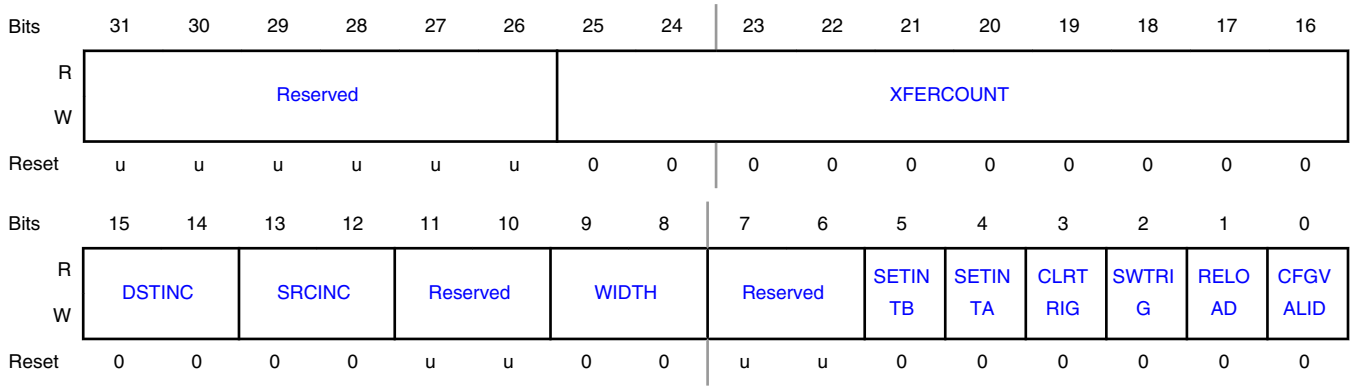
Offset

Register	Offset
XFERCFG1	418h

Function

The XFERCFG1 register contains transfer related configuration information for DMA channel 1. Using the Reload bit, this register can optionally be automatically reloaded when the current settings are exhausted (the full transfer count has been completed), allowing linked transfers with more than one descriptor to be performed.

Diagram



Fields

Field	Function
31-26 —	RESERVED Reserved. The value read from a reserved bit is not defined.
25-16 XFERCOUNT	Transfer Count Total number of transfers to be performed, minus 1 encoded. The number of bytes transferred is: (XFERCOUNT + 1) x data width (as defined by the WIDTH field). **NOTE** The DMA controller uses this bit field during transfer to count down. Hence, it cannot be used by software to read back the size of the transfer, for instance, in an interrupt handler. 0x0 - a total of 1 transfer will be performed. 0x1 - a total of 2 transfers will be performed. ..... 0x3FF - a total of 1,024 transfers will be performed.
15-14 DSTINC	Destination Address Incremented for DMA Transfer Determines whether the destination address is incremented for each DMA transfer. 00b - No increment. The destination address is not incremented for each transfer. This is the usual case when the destination is a peripheral device. 01b - 1 x width. The destination address is incremented by the amount specified by Width for each transfer. This is the usual case when the destination is memory. 10b - 2 x width. The destination address is incremented by 2 times the amount specified by Width for each transfer. 11b - 4 x width. The destination address is incremented by 4 times the amount specified by Width for each transfer.

Table continues on the next page...

Table continued from the previous page...

Field	Function
13-12 SRCINC	Source Address Incremented for DMA Transfer Determines whether the source address is incremented for each DMA transfer. 00b - No increment. The source address is not incremented for each transfer. This is the usual case when the source is a peripheral device. 01b - 1 x width. The source address is incremented by the amount specified by Width for each transfer. This is the usual case when the source is memory. 10b - 2 x width. The source address is incremented by 2 times the amount specified by Width for each transfer. 11b - 4 x width. The source address is incremented by 4 times the amount specified by Width for each transfer.
11-10 —	RESERVED Reserved. The value read from a reserved bit is not defined.
9-8 WIDTH	Transfer Width for DMA Channel Transfer width used for this DMA channel. 0x0 0x1 0x2 . 0x3 00b - 8-bit. 8-bit transfers are performed (8-bit source reads and destination writes). 01b - 16-bit. 16-bit transfers are performed (16-bit source reads and destination writes). 10b - 32-bit. 32-bit transfers are performed (32-bit source reads and destination writes). 11b - Reserved. Reserved setting, do not use.
7-6 —	RESERVED Reserved. The value read from a reserved bit is not defined.
5 SETINTB	Set Interrupt Flag B for Channel Set Interrupt flag B for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. 0b - No effect. 1b - Set. The INTB flag for this channel will be set when the current descriptor is exhausted.
4 SETINTA	Set Interrupt Flag A for Channel Set Interrupt flag A for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. 0b - No effect. 1b - Set. The INTA flag for this channel will be set when the current descriptor is exhausted.

Table continues on the next page...

Table continued from the previous page...

Field	Function
3 CLRTRIG	Clear Trigger 0b - Not cleared. The trigger is not cleared when this descriptor is exhausted. If there is a reload, the next descriptor will be started. 1b - Cleared. The trigger is cleared when this descriptor is exhausted.
2 SWTRIG	Software Trigger 0b - Not set. When written by software, the trigger for this channel is not set. A new trigger, as defined by the HWTRIGEN, TRIGPOL, and TRIGTYPE will be needed to start the channel. 1b - Set. When written by software, the trigger for this channel is set immediately. This feature should not be used with level triggering when TRIGBURST = 0.
1 RELOAD	Control Structure Reload Indicates whether the channel s control structure will be reloaded when the current descriptor is exhausted. Reloading allows ping-pong and linked transfers. 0b - Disabled. Do not reload the channels control structure when the current descriptor is exhausted. 1b - Enabled. Reload the channels control structure when the current descriptor is exhausted.
0 CFGVALID	Configuration Valid Flag This bit indicates whether the current channel descriptor is valid and can potentially be acted upon, if all other activation criteria are fulfilled. 0b - Not valid. The channel descriptor is not considered valid until validated by an associated SETVALID0 setting. 1b - Valid. The current channel descriptor is considered valid.

8.1.21 DMA Channel 2 Transfer Configuration Register (XFER CFG2)

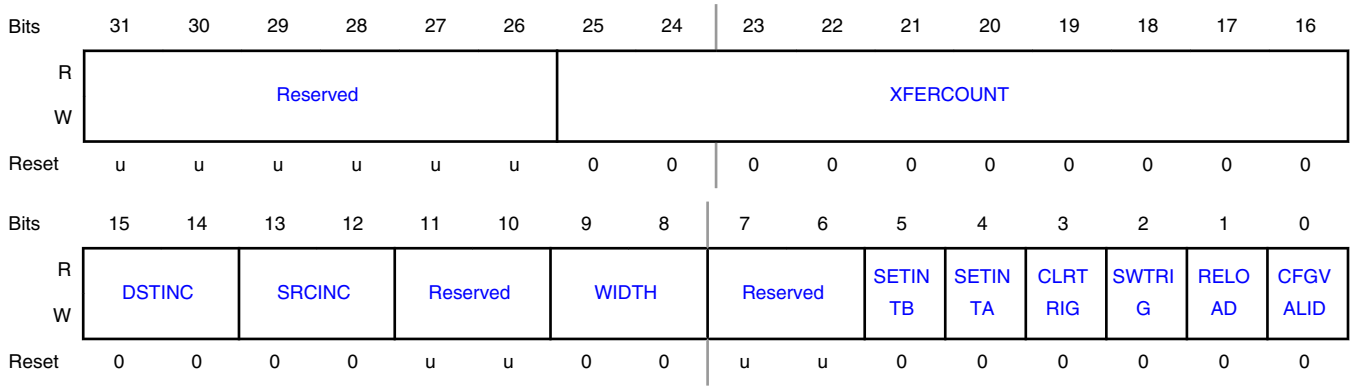
Offset

Register	Offset
XFERCFG2	428h

Function

The XFERCFG2 register contains transfer related configuration information for DMA channel 2. Using the Reload bit, this register can optionally be automatically reloaded when the current settings are exhausted (the full transfer count has been completed), allowing linked transfers with more than one descriptor to be performed.

Diagram



Fields

Field	Function
31-26 —	RESERVED Reserved. The value read from a reserved bit is not defined.
25-16 XFERCOUNT	Transfer Count Total number of transfers to be performed, minus 1 encoded. The number of bytes transferred is: (XFERCOUNT + 1) x data width (as defined by the WIDTH field). **NOTE** The DMA controller uses this bit field during transfer to count down. Hence, it cannot be used by software to read back the size of the transfer, for instance, in an interrupt handler. 0x0 - a total of 1 transfer will be performed. 0x1 - a total of 2 transfers will be performed. ..... 0x3FF - a total of 1,024 transfers will be performed.
15-14 DSTINC	Destination Address Incremented for DMA Transfer Determines whether the destination address is incremented for each DMA transfer. 00b - No increment. The destination address is not incremented for each transfer. This is the usual case when the destination is a peripheral device. 01b - 1 x width. The destination address is incremented by the amount specified by Width for each transfer. This is the usual case when the destination is memory. 10b - 2 x width. The destination address is incremented by 2 times the amount specified by Width for each transfer. 11b - 4 x width. The destination address is incremented by 4 times the amount specified by Width for each transfer.

Table continues on the next page...

Table continued from the previous page...

Field	Function
13-12 SRCINC	Source Address Incremented for DMA Transfer Determines whether the source address is incremented for each DMA transfer. 00b - No increment. The source address is not incremented for each transfer. This is the usual case when the source is a peripheral device. 01b - 1 x width. The source address is incremented by the amount specified by Width for each transfer. This is the usual case when the source is memory. 10b - 2 x width. The source address is incremented by 2 times the amount specified by Width for each transfer. 11b - 4 x width. The source address is incremented by 4 times the amount specified by Width for each transfer.
11-10 —	RESERVED Reserved. The value read from a reserved bit is not defined.
9-8 WIDTH	Transfer Width for DMA Channel Transfer width used for this DMA channel. 0x0 0x1 0x2 . 0x3 00b - 8-bit. 8-bit transfers are performed (8-bit source reads and destination writes). 01b - 16-bit. 16-bit transfers are performed (16-bit source reads and destination writes). 10b - 32-bit. 32-bit transfers are performed (32-bit source reads and destination writes). 11b - Reserved. Reserved setting, do not use.
7-6 —	RESERVED Reserved. The value read from a reserved bit is not defined.
5 SETINTB	Set Interrupt Flag B for Channel Set Interrupt flag B for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. 0b - No effect. 1b - Set. The INTB flag for this channel will be set when the current descriptor is exhausted.
4 SETINTA	Set Interrupt Flag A for Channel Set Interrupt flag A for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. 0b - No effect. 1b - Set. The INTA flag for this channel will be set when the current descriptor is exhausted.

Table continues on the next page...

Table continued from the previous page...

Field	Function
3 CLRTRIG	Clear Trigger 0b - Not cleared. The trigger is not cleared when this descriptor is exhausted. If there is a reload, the next descriptor will be started. 1b - Cleared. The trigger is cleared when this descriptor is exhausted.
2 SWTRIG	Software Trigger 0b - Not set. When written by software, the trigger for this channel is not set. A new trigger, as defined by the HWTRIGEN, TRIGPOL, and TRIGTYPE will be needed to start the channel. 1b - Set. When written by software, the trigger for this channel is set immediately. This feature should not be used with level triggering when TRIGBURST = 0.
1 RELOAD	Control Structure Reload Indicates whether the channel s control structure will be reloaded when the current descriptor is exhausted. Reloading allows ping-pong and linked transfers. 0b - Disabled. Do not reload the channels control structure when the current descriptor is exhausted. 1b - Enabled. Reload the channels control structure when the current descriptor is exhausted.
0 CFGVALID	Configuration Valid Flag This bit indicates whether the current channel descriptor is valid and can potentially be acted upon, if all other activation criteria are fulfilled. 0b - Not valid. The channel descriptor is not considered valid until validated by an associated SETVALID0 setting. 1b - Valid. The current channel descriptor is considered valid.

8.1.22 DMA Channel 3 Transfer Configuration Register (XFER CFG3)

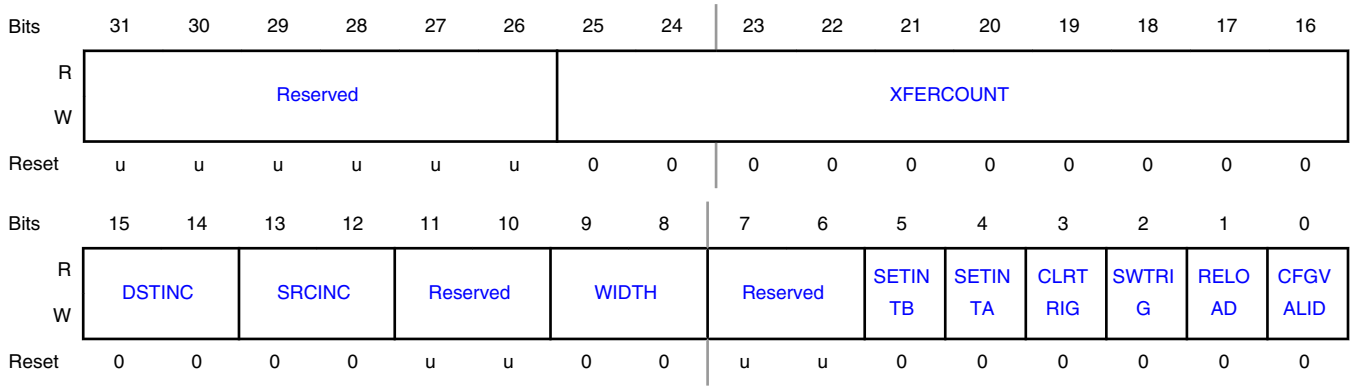
Offset

Register	Offset
XFERCFG3	438h

Function

The XFERCFG3 register contains transfer related configuration information for DMA channel 3. Using the Reload bit, this register can optionally be automatically reloaded when the current settings are exhausted (the full transfer count has been completed), allowing linked transfers with more than one descriptor to be performed.

Diagram



Fields

Field	Function
31-26 —	RESERVED Reserved. The value read from a reserved bit is not defined.
25-16 XFERCOUNT	Transfer Count Total number of transfers to be performed, minus 1 encoded. The number of bytes transferred is: (XFERCOUNT + 1) x data width (as defined by the WIDTH field). **NOTE** The DMA controller uses this bit field during transfer to count down. Hence, it cannot be used by software to read back the size of the transfer, for instance, in an interrupt handler. 0x0 - a total of 1 transfer will be performed. 0x1 - a total of 2 transfers will be performed. ..... 0x3FF - a total of 1,024 transfers will be performed.
15-14 DSTINC	Destination Address Incremented for DMA Transfer Determines whether the destination address is incremented for each DMA transfer. 00b - No increment. The destination address is not incremented for each transfer. This is the usual case when the destination is a peripheral device. 01b - 1 x width. The destination address is incremented by the amount specified by Width for each transfer. This is the usual case when the destination is memory. 10b - 2 x width. The destination address is incremented by 2 times the amount specified by Width for each transfer. 11b - 4 x width. The destination address is incremented by 4 times the amount specified by Width for each transfer.

Table continues on the next page...

Table continued from the previous page...

Field	Function
13-12 SRCINC	Source Address Incremented for DMA Transfer Determines whether the source address is incremented for each DMA transfer. 00b - No increment. The source address is not incremented for each transfer. This is the usual case when the source is a peripheral device. 01b - 1 x width. The source address is incremented by the amount specified by Width for each transfer. This is the usual case when the source is memory. 10b - 2 x width. The source address is incremented by 2 times the amount specified by Width for each transfer. 11b - 4 x width. The source address is incremented by 4 times the amount specified by Width for each transfer.
11-10 —	RESERVED Reserved. The value read from a reserved bit is not defined.
9-8 WIDTH	Transfer Width for DMA Channel Transfer width used for this DMA channel. 0x0 0x1 0x2 . 0x3 00b - 8-bit. 8-bit transfers are performed (8-bit source reads and destination writes). 01b - 16-bit. 16-bit transfers are performed (16-bit source reads and destination writes). 10b - 32-bit. 32-bit transfers are performed (32-bit source reads and destination writes). 11b - Reserved. Reserved setting, do not use.
7-6 —	RESERVED Reserved. The value read from a reserved bit is not defined.
5 SETINTB	Set Interrupt Flag B for Channel Set Interrupt flag B for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. 0b - No effect. 1b - Set. The INTB flag for this channel will be set when the current descriptor is exhausted.
4 SETINTA	Set Interrupt Flag A for Channel Set Interrupt flag A for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. 0b - No effect. 1b - Set. The INTA flag for this channel will be set when the current descriptor is exhausted.

Table continues on the next page...

Table continued from the previous page...

Field	Function
3 CLRTRIG	Clear Trigger 0b - Not cleared. The trigger is not cleared when this descriptor is exhausted. If there is a reload, the next descriptor will be started. 1b - Cleared. The trigger is cleared when this descriptor is exhausted.
2 SWTRIG	Software Trigger 0b - Not set. When written by software, the trigger for this channel is not set. A new trigger, as defined by the HWTRIGEN, TRIGPOL, and TRIGTYPE will be needed to start the channel. 1b - Set. When written by software, the trigger for this channel is set immediately. This feature should not be used with level triggering when TRIGBURST = 0.
1 RELOAD	Control Structure Reload Indicates whether the channel s control structure will be reloaded when the current descriptor is exhausted. Reloading allows ping-pong and linked transfers. 0b - Disabled. Do not reload the channels control structure when the current descriptor is exhausted. 1b - Enabled. Reload the channels control structure when the current descriptor is exhausted.
0 CFGVALID	Configuration Valid Flag This bit indicates whether the current channel descriptor is valid and can potentially be acted upon, if all other activation criteria are fulfilled. 0b - Not valid. The channel descriptor is not considered valid until validated by an associated SETVALID0 setting. 1b - Valid. The current channel descriptor is considered valid.

8.1.23 DMA Channel 4 Transfer Configuration Register (XFER CFG4)

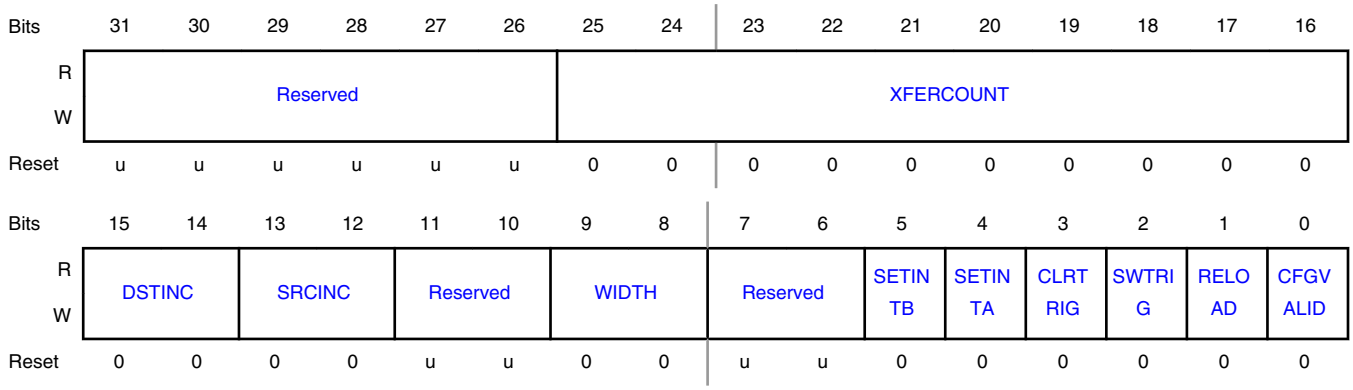
Offset

Register	Offset
XFERCFG4	448h

Function

The XFERCFG4 register contains transfer related configuration information for DMA channel 4. Using the Reload bit, this register can optionally be automatically reloaded when the current settings are exhausted (the full transfer count has been completed), allowing linked transfers with more than one descriptor to be performed.

Diagram



Fields

Field	Function
31-26 —	RESERVED Reserved. The value read from a reserved bit is not defined.
25-16 XFERCOUNT	Transfer Count Total number of transfers to be performed, minus 1 encoded. The number of bytes transferred is: (XFERCOUNT + 1) x data width (as defined by the WIDTH field). **NOTE** The DMA controller uses this bit field during transfer to count down. Hence, it cannot be used by software to read back the size of the transfer, for instance, in an interrupt handler. 0x0 - a total of 1 transfer will be performed. 0x1 - a total of 2 transfers will be performed. ..... 0x3FF - a total of 1,024 transfers will be performed.
15-14 DSTINC	Destination Address Incremented for DMA Transfer Determines whether the destination address is incremented for each DMA transfer. 00b - No increment. The destination address is not incremented for each transfer. This is the usual case when the destination is a peripheral device. 01b - 1 x width. The destination address is incremented by the amount specified by Width for each transfer. This is the usual case when the destination is memory. 10b - 2 x width. The destination address is incremented by 2 times the amount specified by Width for each transfer. 11b - 4 x width. The destination address is incremented by 4 times the amount specified by Width for each transfer.

Table continues on the next page...

Table continued from the previous page...

Field	Function
13-12 SRCINC	Source Address Incremented for DMA Transfer Determines whether the source address is incremented for each DMA transfer. 00b - No increment. The source address is not incremented for each transfer. This is the usual case when the source is a peripheral device. 01b - 1 x width. The source address is incremented by the amount specified by Width for each transfer. This is the usual case when the source is memory. 10b - 2 x width. The source address is incremented by 2 times the amount specified by Width for each transfer. 11b - 4 x width. The source address is incremented by 4 times the amount specified by Width for each transfer.
11-10 —	RESERVED Reserved. The value read from a reserved bit is not defined.
9-8 WIDTH	Transfer Width for DMA Channel Transfer width used for this DMA channel. 0x0 0x1 0x2 . 0x3 00b - 8-bit. 8-bit transfers are performed (8-bit source reads and destination writes). 01b - 16-bit. 16-bit transfers are performed (16-bit source reads and destination writes). 10b - 32-bit. 32-bit transfers are performed (32-bit source reads and destination writes). 11b - Reserved. Reserved setting, do not use.
7-6 —	RESERVED Reserved. The value read from a reserved bit is not defined.
5 SETINTB	Set Interrupt Flag B for Channel Set Interrupt flag B for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. 0b - No effect. 1b - Set. The INTB flag for this channel will be set when the current descriptor is exhausted.
4 SETINTA	Set Interrupt Flag A for Channel Set Interrupt flag A for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. 0b - No effect. 1b - Set. The INTA flag for this channel will be set when the current descriptor is exhausted.

Table continues on the next page...

Table continued from the previous page...

Field	Function
3 CLRTRIG	Clear Trigger 0b - Not cleared. The trigger is not cleared when this descriptor is exhausted. If there is a reload, the next descriptor will be started. 1b - Cleared. The trigger is cleared when this descriptor is exhausted.
2 SWTRIG	Software Trigger 0b - Not set. When written by software, the trigger for this channel is not set. A new trigger, as defined by the HWTRIGEN, TRIGPOL, and TRIGTYPE will be needed to start the channel. 1b - Set. When written by software, the trigger for this channel is set immediately. This feature should not be used with level triggering when TRIGBURST = 0.
1 RELOAD	Control Structure Reload Indicates whether the channel s control structure will be reloaded when the current descriptor is exhausted. Reloading allows ping-pong and linked transfers. 0b - Disabled. Do not reload the channels control structure when the current descriptor is exhausted. 1b - Enabled. Reload the channels control structure when the current descriptor is exhausted.
0 CFGVALID	Configuration Valid Flag This bit indicates whether the current channel descriptor is valid and can potentially be acted upon, if all other activation criteria are fulfilled. 0b - Not valid. The channel descriptor is not considered valid until validated by an associated SETVALID0 setting. 1b - Valid. The current channel descriptor is considered valid.

8.1.24 DMA Channel 5 Transfer Configuration Register (XFER CFG5)

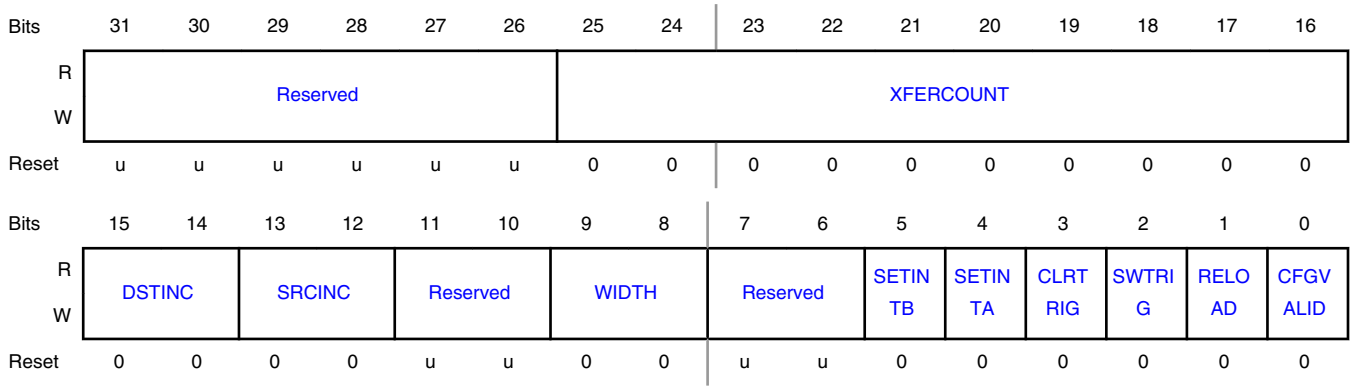
Offset

Register	Offset
XFERCFG5	458h

Function

The XFERCFG5 register contains transfer related configuration information for DMA channel 5. Using the Reload bit, this register can optionally be automatically reloaded when the current settings are exhausted (the full transfer count has been completed), allowing linked transfers with more than one descriptor to be performed.

Diagram



Fields

Field	Function
31-26 —	RESERVED Reserved. The value read from a reserved bit is not defined.
25-16 XFERCOUNT	Transfer Count Total number of transfers to be performed, minus 1 encoded. The number of bytes transferred is: (XFERCOUNT + 1) x data width (as defined by the WIDTH field). **NOTE** The DMA controller uses this bit field during transfer to count down. Hence, it cannot be used by software to read back the size of the transfer, for instance, in an interrupt handler. 0x0 - a total of 1 transfer will be performed. 0x1 - a total of 2 transfers will be performed. ..... 0x3FF - a total of 1,024 transfers will be performed.
15-14 DSTINC	Destination Address Incremented for DMA Transfer Determines whether the destination address is incremented for each DMA transfer. 00b - No increment. The destination address is not incremented for each transfer. This is the usual case when the destination is a peripheral device. 01b - 1 x width. The destination address is incremented by the amount specified by Width for each transfer. This is the usual case when the destination is memory. 10b - 2 x width. The destination address is incremented by 2 times the amount specified by Width for each transfer. 11b - 4 x width. The destination address is incremented by 4 times the amount specified by Width for each transfer.

Table continues on the next page...

Table continued from the previous page...

Field	Function
13-12 SRCINC	Source Address Incremented for DMA Transfer Determines whether the source address is incremented for each DMA transfer. 00b - No increment. The source address is not incremented for each transfer. This is the usual case when the source is a peripheral device. 01b - 1 x width. The source address is incremented by the amount specified by Width for each transfer. This is the usual case when the source is memory. 10b - 2 x width. The source address is incremented by 2 times the amount specified by Width for each transfer. 11b - 4 x width. The source address is incremented by 4 times the amount specified by Width for each transfer.
11-10 —	RESERVED Reserved. The value read from a reserved bit is not defined.
9-8 WIDTH	Transfer Width for DMA Channel Transfer width used for this DMA channel. 0x0 0x1 0x2 . 0x3 00b - 8-bit. 8-bit transfers are performed (8-bit source reads and destination writes). 01b - 16-bit. 16-bit transfers are performed (16-bit source reads and destination writes). 10b - 32-bit. 32-bit transfers are performed (32-bit source reads and destination writes). 11b - Reserved. Reserved setting, do not use.
7-6 —	RESERVED Reserved. The value read from a reserved bit is not defined.
5 SETINTB	Set Interrupt Flag B for Channel Set Interrupt flag B for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. 0b - No effect. 1b - Set. The INTB flag for this channel will be set when the current descriptor is exhausted.
4 SETINTA	Set Interrupt Flag A for Channel Set Interrupt flag A for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. 0b - No effect. 1b - Set. The INTA flag for this channel will be set when the current descriptor is exhausted.

Table continues on the next page...

Table continued from the previous page...

Field	Function
3 CLRTRIG	Clear Trigger 0b - Not cleared. The trigger is not cleared when this descriptor is exhausted. If there is a reload, the next descriptor will be started. 1b - Cleared. The trigger is cleared when this descriptor is exhausted.
2 SWTRIG	Software Trigger 0b - Not set. When written by software, the trigger for this channel is not set. A new trigger, as defined by the HWTRIGEN, TRIGPOL, and TRIGTYPE will be needed to start the channel. 1b - Set. When written by software, the trigger for this channel is set immediately. This feature should not be used with level triggering when TRIGBURST = 0.
1 RELOAD	Control Structure Reload Indicates whether the channel s control structure will be reloaded when the current descriptor is exhausted. Reloading allows ping-pong and linked transfers. 0b - Disabled. Do not reload the channels control structure when the current descriptor is exhausted. 1b - Enabled. Reload the channels control structure when the current descriptor is exhausted.
0 CFGVALID	Configuration Valid Flag This bit indicates whether the current channel descriptor is valid and can potentially be acted upon, if all other activation criteria are fulfilled. 0b - Not valid. The channel descriptor is not considered valid until validated by an associated SETVALID0 setting. 1b - Valid. The current channel descriptor is considered valid.

8.1.25 DMA Channel 6 Transfer Configuration Register (XFER CFG6)

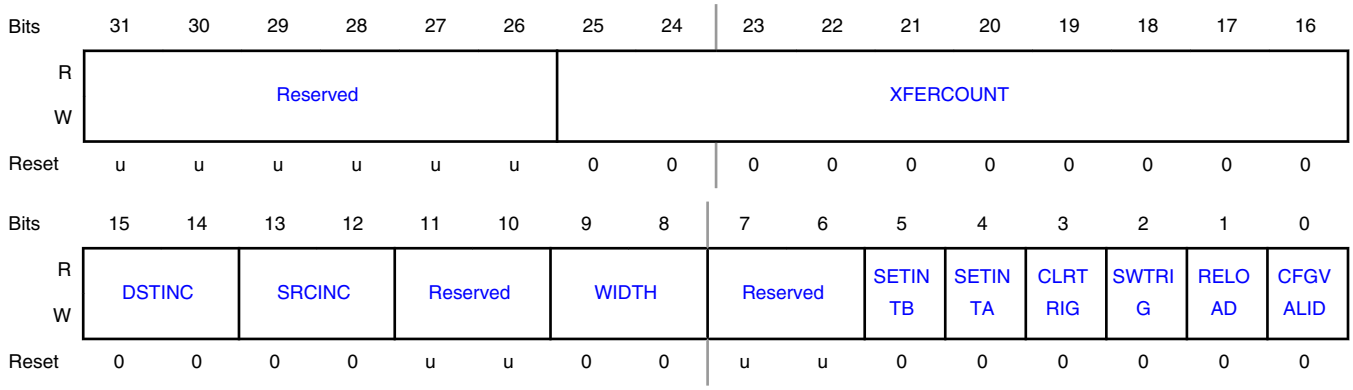
Offset

Register	Offset
XFERCFG6	468h

Function

The XFERCFG6 register contains transfer related configuration information for DMA channel 6. Using the Reload bit, this register can optionally be automatically reloaded when the current settings are exhausted (the full transfer count has been completed), allowing linked transfers with more than one descriptor to be performed.

Diagram



Fields

Field	Function
31-26 —	RESERVED Reserved. The value read from a reserved bit is not defined.
25-16 XFERCOUNT	Transfer Count Total number of transfers to be performed, minus 1 encoded. The number of bytes transferred is: (XFERCOUNT + 1) x data width (as defined by the WIDTH field). **NOTE** The DMA controller uses this bit field during transfer to count down. Hence, it cannot be used by software to read back the size of the transfer, for instance, in an interrupt handler. 0x0 - a total of 1 transfer will be performed. 0x1 - a total of 2 transfers will be performed. ..... 0x3FF - a total of 1,024 transfers will be performed.
15-14 DSTINC	Destination Address Incremented for DMA Transfer Determines whether the destination address is incremented for each DMA transfer. 00b - No increment. The destination address is not incremented for each transfer. This is the usual case when the destination is a peripheral device. 01b - 1 x width. The destination address is incremented by the amount specified by Width for each transfer. This is the usual case when the destination is memory. 10b - 2 x width. The destination address is incremented by 2 times the amount specified by Width for each transfer. 11b - 4 x width. The destination address is incremented by 4 times the amount specified by Width for each transfer.

Table continues on the next page...

Table continued from the previous page...

Field	Function
13-12 SRCINC	Source Address Incremented for DMA Transfer Determines whether the source address is incremented for each DMA transfer. 00b - No increment. The source address is not incremented for each transfer. This is the usual case when the source is a peripheral device. 01b - 1 x width. The source address is incremented by the amount specified by Width for each transfer. This is the usual case when the source is memory. 10b - 2 x width. The source address is incremented by 2 times the amount specified by Width for each transfer. 11b - 4 x width. The source address is incremented by 4 times the amount specified by Width for each transfer.
11-10 —	RESERVED Reserved. The value read from a reserved bit is not defined.
9-8 WIDTH	Transfer Width for DMA Channel Transfer width used for this DMA channel. 0x0 0x1 0x2 . 0x3 00b - 8-bit. 8-bit transfers are performed (8-bit source reads and destination writes). 01b - 16-bit. 16-bit transfers are performed (16-bit source reads and destination writes). 10b - 32-bit. 32-bit transfers are performed (32-bit source reads and destination writes). 11b - Reserved. Reserved setting, do not use.
7-6 —	RESERVED Reserved. The value read from a reserved bit is not defined.
5 SETINTB	Set Interrupt Flag B for Channel Set Interrupt flag B for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. 0b - No effect. 1b - Set. The INTB flag for this channel will be set when the current descriptor is exhausted.
4 SETINTA	Set Interrupt Flag A for Channel Set Interrupt flag A for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. 0b - No effect. 1b - Set. The INTA flag for this channel will be set when the current descriptor is exhausted.

Table continues on the next page...

Table continued from the previous page...

Field	Function
3 CLRTRIG	Clear Trigger 0b - Not cleared. The trigger is not cleared when this descriptor is exhausted. If there is a reload, the next descriptor will be started. 1b - Cleared. The trigger is cleared when this descriptor is exhausted.
2 SWTRIG	Software Trigger 0b - Not set. When written by software, the trigger for this channel is not set. A new trigger, as defined by the HWTRIGEN, TRIGPOL, and TRIGTYPE will be needed to start the channel. 1b - Set. When written by software, the trigger for this channel is set immediately. This feature should not be used with level triggering when TRIGBURST = 0.
1 RELOAD	Control Structure Reload Indicates whether the channel s control structure will be reloaded when the current descriptor is exhausted. Reloading allows ping-pong and linked transfers. 0b - Disabled. Do not reload the channels control structure when the current descriptor is exhausted. 1b - Enabled. Reload the channels control structure when the current descriptor is exhausted.
0 CFGVALID	Configuration Valid Flag This bit indicates whether the current channel descriptor is valid and can potentially be acted upon, if all other activation criteria are fulfilled. 0b - Not valid. The channel descriptor is not considered valid until validated by an associated SETVALID0 setting. 1b - Valid. The current channel descriptor is considered valid.

8.1.26 DMA Channel 7 Transfer Configuration Register (XFER CFG7)

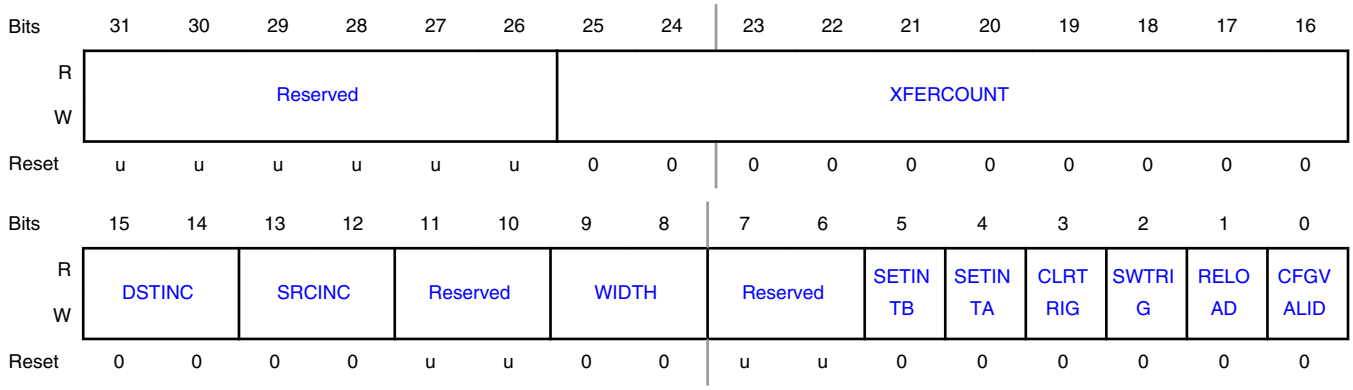
Offset

Register	Offset
XFERCFG7	478h

Function

The XFERCFG7 register contains transfer related configuration information for DMA channel 7. Using the Reload bit, this register can optionally be automatically reloaded when the current settings are exhausted (the full transfer count has been completed), allowing linked transfers with more than one descriptor to be performed.

Diagram



Fields

Field	Function
31-26 —	RESERVED Reserved. The value read from a reserved bit is not defined.
25-16 XFERCOUNT	Transfer Count Total number of transfers to be performed, minus 1 encoded. The number of bytes transferred is: (XFERCOUNT + 1) x data width (as defined by the WIDTH field). **NOTE** The DMA controller uses this bit field during transfer to count down. Hence, it cannot be used by software to read back the size of the transfer, for instance, in an interrupt handler. 0x0 - a total of 1 transfer will be performed. 0x1 - a total of 2 transfers will be performed. ..... 0x3FF - a total of 1,024 transfers will be performed.
15-14 DSTINC	Destination Address Incremented for DMA Transfer Determines whether the destination address is incremented for each DMA transfer. 00b - No increment. The destination address is not incremented for each transfer. This is the usual case when the destination is a peripheral device. 01b - 1 x width. The destination address is incremented by the amount specified by Width for each transfer. This is the usual case when the destination is memory. 10b - 2 x width. The destination address is incremented by 2 times the amount specified by Width for each transfer. 11b - 4 x width. The destination address is incremented by 4 times the amount specified by Width for each transfer.

Table continues on the next page...

Table continued from the previous page...

Field	Function
13-12 SRCINC	Source Address Incremented for DMA Transfer Determines whether the source address is incremented for each DMA transfer. 00b - No increment. The source address is not incremented for each transfer. This is the usual case when the source is a peripheral device. 01b - 1 x width. The source address is incremented by the amount specified by Width for each transfer. This is the usual case when the source is memory. 10b - 2 x width. The source address is incremented by 2 times the amount specified by Width for each transfer. 11b - 4 x width. The source address is incremented by 4 times the amount specified by Width for each transfer.
11-10 —	RESERVED Reserved. The value read from a reserved bit is not defined.
9-8 WIDTH	Transfer Width for DMA Channel Transfer width used for this DMA channel. 0x0 0x1 0x2 . 0x3 00b - 8-bit. 8-bit transfers are performed (8-bit source reads and destination writes). 01b - 16-bit. 16-bit transfers are performed (16-bit source reads and destination writes). 10b - 32-bit. 32-bit transfers are performed (32-bit source reads and destination writes). 11b - Reserved. Reserved setting, do not use.
7-6 —	RESERVED Reserved. The value read from a reserved bit is not defined.
5 SETINTB	Set Interrupt Flag B for Channel Set Interrupt flag B for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. 0b - No effect. 1b - Set. The INTB flag for this channel will be set when the current descriptor is exhausted.
4 SETINTA	Set Interrupt Flag A for Channel Set Interrupt flag A for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. 0b - No effect. 1b - Set. The INTA flag for this channel will be set when the current descriptor is exhausted.

Table continues on the next page...

Table continued from the previous page...

Field	Function
3 CLRTRIG	Clear Trigger 0b - Not cleared. The trigger is not cleared when this descriptor is exhausted. If there is a reload, the next descriptor will be started. 1b - Cleared. The trigger is cleared when this descriptor is exhausted.
2 SWTRIG	Software Trigger 0b - Not set. When written by software, the trigger for this channel is not set. A new trigger, as defined by the HWTRIGEN, TRIGPOL, and TRIGTYPE will be needed to start the channel. 1b - Set. When written by software, the trigger for this channel is set immediately. This feature should not be used with level triggering when TRIGBURST = 0.
1 RELOAD	Control Structure Reload Indicates whether the channel s control structure will be reloaded when the current descriptor is exhausted. Reloading allows ping-pong and linked transfers. 0b - Disabled. Do not reload the channels control structure when the current descriptor is exhausted. 1b - Enabled. Reload the channels control structure when the current descriptor is exhausted.
0 CFGVALID	Configuration Valid Flag This bit indicates whether the current channel descriptor is valid and can potentially be acted upon, if all other activation criteria are fulfilled. 0b - Not valid. The channel descriptor is not considered valid until validated by an associated SETVALID0 setting. 1b - Valid. The current channel descriptor is considered valid.

8.1.27 DMA Channel 8 Transfer Configuration Register (XFER CFG8)

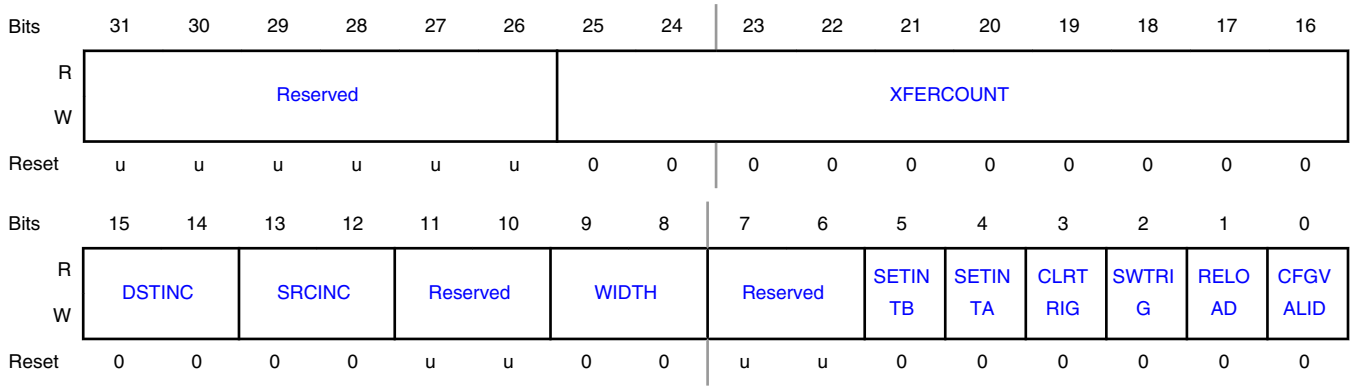
Offset

Register	Offset
XFERCFG8	488h

Function

The XFERCFG8 register contains transfer related configuration information for DMA channel 8. Using the Reload bit, this register can optionally be automatically reloaded when the current settings are exhausted (the full transfer count has been completed), allowing linked transfers with more than one descriptor to be performed.

Diagram



Fields

Field	Function
31-26 —	RESERVED Reserved. The value read from a reserved bit is not defined.
25-16 XFERCOUNT	Transfer Count Total number of transfers to be performed, minus 1 encoded. The number of bytes transferred is: (XFERCOUNT + 1) x data width (as defined by the WIDTH field). **NOTE** The DMA controller uses this bit field during transfer to count down. Hence, it cannot be used by software to read back the size of the transfer, for instance, in an interrupt handler. 0x0 - a total of 1 transfer will be performed. 0x1 - a total of 2 transfers will be performed. ..... 0x3FF - a total of 1,024 transfers will be performed.
15-14 DSTINC	Destination Address Incremented for DMA Transfer Determines whether the destination address is incremented for each DMA transfer. 00b - No increment. The destination address is not incremented for each transfer. This is the usual case when the destination is a peripheral device. 01b - 1 x width. The destination address is incremented by the amount specified by Width for each transfer. This is the usual case when the destination is memory. 10b - 2 x width. The destination address is incremented by 2 times the amount specified by Width for each transfer. 11b - 4 x width. The destination address is incremented by 4 times the amount specified by Width for each transfer.

Table continues on the next page...

Table continued from the previous page...

Field	Function
13-12 SRCINC	Source Address Incremented for DMA Transfer Determines whether the source address is incremented for each DMA transfer. 00b - No increment. The source address is not incremented for each transfer. This is the usual case when the source is a peripheral device. 01b - 1 x width. The source address is incremented by the amount specified by Width for each transfer. This is the usual case when the source is memory. 10b - 2 x width. The source address is incremented by 2 times the amount specified by Width for each transfer. 11b - 4 x width. The source address is incremented by 4 times the amount specified by Width for each transfer.
11-10 —	RESERVED Reserved. The value read from a reserved bit is not defined.
9-8 WIDTH	Transfer Width for DMA Channel Transfer width used for this DMA channel. 0x0 0x1 0x2 . 0x3 00b - 8-bit. 8-bit transfers are performed (8-bit source reads and destination writes). 01b - 16-bit. 16-bit transfers are performed (16-bit source reads and destination writes). 10b - 32-bit. 32-bit transfers are performed (32-bit source reads and destination writes). 11b - Reserved. Reserved setting, do not use.
7-6 —	RESERVED Reserved. The value read from a reserved bit is not defined.
5 SETINTB	Set Interrupt Flag B for Channel Set Interrupt flag B for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. 0b - No effect. 1b - Set. The INTB flag for this channel will be set when the current descriptor is exhausted.
4 SETINTA	Set Interrupt Flag A for Channel Set Interrupt flag A for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. 0b - No effect. 1b - Set. The INTA flag for this channel will be set when the current descriptor is exhausted.

Table continues on the next page...

Table continued from the previous page...

Field	Function
3 CLRTRIG	Clear Trigger 0b - Not cleared. The trigger is not cleared when this descriptor is exhausted. If there is a reload, the next descriptor will be started. 1b - Cleared. The trigger is cleared when this descriptor is exhausted.
2 SWTRIG	Software Trigger 0b - Not set. When written by software, the trigger for this channel is not set. A new trigger, as defined by the HWTRIGEN, TRIGPOL, and TRIGTYPE will be needed to start the channel. 1b - Set. When written by software, the trigger for this channel is set immediately. This feature should not be used with level triggering when TRIGBURST = 0.
1 RELOAD	Control Structure Reload Indicates whether the channel s control structure will be reloaded when the current descriptor is exhausted. Reloading allows ping-pong and linked transfers. 0b - Disabled. Do not reload the channels control structure when the current descriptor is exhausted. 1b - Enabled. Reload the channels control structure when the current descriptor is exhausted.
0 CFGVALID	Configuration Valid Flag This bit indicates whether the current channel descriptor is valid and can potentially be acted upon, if all other activation criteria are fulfilled. 0b - Not valid. The channel descriptor is not considered valid until validated by an associated SETVALID0 setting. 1b - Valid. The current channel descriptor is considered valid.

8.1.28 DMA Channel 9 Transfer Configuration Register (XFER CFG9)

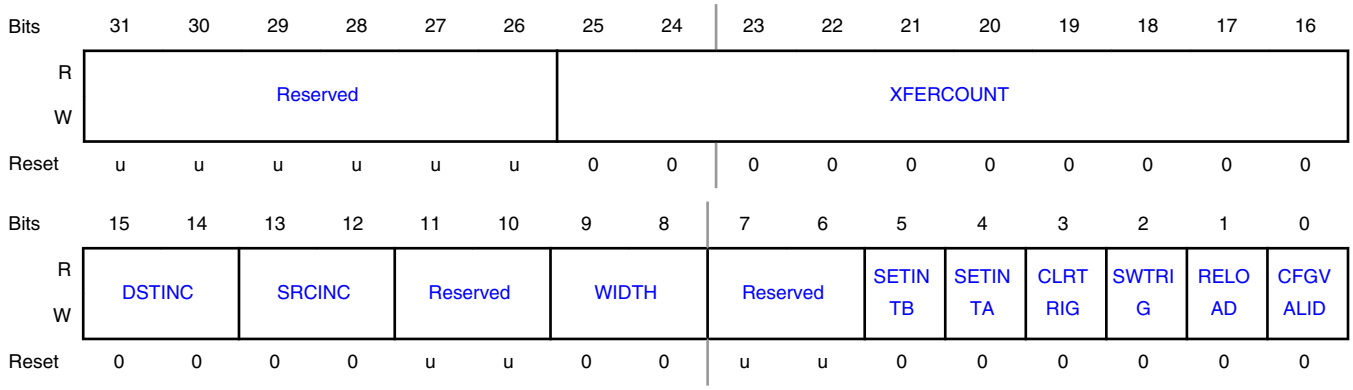
Offset

Register	Offset
XFERCFG9	498h

Function

The XFERCFG9 register contains transfer related configuration information for DMA channel 9. Using the Reload bit, this register can optionally be automatically reloaded when the current settings are exhausted (the full transfer count has been completed), allowing linked transfers with more than one descriptor to be performed.

Diagram



Fields

Field	Function
31-26 —	RESERVED Reserved. The value read from a reserved bit is not defined.
25-16 XFERCOUNT	Transfer Count Total number of transfers to be performed, minus 1 encoded. The number of bytes transferred is: (XFERCOUNT + 1) x data width (as defined by the WIDTH field). **NOTE** The DMA controller uses this bit field during transfer to count down. Hence, it cannot be used by software to read back the size of the transfer, for instance, in an interrupt handler. 0x0 - a total of 1 transfer will be performed. 0x1 - a total of 2 transfers will be performed. ..... 0x3FF - a total of 1,024 transfers will be performed.
15-14 DSTINC	Destination Address Incremented for DMA Transfer Determines whether the destination address is incremented for each DMA transfer. 00b - No increment. The destination address is not incremented for each transfer. This is the usual case when the destination is a peripheral device. 01b - 1 x width. The destination address is incremented by the amount specified by Width for each transfer. This is the usual case when the destination is memory. 10b - 2 x width. The destination address is incremented by 2 times the amount specified by Width for each transfer. 11b - 4 x width. The destination address is incremented by 4 times the amount specified by Width for each transfer.

Table continues on the next page...

Table continued from the previous page...

Field	Function
13-12 SRCINC	Source Address Incremented for DMA Transfer Determines whether the source address is incremented for each DMA transfer. 00b - No increment. The source address is not incremented for each transfer. This is the usual case when the source is a peripheral device. 01b - 1 x width. The source address is incremented by the amount specified by Width for each transfer. This is the usual case when the source is memory. 10b - 2 x width. The source address is incremented by 2 times the amount specified by Width for each transfer. 11b - 4 x width. The source address is incremented by 4 times the amount specified by Width for each transfer.
11-10 —	RESERVED Reserved. The value read from a reserved bit is not defined.
9-8 WIDTH	Transfer Width for DMA Channel Transfer width used for this DMA channel. 0x0 0x1 0x2 . 0x3 00b - 8-bit. 8-bit transfers are performed (8-bit source reads and destination writes). 01b - 16-bit. 16-bit transfers are performed (16-bit source reads and destination writes). 10b - 32-bit. 32-bit transfers are performed (32-bit source reads and destination writes). 11b - Reserved. Reserved setting, do not use.
7-6 —	RESERVED Reserved. The value read from a reserved bit is not defined.
5 SETINTB	Set Interrupt Flag B for Channel Set Interrupt flag B for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. 0b - No effect. 1b - Set. The INTB flag for this channel will be set when the current descriptor is exhausted.
4 SETINTA	Set Interrupt Flag A for Channel Set Interrupt flag A for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. 0b - No effect. 1b - Set. The INTA flag for this channel will be set when the current descriptor is exhausted.

Table continues on the next page...

Table continued from the previous page...

Field	Function
3 CLRTRIG	Clear Trigger 0b - Not cleared. The trigger is not cleared when this descriptor is exhausted. If there is a reload, the next descriptor will be started. 1b - Cleared. The trigger is cleared when this descriptor is exhausted.
2 SWTRIG	Software Trigger 0b - Not set. When written by software, the trigger for this channel is not set. A new trigger, as defined by the HWTRIGEN, TRIGPOL, and TRIGTYPE will be needed to start the channel. 1b - Set. When written by software, the trigger for this channel is set immediately. This feature should not be used with level triggering when TRIGBURST = 0.
1 RELOAD	Control Structure Reload Indicates whether the channel s control structure will be reloaded when the current descriptor is exhausted. Reloading allows ping-pong and linked transfers. 0b - Disabled. Do not reload the channels control structure when the current descriptor is exhausted. 1b - Enabled. Reload the channels control structure when the current descriptor is exhausted.
0 CFGVALID	Configuration Valid Flag This bit indicates whether the current channel descriptor is valid and can potentially be acted upon, if all other activation criteria are fulfilled. 0b - Not valid. The channel descriptor is not considered valid until validated by an associated SETVALID0 setting. 1b - Valid. The current channel descriptor is considered valid.

8.1.29 DMA Channel 10 Transfer Configuration Register (XFER CFG10)

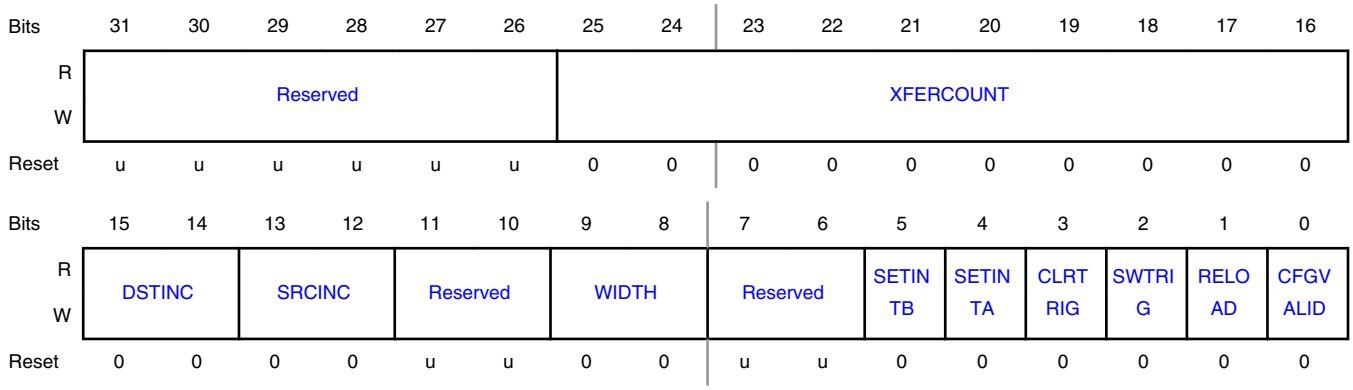
Offset

Register	Offset
XFERCFG10	4A8h

Function

The XFERCFG10 register contains transfer related configuration information for DMA channel 10. Using the Reload bit, this register can optionally be automatically reloaded when the current settings are exhausted (the full transfer count has been completed), allowing linked transfers with more than one descriptor to be performed.

Diagram



Fields

Field	Function
31-26 —	RESERVED Reserved. The value read from a reserved bit is not defined.
25-16 XFERCOUNT	Transfer Count Total number of transfers to be performed, minus 1 encoded. The number of bytes transferred is: (XFERCOUNT + 1) x data width (as defined by the WIDTH field). **NOTE** The DMA controller uses this bit field during transfer to count down. Hence, it cannot be used by software to read back the size of the transfer, for instance, in an interrupt handler. 0x0 - a total of 1 transfer will be performed. 0x1 - a total of 2 transfers will be performed. ..... 0x3FF - a total of 1,024 transfers will be performed.
15-14 DSTINC	Destination Address Incremented for DMA Transfer Determines whether the destination address is incremented for each DMA transfer. 00b - No increment. The destination address is not incremented for each transfer. This is the usual case when the destination is a peripheral device. 01b - 1 x width. The destination address is incremented by the amount specified by Width for each transfer. This is the usual case when the destination is memory. 10b - 2 x width. The destination address is incremented by 2 times the amount specified by Width for each transfer. 11b - 4 x width. The destination address is incremented by 4 times the amount specified by Width for each transfer.

Table continues on the next page...

Table continued from the previous page...

Field	Function
13-12 SRCINC	Source Address Incremented for DMA Transfer Determines whether the source address is incremented for each DMA transfer. 00b - No increment. The source address is not incremented for each transfer. This is the usual case when the source is a peripheral device. 01b - 1 x width. The source address is incremented by the amount specified by Width for each transfer. This is the usual case when the source is memory. 10b - 2 x width. The source address is incremented by 2 times the amount specified by Width for each transfer. 11b - 4 x width. The source address is incremented by 4 times the amount specified by Width for each transfer.
11-10 —	RESERVED Reserved. The value read from a reserved bit is not defined.
9-8 WIDTH	Transfer Width for DMA Channel Transfer width used for this DMA channel. 0x0 0x1 0x2 . 0x3 00b - 8-bit. 8-bit transfers are performed (8-bit source reads and destination writes). 01b - 16-bit. 16-bit transfers are performed (16-bit source reads and destination writes). 10b - 32-bit. 32-bit transfers are performed (32-bit source reads and destination writes). 11b - Reserved. Reserved setting, do not use.
7-6 —	RESERVED Reserved. The value read from a reserved bit is not defined.
5 SETINTB	Set Interrupt Flag B for Channel Set Interrupt flag B for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. 0b - No effect. 1b - Set. The INTB flag for this channel will be set when the current descriptor is exhausted.
4 SETINTA	Set Interrupt Flag A for Channel Set Interrupt flag A for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. 0b - No effect. 1b - Set. The INTA flag for this channel will be set when the current descriptor is exhausted.

Table continues on the next page...

Table continued from the previous page...

Field	Function
3 CLRTRIG	Clear Trigger 0b - Not cleared. The trigger is not cleared when this descriptor is exhausted. If there is a reload, the next descriptor will be started. 1b - Cleared. The trigger is cleared when this descriptor is exhausted.
2 SWTRIG	Software Trigger 0b - Not set. When written by software, the trigger for this channel is not set. A new trigger, as defined by the HWTRIGEN, TRIGPOL, and TRIGTYPE will be needed to start the channel. 1b - Set. When written by software, the trigger for this channel is set immediately. This feature should not be used with level triggering when TRIGBURST = 0.
1 RELOAD	Control Structure Reload Indicates whether the channel s control structure will be reloaded when the current descriptor is exhausted. Reloading allows ping-pong and linked transfers. 0b - Disabled. Do not reload the channels control structure when the current descriptor is exhausted. 1b - Enabled. Reload the channels control structure when the current descriptor is exhausted.
0 CFGVALID	Configuration Valid Flag This bit indicates whether the current channel descriptor is valid and can potentially be acted upon, if all other activation criteria are fulfilled. 0b - Not valid. The channel descriptor is not considered valid until validated by an associated SETVALID0 setting. 1b - Valid. The current channel descriptor is considered valid.

8.1.30 DMA Channel 11 Transfer Configuration Register (XFER CFG11)

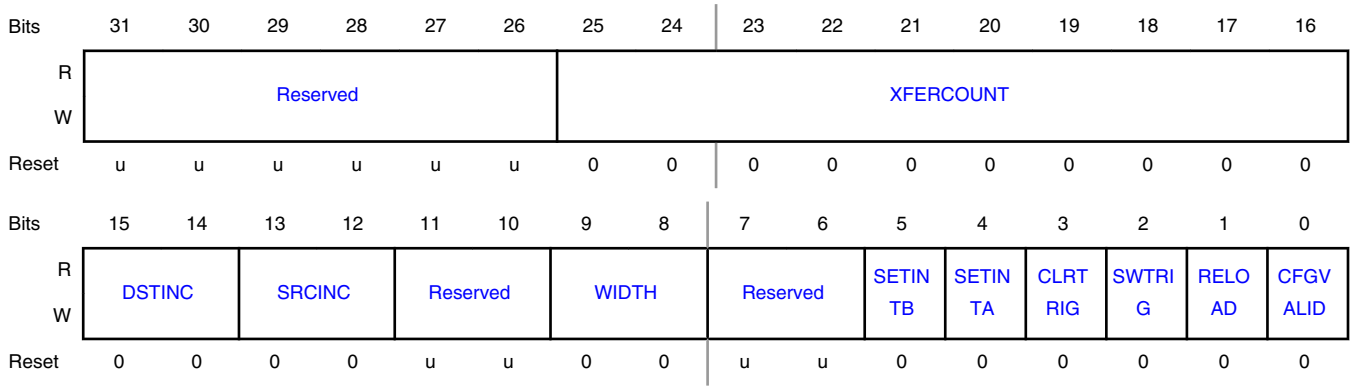
Offset

Register	Offset
XFERCFG11	4B8h

Function

The XFERCFG11 register contains transfer related configuration information for DMA channel 11. Using the Reload bit, this register can optionally be automatically reloaded when the current settings are exhausted (the full transfer count has been completed), allowing linked transfers with more than one descriptor to be performed.

Diagram



Fields

Field	Function
31-26 —	RESERVED Reserved. The value read from a reserved bit is not defined.
25-16 XFERCOUNT	Transfer Count Total number of transfers to be performed, minus 1 encoded. The number of bytes transferred is: (XFERCOUNT + 1) x data width (as defined by the WIDTH field). **NOTE** The DMA controller uses this bit field during transfer to count down. Hence, it cannot be used by software to read back the size of the transfer, for instance, in an interrupt handler. 0x0 - a total of 1 transfer will be performed. 0x1 - a total of 2 transfers will be performed. ..... 0x3FF - a total of 1,024 transfers will be performed.
15-14 DSTINC	Destination Address Incremented for DMA Transfer Determines whether the destination address is incremented for each DMA transfer. 00b - No increment. The destination address is not incremented for each transfer. This is the usual case when the destination is a peripheral device. 01b - 1 x width. The destination address is incremented by the amount specified by Width for each transfer. This is the usual case when the destination is memory. 10b - 2 x width. The destination address is incremented by 2 times the amount specified by Width for each transfer. 11b - 4 x width. The destination address is incremented by 4 times the amount specified by Width for each transfer.

Table continues on the next page...

Table continued from the previous page...

Field	Function
13-12 SRCINC	Source Address Incremented for DMA Transfer Determines whether the source address is incremented for each DMA transfer. 00b - No increment. The source address is not incremented for each transfer. This is the usual case when the source is a peripheral device. 01b - 1 x width. The source address is incremented by the amount specified by Width for each transfer. This is the usual case when the source is memory. 10b - 2 x width. The source address is incremented by 2 times the amount specified by Width for each transfer. 11b - 4 x width. The source address is incremented by 4 times the amount specified by Width for each transfer.
11-10 —	RESERVED Reserved. The value read from a reserved bit is not defined.
9-8 WIDTH	Transfer Width for DMA Channel Transfer width used for this DMA channel. 0x0 0x1 0x2 . 0x3 00b - 8-bit. 8-bit transfers are performed (8-bit source reads and destination writes). 01b - 16-bit. 16-bit transfers are performed (16-bit source reads and destination writes). 10b - 32-bit. 32-bit transfers are performed (32-bit source reads and destination writes). 11b - Reserved. Reserved setting, do not use.
7-6 —	RESERVED Reserved. The value read from a reserved bit is not defined.
5 SETINTB	Set Interrupt Flag B for Channel Set Interrupt flag B for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. 0b - No effect. 1b - Set. The INTB flag for this channel will be set when the current descriptor is exhausted.
4 SETINTA	Set Interrupt Flag A for Channel Set Interrupt flag A for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. 0b - No effect. 1b - Set. The INTA flag for this channel will be set when the current descriptor is exhausted.

Table continues on the next page...

Table continued from the previous page...

Field	Function
3 CLRTRIG	Clear Trigger 0b - Not cleared. The trigger is not cleared when this descriptor is exhausted. If there is a reload, the next descriptor will be started. 1b - Cleared. The trigger is cleared when this descriptor is exhausted.
2 SWTRIG	Software Trigger 0b - Not set. When written by software, the trigger for this channel is not set. A new trigger, as defined by the HWTRIGEN, TRIGPOL, and TRIGTYPE will be needed to start the channel. 1b - Set. When written by software, the trigger for this channel is set immediately. This feature should not be used with level triggering when TRIGBURST = 0.
1 RELOAD	Control Structure Reload Indicates whether the channel s control structure will be reloaded when the current descriptor is exhausted. Reloading allows ping-pong and linked transfers. 0b - Disabled. Do not reload the channels control structure when the current descriptor is exhausted. 1b - Enabled. Reload the channels control structure when the current descriptor is exhausted.
0 CFGVALID	Configuration Valid Flag This bit indicates whether the current channel descriptor is valid and can potentially be acted upon, if all other activation criteria are fulfilled. 0b - Not valid. The channel descriptor is not considered valid until validated by an associated SETVALID0 setting. 1b - Valid. The current channel descriptor is considered valid.

8.1.31 DMA Channel 12 Transfer Configuration Register (XFER CFG12)

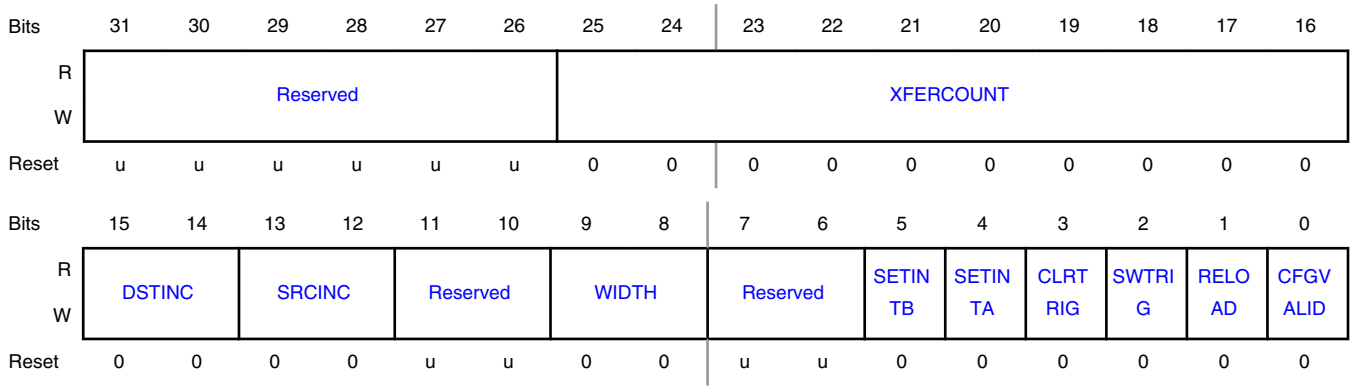
Offset

Register	Offset
XFERCFG12	4C8h

Function

The XFERCFG12 register contains transfer related configuration information for DMA channel 12. Using the Reload bit, this register can optionally be automatically reloaded when the current settings are exhausted (the full transfer count has been completed), allowing linked transfers with more than one descriptor to be performed.

Diagram



Fields

Field	Function
31-26 —	RESERVED Reserved. The value read from a reserved bit is not defined.
25-16 XFERCOUNT	Transfer Count Total number of transfers to be performed, minus 1 encoded. The number of bytes transferred is: (XFERCOUNT + 1) x data width (as defined by the WIDTH field). **NOTE** The DMA controller uses this bit field during transfer to count down. Hence, it cannot be used by software to read back the size of the transfer, for instance, in an interrupt handler. 0x0 - a total of 1 transfer will be performed. 0x1 - a total of 2 transfers will be performed. ..... 0x3FF - a total of 1,024 transfers will be performed.
15-14 DSTINC	Destination Address Incremented for DMA Transfer Determines whether the destination address is incremented for each DMA transfer. 00b - No increment. The destination address is not incremented for each transfer. This is the usual case when the destination is a peripheral device. 01b - 1 x width. The destination address is incremented by the amount specified by Width for each transfer. This is the usual case when the destination is memory. 10b - 2 x width. The destination address is incremented by 2 times the amount specified by Width for each transfer. 11b - 4 x width. The destination address is incremented by 4 times the amount specified by Width for each transfer.

Table continues on the next page...

Table continued from the previous page...

Field	Function
13-12 SRCINC	Source Address Incremented for DMA Transfer Determines whether the source address is incremented for each DMA transfer. 00b - No increment. The source address is not incremented for each transfer. This is the usual case when the source is a peripheral device. 01b - 1 x width. The source address is incremented by the amount specified by Width for each transfer. This is the usual case when the source is memory. 10b - 2 x width. The source address is incremented by 2 times the amount specified by Width for each transfer. 11b - 4 x width. The source address is incremented by 4 times the amount specified by Width for each transfer.
11-10 —	RESERVED Reserved. The value read from a reserved bit is not defined.
9-8 WIDTH	Transfer Width for DMA Channel Transfer width used for this DMA channel. 0x0 0x1 0x2 . 0x3 00b - 8-bit. 8-bit transfers are performed (8-bit source reads and destination writes). 01b - 16-bit. 16-bit transfers are performed (16-bit source reads and destination writes). 10b - 32-bit. 32-bit transfers are performed (32-bit source reads and destination writes). 11b - Reserved. Reserved setting, do not use.
7-6 —	RESERVED Reserved. The value read from a reserved bit is not defined.
5 SETINTB	Set Interrupt Flag B for Channel Set Interrupt flag B for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. 0b - No effect. 1b - Set. The INTB flag for this channel will be set when the current descriptor is exhausted.
4 SETINTA	Set Interrupt Flag A for Channel Set Interrupt flag A for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. 0b - No effect. 1b - Set. The INTA flag for this channel will be set when the current descriptor is exhausted.

Table continues on the next page...

Table continued from the previous page...

Field	Function
3 CLRTRIG	Clear Trigger 0b - Not cleared. The trigger is not cleared when this descriptor is exhausted. If there is a reload, the next descriptor will be started. 1b - Cleared. The trigger is cleared when this descriptor is exhausted.
2 SWTRIG	Software Trigger 0b - Not set. When written by software, the trigger for this channel is not set. A new trigger, as defined by the HWTRIGEN, TRIGPOL, and TRIGTYPE will be needed to start the channel. 1b - Set. When written by software, the trigger for this channel is set immediately. This feature should not be used with level triggering when TRIGBURST = 0.
1 RELOAD	Control Structure Reload Indicates whether the channel s control structure will be reloaded when the current descriptor is exhausted. Reloading allows ping-pong and linked transfers. 0b - Disabled. Do not reload the channels control structure when the current descriptor is exhausted. 1b - Enabled. Reload the channels control structure when the current descriptor is exhausted.
0 CFGVALID	Configuration Valid Flag This bit indicates whether the current channel descriptor is valid and can potentially be acted upon, if all other activation criteria are fulfilled. 0b - Not valid. The channel descriptor is not considered valid until validated by an associated SETVALID0 setting. 1b - Valid. The current channel descriptor is considered valid.

8.1.32 DMA Channel 13 Transfer Configuration Register (XFER CFG13)

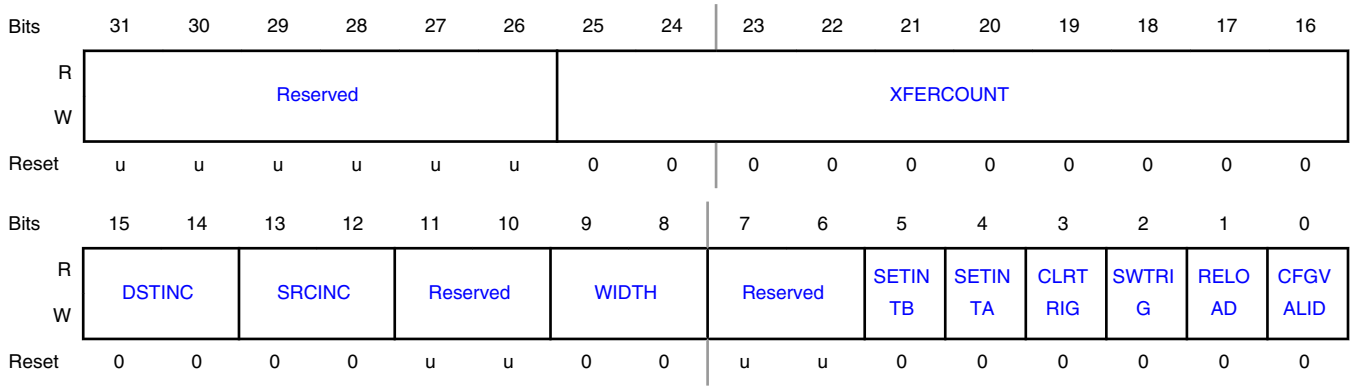
Offset

Register	Offset
XFERCFG13	4D8h

Function

The XFERCFG13 register contains transfer related configuration information for DMA channel 13. Using the Reload bit, this register can optionally be automatically reloaded when the current settings are exhausted (the full transfer count has been completed), allowing linked transfers with more than one descriptor to be performed.

Diagram



Fields

Field	Function
31-26 —	RESERVED Reserved. The value read from a reserved bit is not defined.
25-16 XFERCOUNT	Transfer Count Total number of transfers to be performed, minus 1 encoded. The number of bytes transferred is: (XFERCOUNT + 1) x data width (as defined by the WIDTH field). **NOTE** The DMA controller uses this bit field during transfer to count down. Hence, it cannot be used by software to read back the size of the transfer, for instance, in an interrupt handler. 0x0 - a total of 1 transfer will be performed. 0x1 - a total of 2 transfers will be performed. ..... 0x3FF - a total of 1,024 transfers will be performed.
15-14 DSTINC	Destination Address Incremented for DMA Transfer Determines whether the destination address is incremented for each DMA transfer. 00b - No increment. The destination address is not incremented for each transfer. This is the usual case when the destination is a peripheral device. 01b - 1 x width. The destination address is incremented by the amount specified by Width for each transfer. This is the usual case when the destination is memory. 10b - 2 x width. The destination address is incremented by 2 times the amount specified by Width for each transfer. 11b - 4 x width. The destination address is incremented by 4 times the amount specified by Width for each transfer.

Table continues on the next page...

Table continued from the previous page...

Field	Function
13-12 SRCINC	Source Address Incremented for DMA Transfer Determines whether the source address is incremented for each DMA transfer. 00b - No increment. The source address is not incremented for each transfer. This is the usual case when the source is a peripheral device. 01b - 1 x width. The source address is incremented by the amount specified by Width for each transfer. This is the usual case when the source is memory. 10b - 2 x width. The source address is incremented by 2 times the amount specified by Width for each transfer. 11b - 4 x width. The source address is incremented by 4 times the amount specified by Width for each transfer.
11-10 —	RESERVED Reserved. The value read from a reserved bit is not defined.
9-8 WIDTH	Transfer Width for DMA Channel Transfer width used for this DMA channel. 0x0 0x1 0x2 . 0x3 00b - 8-bit. 8-bit transfers are performed (8-bit source reads and destination writes). 01b - 16-bit. 16-bit transfers are performed (16-bit source reads and destination writes). 10b - 32-bit. 32-bit transfers are performed (32-bit source reads and destination writes). 11b - Reserved. Reserved setting, do not use.
7-6 —	RESERVED Reserved. The value read from a reserved bit is not defined.
5 SETINTB	Set Interrupt Flag B for Channel Set Interrupt flag B for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. 0b - No effect. 1b - Set. The INTB flag for this channel will be set when the current descriptor is exhausted.
4 SETINTA	Set Interrupt Flag A for Channel Set Interrupt flag A for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. 0b - No effect. 1b - Set. The INTA flag for this channel will be set when the current descriptor is exhausted.

Table continues on the next page...

Table continued from the previous page...

Field	Function
3 CLRTRIG	Clear Trigger 0b - Not cleared. The trigger is not cleared when this descriptor is exhausted. If there is a reload, the next descriptor will be started. 1b - Cleared. The trigger is cleared when this descriptor is exhausted.
2 SWTRIG	Software Trigger 0b - Not set. When written by software, the trigger for this channel is not set. A new trigger, as defined by the HWTRIGEN, TRIGPOL, and TRIGTYPE will be needed to start the channel. 1b - Set. When written by software, the trigger for this channel is set immediately. This feature should not be used with level triggering when TRIGBURST = 0.
1 RELOAD	Control Structure Reload Indicates whether the channel s control structure will be reloaded when the current descriptor is exhausted. Reloading allows ping-pong and linked transfers. 0b - Disabled. Do not reload the channels control structure when the current descriptor is exhausted. 1b - Enabled. Reload the channels control structure when the current descriptor is exhausted.
0 CFGVALID	Configuration Valid Flag This bit indicates whether the current channel descriptor is valid and can potentially be acted upon, if all other activation criteria are fulfilled. 0b - Not valid. The channel descriptor is not considered valid until validated by an associated SETVALID0 setting. 1b - Valid. The current channel descriptor is considered valid.

8.1.33 DMA Channel 14 Transfer Configuration Register (XFER CFG14)

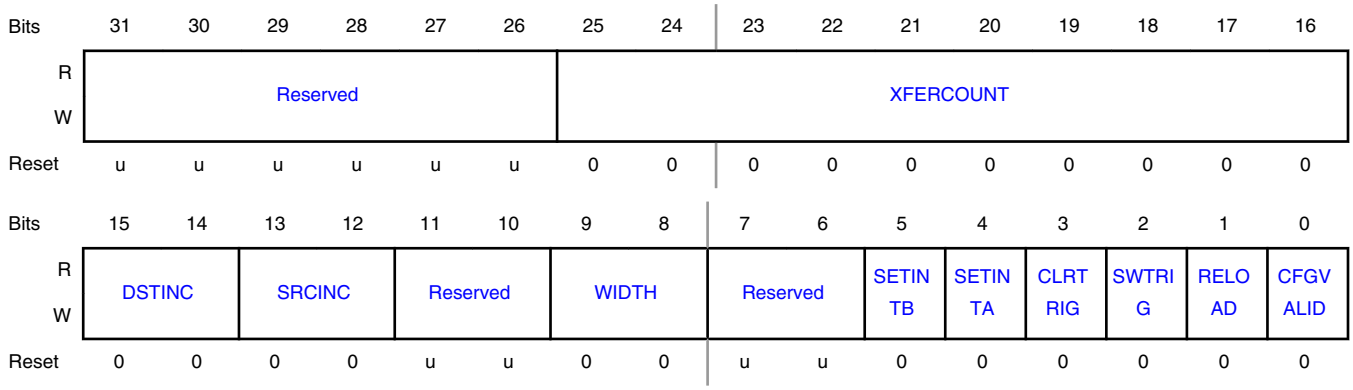
Offset

Register	Offset
XFERCFG14	4E8h

Function

The XFERCFG14 register contains transfer related configuration information for DMA channel 14. Using the Reload bit, this register can optionally be automatically reloaded when the current settings are exhausted (the full transfer count has been completed), allowing linked transfers with more than one descriptor to be performed.

Diagram



Fields

Field	Function
31-26 —	RESERVED Reserved. The value read from a reserved bit is not defined.
25-16 XFERCOUNT	Transfer Count Total number of transfers to be performed, minus 1 encoded. The number of bytes transferred is: (XFERCOUNT + 1) x data width (as defined by the WIDTH field). **NOTE** The DMA controller uses this bit field during transfer to count down. Hence, it cannot be used by software to read back the size of the transfer, for instance, in an interrupt handler. 0x0 - a total of 1 transfer will be performed. 0x1 - a total of 2 transfers will be performed. ..... 0x3FF - a total of 1,024 transfers will be performed.
15-14 DSTINC	Destination Address Incremented for DMA Transfer Determines whether the destination address is incremented for each DMA transfer. 00b - No increment. The destination address is not incremented for each transfer. This is the usual case when the destination is a peripheral device. 01b - 1 x width. The destination address is incremented by the amount specified by Width for each transfer. This is the usual case when the destination is memory. 10b - 2 x width. The destination address is incremented by 2 times the amount specified by Width for each transfer. 11b - 4 x width. The destination address is incremented by 4 times the amount specified by Width for each transfer.

Table continues on the next page...

Table continued from the previous page...

Field	Function
13-12 SRCINC	Source Address Incremented for DMA Transfer Determines whether the source address is incremented for each DMA transfer. 00b - No increment. The source address is not incremented for each transfer. This is the usual case when the source is a peripheral device. 01b - 1 x width. The source address is incremented by the amount specified by Width for each transfer. This is the usual case when the source is memory. 10b - 2 x width. The source address is incremented by 2 times the amount specified by Width for each transfer. 11b - 4 x width. The source address is incremented by 4 times the amount specified by Width for each transfer.
11-10 —	RESERVED Reserved. The value read from a reserved bit is not defined.
9-8 WIDTH	Transfer Width for DMA Channel Transfer width used for this DMA channel. 0x0 0x1 0x2 . 0x3 00b - 8-bit. 8-bit transfers are performed (8-bit source reads and destination writes). 01b - 16-bit. 16-bit transfers are performed (16-bit source reads and destination writes). 10b - 32-bit. 32-bit transfers are performed (32-bit source reads and destination writes). 11b - Reserved. Reserved setting, do not use.
7-6 —	RESERVED Reserved. The value read from a reserved bit is not defined.
5 SETINTB	Set Interrupt Flag B for Channel Set Interrupt flag B for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. 0b - No effect. 1b - Set. The INTB flag for this channel will be set when the current descriptor is exhausted.
4 SETINTA	Set Interrupt Flag A for Channel Set Interrupt flag A for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. 0b - No effect. 1b - Set. The INTA flag for this channel will be set when the current descriptor is exhausted.

Table continues on the next page...

Table continued from the previous page...

Field	Function
3 CLRTRIG	Clear Trigger 0b - Not cleared. The trigger is not cleared when this descriptor is exhausted. If there is a reload, the next descriptor will be started. 1b - Cleared. The trigger is cleared when this descriptor is exhausted.
2 SWTRIG	Software Trigger 0b - Not set. When written by software, the trigger for this channel is not set. A new trigger, as defined by the HWTRIGEN, TRIGPOL, and TRIGTYPE will be needed to start the channel. 1b - Set. When written by software, the trigger for this channel is set immediately. This feature should not be used with level triggering when TRIGBURST = 0.
1 RELOAD	Control Structure Reload Indicates whether the channel s control structure will be reloaded when the current descriptor is exhausted. Reloading allows ping-pong and linked transfers. 0b - Disabled. Do not reload the channels control structure when the current descriptor is exhausted. 1b - Enabled. Reload the channels control structure when the current descriptor is exhausted.
0 CFGVALID	Configuration Valid Flag This bit indicates whether the current channel descriptor is valid and can potentially be acted upon, if all other activation criteria are fulfilled. 0b - Not valid. The channel descriptor is not considered valid until validated by an associated SETVALID0 setting. 1b - Valid. The current channel descriptor is considered valid.

8.1.34 DMA Channel 15 Transfer Configuration Register (XFER CFG15)

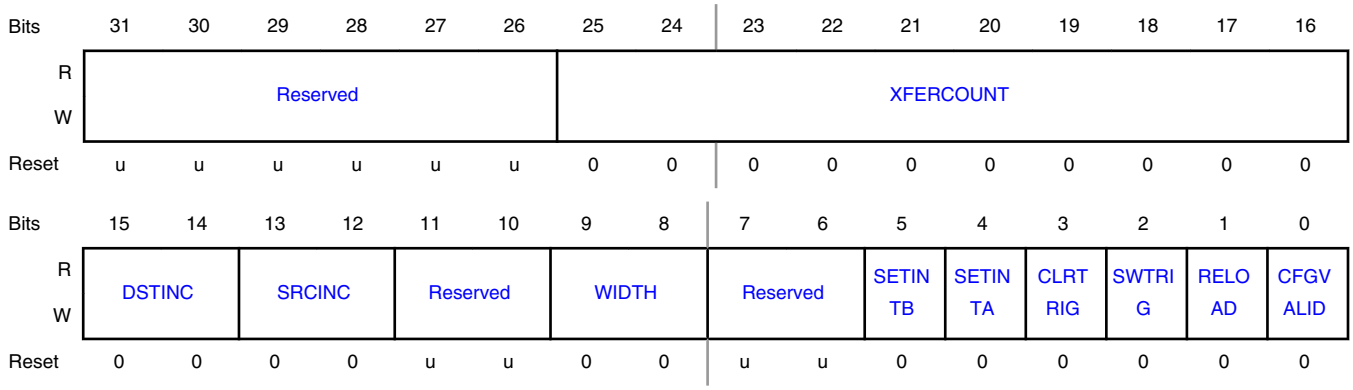
Offset

Register	Offset
XFERCFG15	4F8h

Function

The XFERCFG15 register contains transfer related configuration information for DMA channel 15. Using the Reload bit, this register can optionally be automatically reloaded when the current settings are exhausted (the full transfer count has been completed), allowing linked transfers with more than one descriptor to be performed.

Diagram



Fields

Field	Function
31-26 —	RESERVED Reserved. The value read from a reserved bit is not defined.
25-16 XFERCOUNT	Transfer Count Total number of transfers to be performed, minus 1 encoded. The number of bytes transferred is: (XFERCOUNT + 1) x data width (as defined by the WIDTH field). **NOTE** The DMA controller uses this bit field during transfer to count down. Hence, it cannot be used by software to read back the size of the transfer, for instance, in an interrupt handler. 0x0 - a total of 1 transfer will be performed. 0x1 - a total of 2 transfers will be performed. ..... 0x3FF - a total of 1,024 transfers will be performed.
15-14 DSTINC	Destination Address Incremented for DMA Transfer Determines whether the destination address is incremented for each DMA transfer. 00b - No increment. The destination address is not incremented for each transfer. This is the usual case when the destination is a peripheral device. 01b - 1 x width. The destination address is incremented by the amount specified by Width for each transfer. This is the usual case when the destination is memory. 10b - 2 x width. The destination address is incremented by 2 times the amount specified by Width for each transfer. 11b - 4 x width. The destination address is incremented by 4 times the amount specified by Width for each transfer.

Table continues on the next page...

Table continued from the previous page...

Field	Function
13-12 SRCINC	Source Address Incremented for DMA Transfer Determines whether the source address is incremented for each DMA transfer. 00b - No increment. The source address is not incremented for each transfer. This is the usual case when the source is a peripheral device. 01b - 1 x width. The source address is incremented by the amount specified by Width for each transfer. This is the usual case when the source is memory. 10b - 2 x width. The source address is incremented by 2 times the amount specified by Width for each transfer. 11b - 4 x width. The source address is incremented by 4 times the amount specified by Width for each transfer.
11-10 —	RESERVED Reserved. The value read from a reserved bit is not defined.
9-8 WIDTH	Transfer Width for DMA Channel Transfer width used for this DMA channel. 0x0 0x1 0x2 . 0x3 00b - 8-bit. 8-bit transfers are performed (8-bit source reads and destination writes). 01b - 16-bit. 16-bit transfers are performed (16-bit source reads and destination writes). 10b - 32-bit. 32-bit transfers are performed (32-bit source reads and destination writes). 11b - Reserved. Reserved setting, do not use.
7-6 —	RESERVED Reserved. The value read from a reserved bit is not defined.
5 SETINTB	Set Interrupt Flag B for Channel Set Interrupt flag B for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. 0b - No effect. 1b - Set. The INTB flag for this channel will be set when the current descriptor is exhausted.
4 SETINTA	Set Interrupt Flag A for Channel Set Interrupt flag A for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. 0b - No effect. 1b - Set. The INTA flag for this channel will be set when the current descriptor is exhausted.

Table continues on the next page...

Table continued from the previous page...

Field	Function
3 CLRTRIG	Clear Trigger 0b - Not cleared. The trigger is not cleared when this descriptor is exhausted. If there is a reload, the next descriptor will be started. 1b - Cleared. The trigger is cleared when this descriptor is exhausted.
2 SWTRIG	Software Trigger 0b - Not set. When written by software, the trigger for this channel is not set. A new trigger, as defined by the HWTRIGEN, TRIGPOL, and TRIGTYPE will be needed to start the channel. 1b - Set. When written by software, the trigger for this channel is set immediately. This feature should not be used with level triggering when TRIGBURST = 0.
1 RELOAD	Control Structure Reload Indicates whether the channel s control structure will be reloaded when the current descriptor is exhausted. Reloading allows ping-pong and linked transfers. 0b - Disabled. Do not reload the channels control structure when the current descriptor is exhausted. 1b - Enabled. Reload the channels control structure when the current descriptor is exhausted.
0 CFGVALID	Configuration Valid Flag This bit indicates whether the current channel descriptor is valid and can potentially be acted upon, if all other activation criteria are fulfilled. 0b - Not valid. The channel descriptor is not considered valid until validated by an associated SETVALID0 setting. 1b - Valid. The current channel descriptor is considered valid.

8.1.35 DMA Channel 16 Transfer Configuration Register (XFER CFG16)

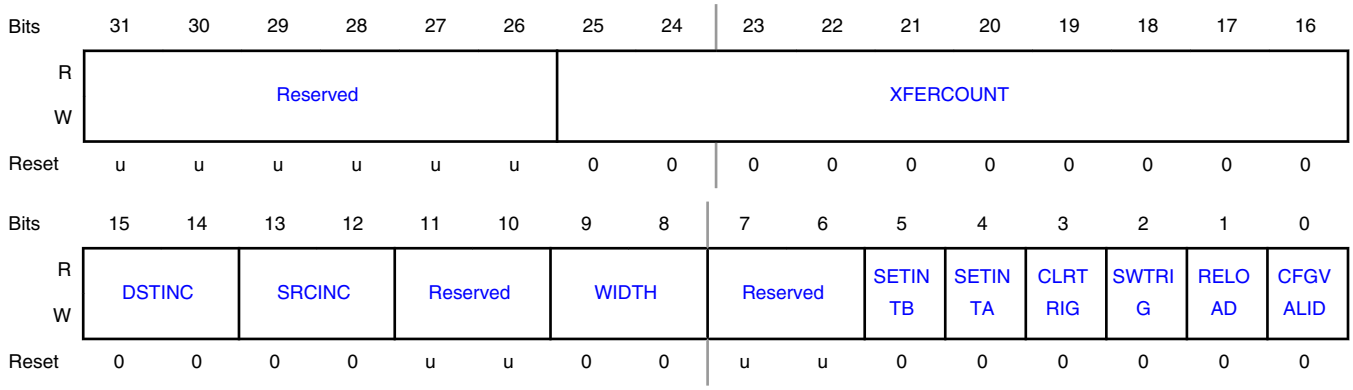
Offset

Register	Offset
XFERCFG16	508h

Function

The XFERCFG16 register contains transfer related configuration information for DMA channel 16. Using the Reload bit, this register can optionally be automatically reloaded when the current settings are exhausted (the full transfer count has been completed), allowing linked transfers with more than one descriptor to be performed.

Diagram



Fields

Field	Function
31-26 —	RESERVED Reserved. The value read from a reserved bit is not defined.
25-16 XFERCOUNT	Transfer Count Total number of transfers to be performed, minus 1 encoded. The number of bytes transferred is: (XFERCOUNT + 1) x data width (as defined by the WIDTH field). **NOTE** The DMA controller uses this bit field during transfer to count down. Hence, it cannot be used by software to read back the size of the transfer, for instance, in an interrupt handler. 0x0 - a total of 1 transfer will be performed. 0x1 - a total of 2 transfers will be performed. ..... 0x3FF - a total of 1,024 transfers will be performed.
15-14 DSTINC	Destination Address Incremented for DMA Transfer Determines whether the destination address is incremented for each DMA transfer. 00b - No increment. The destination address is not incremented for each transfer. This is the usual case when the destination is a peripheral device. 01b - 1 x width. The destination address is incremented by the amount specified by Width for each transfer. This is the usual case when the destination is memory. 10b - 2 x width. The destination address is incremented by 2 times the amount specified by Width for each transfer. 11b - 4 x width. The destination address is incremented by 4 times the amount specified by Width for each transfer.

Table continues on the next page...

Table continued from the previous page...

Field	Function
13-12 SRCINC	Source Address Incremented for DMA Transfer Determines whether the source address is incremented for each DMA transfer. 00b - No increment. The source address is not incremented for each transfer. This is the usual case when the source is a peripheral device. 01b - 1 x width. The source address is incremented by the amount specified by Width for each transfer. This is the usual case when the source is memory. 10b - 2 x width. The source address is incremented by 2 times the amount specified by Width for each transfer. 11b - 4 x width. The source address is incremented by 4 times the amount specified by Width for each transfer.
11-10 —	RESERVED Reserved. The value read from a reserved bit is not defined.
9-8 WIDTH	Transfer Width for DMA Channel Transfer width used for this DMA channel. 0x0 0x1 0x2 . 0x3 00b - 8-bit. 8-bit transfers are performed (8-bit source reads and destination writes). 01b - 16-bit. 16-bit transfers are performed (16-bit source reads and destination writes). 10b - 32-bit. 32-bit transfers are performed (32-bit source reads and destination writes). 11b - Reserved. Reserved setting, do not use.
7-6 —	RESERVED Reserved. The value read from a reserved bit is not defined.
5 SETINTB	Set Interrupt Flag B for Channel Set Interrupt flag B for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. 0b - No effect. 1b - Set. The INTB flag for this channel will be set when the current descriptor is exhausted.
4 SETINTA	Set Interrupt Flag A for Channel Set Interrupt flag A for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. 0b - No effect. 1b - Set. The INTA flag for this channel will be set when the current descriptor is exhausted.

Table continues on the next page...

Table continued from the previous page...

Field	Function
3 CLRTRIG	Clear Trigger 0b - Not cleared. The trigger is not cleared when this descriptor is exhausted. If there is a reload, the next descriptor will be started. 1b - Cleared. The trigger is cleared when this descriptor is exhausted.
2 SWTRIG	Software Trigger 0b - Not set. When written by software, the trigger for this channel is not set. A new trigger, as defined by the HWTRIGEN, TRIGPOL, and TRIGTYPE will be needed to start the channel. 1b - Set. When written by software, the trigger for this channel is set immediately. This feature should not be used with level triggering when TRIGBURST = 0.
1 RELOAD	Control Structure Reload Indicates whether the channel s control structure will be reloaded when the current descriptor is exhausted. Reloading allows ping-pong and linked transfers. 0b - Disabled. Do not reload the channels control structure when the current descriptor is exhausted. 1b - Enabled. Reload the channels control structure when the current descriptor is exhausted.
0 CFGVALID	Configuration Valid Flag This bit indicates whether the current channel descriptor is valid and can potentially be acted upon, if all other activation criteria are fulfilled. 0b - Not valid. The channel descriptor is not considered valid until validated by an associated SETVALID0 setting. 1b - Valid. The current channel descriptor is considered valid.

8.1.36 DMA Channel 17 Transfer Configuration Register (XFER CFG17)

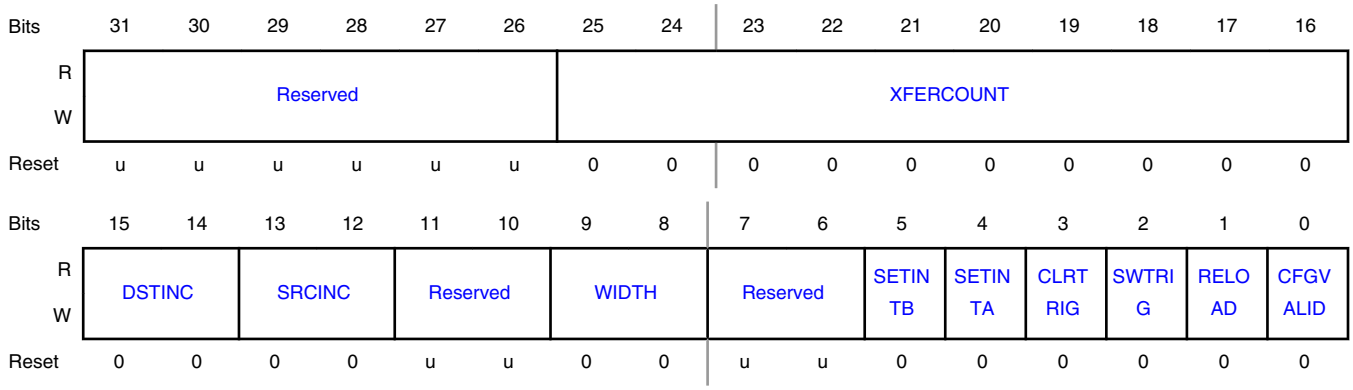
Offset

Register	Offset
XFERCFG17	518h

Function

The XFERCFG17 register contains transfer related configuration information for DMA channel 17. Using the Reload bit, this register can optionally be automatically reloaded when the current settings are exhausted (the full transfer count has been completed), allowing linked transfers with more than one descriptor to be performed.

Diagram



Fields

Field	Function
31-26 —	RESERVED Reserved. The value read from a reserved bit is not defined.
25-16 XFERCOUNT	Transfer Count Total number of transfers to be performed, minus 1 encoded. The number of bytes transferred is: (XFERCOUNT + 1) x data width (as defined by the WIDTH field). **NOTE** The DMA controller uses this bit field during transfer to count down. Hence, it cannot be used by software to read back the size of the transfer, for instance, in an interrupt handler. 0x0 - a total of 1 transfer will be performed. 0x1 - a total of 2 transfers will be performed. ..... 0x3FF - a total of 1,024 transfers will be performed.
15-14 DSTINC	Destination Address Incremented for DMA Transfer Determines whether the destination address is incremented for each DMA transfer. 00b - No increment. The destination address is not incremented for each transfer. This is the usual case when the destination is a peripheral device. 01b - 1 x width. The destination address is incremented by the amount specified by Width for each transfer. This is the usual case when the destination is memory. 10b - 2 x width. The destination address is incremented by 2 times the amount specified by Width for each transfer. 11b - 4 x width. The destination address is incremented by 4 times the amount specified by Width for each transfer.

Table continues on the next page...

Table continued from the previous page...

Field	Function
13-12 SRCINC	Source Address Incremented for DMA Transfer Determines whether the source address is incremented for each DMA transfer. 00b - No increment. The source address is not incremented for each transfer. This is the usual case when the source is a peripheral device. 01b - 1 x width. The source address is incremented by the amount specified by Width for each transfer. This is the usual case when the source is memory. 10b - 2 x width. The source address is incremented by 2 times the amount specified by Width for each transfer. 11b - 4 x width. The source address is incremented by 4 times the amount specified by Width for each transfer.
11-10 —	RESERVED Reserved. The value read from a reserved bit is not defined.
9-8 WIDTH	Transfer Width for DMA Channel Transfer width used for this DMA channel. 0x0 0x1 0x2 . 0x3 00b - 8-bit. 8-bit transfers are performed (8-bit source reads and destination writes). 01b - 16-bit. 16-bit transfers are performed (16-bit source reads and destination writes). 10b - 32-bit. 32-bit transfers are performed (32-bit source reads and destination writes). 11b - Reserved. Reserved setting, do not use.
7-6 —	RESERVED Reserved. The value read from a reserved bit is not defined.
5 SETINTB	Set Interrupt Flag B for Channel Set Interrupt flag B for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. 0b - No effect. 1b - Set. The INTB flag for this channel will be set when the current descriptor is exhausted.
4 SETINTA	Set Interrupt Flag A for Channel Set Interrupt flag A for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. 0b - No effect. 1b - Set. The INTA flag for this channel will be set when the current descriptor is exhausted.

Table continues on the next page...

Table continued from the previous page...

Field	Function
3 CLRTRIG	Clear Trigger 0b - Not cleared. The trigger is not cleared when this descriptor is exhausted. If there is a reload, the next descriptor will be started. 1b - Cleared. The trigger is cleared when this descriptor is exhausted.
2 SWTRIG	Software Trigger 0b - Not set. When written by software, the trigger for this channel is not set. A new trigger, as defined by the HWTRIGEN, TRIGPOL, and TRIGTYPE will be needed to start the channel. 1b - Set. When written by software, the trigger for this channel is set immediately. This feature should not be used with level triggering when TRIGBURST = 0.
1 RELOAD	Control Structure Reload Indicates whether the channel s control structure will be reloaded when the current descriptor is exhausted. Reloading allows ping-pong and linked transfers. 0b - Disabled. Do not reload the channels control structure when the current descriptor is exhausted. 1b - Enabled. Reload the channels control structure when the current descriptor is exhausted.
0 CFGVALID	Configuration Valid Flag This bit indicates whether the current channel descriptor is valid and can potentially be acted upon, if all other activation criteria are fulfilled. 0b - Not valid. The channel descriptor is not considered valid until validated by an associated SETVALID0 setting. 1b - Valid. The current channel descriptor is considered valid.

8.1.37 DMA Channel 18 Transfer Configuration Register (XFER CFG18)

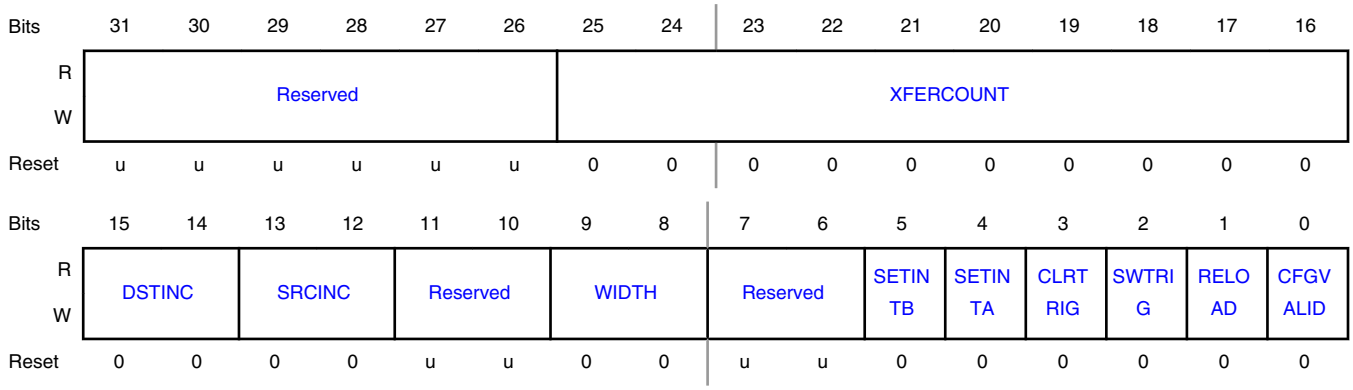
Offset

Register	Offset
XFERCFG18	528h

Function

The XFERCFG18 register contains transfer related configuration information for DMA channel 18. Using the Reload bit, this register can optionally be automatically reloaded when the current settings are exhausted (the full transfer count has been completed), allowing linked transfers with more than one descriptor to be performed.

Diagram



Fields

Field	Function
31-26 —	RESERVED Reserved. The value read from a reserved bit is not defined.
25-16 XFERCOUNT	Transfer Count Total number of transfers to be performed, minus 1 encoded. The number of bytes transferred is: (XFERCOUNT + 1) x data width (as defined by the WIDTH field). **NOTE** The DMA controller uses this bit field during transfer to count down. Hence, it cannot be used by software to read back the size of the transfer, for instance, in an interrupt handler. 0x0 - a total of 1 transfer will be performed. 0x1 - a total of 2 transfers will be performed. ..... 0x3FF - a total of 1,024 transfers will be performed.
15-14 DSTINC	Destination Address Incremented for DMA Transfer Determines whether the destination address is incremented for each DMA transfer. 00b - No increment. The destination address is not incremented for each transfer. This is the usual case when the destination is a peripheral device. 01b - 1 x width. The destination address is incremented by the amount specified by Width for each transfer. This is the usual case when the destination is memory. 10b - 2 x width. The destination address is incremented by 2 times the amount specified by Width for each transfer. 11b - 4 x width. The destination address is incremented by 4 times the amount specified by Width for each transfer.

Table continues on the next page...

Table continued from the previous page...

Field	Function
13-12 SRCINC	Source Address Incremented for DMA Transfer Determines whether the source address is incremented for each DMA transfer. 00b - No increment. The source address is not incremented for each transfer. This is the usual case when the source is a peripheral device. 01b - 1 x width. The source address is incremented by the amount specified by Width for each transfer. This is the usual case when the source is memory. 10b - 2 x width. The source address is incremented by 2 times the amount specified by Width for each transfer. 11b - 4 x width. The source address is incremented by 4 times the amount specified by Width for each transfer.
11-10 —	RESERVED Reserved. The value read from a reserved bit is not defined.
9-8 WIDTH	Transfer Width for DMA Channel Transfer width used for this DMA channel. 0x0 0x1 0x2 . 0x3 00b - 8-bit. 8-bit transfers are performed (8-bit source reads and destination writes). 01b - 16-bit. 16-bit transfers are performed (16-bit source reads and destination writes). 10b - 32-bit. 32-bit transfers are performed (32-bit source reads and destination writes). 11b - Reserved. Reserved setting, do not use.
7-6 —	RESERVED Reserved. The value read from a reserved bit is not defined.
5 SETINTB	Set Interrupt Flag B for Channel Set Interrupt flag B for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. 0b - No effect. 1b - Set. The INTB flag for this channel will be set when the current descriptor is exhausted.
4 SETINTA	Set Interrupt Flag A for Channel Set Interrupt flag A for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. 0b - No effect. 1b - Set. The INTA flag for this channel will be set when the current descriptor is exhausted.

Table continues on the next page...

Table continued from the previous page...

Field	Function
3 CLRTRIG	Clear Trigger 0b - Not cleared. The trigger is not cleared when this descriptor is exhausted. If there is a reload, the next descriptor will be started. 1b - Cleared. The trigger is cleared when this descriptor is exhausted.
2 SWTRIG	Software Trigger 0b - Not set. When written by software, the trigger for this channel is not set. A new trigger, as defined by the HWTRIGEN, TRIGPOL, and TRIGTYPE will be needed to start the channel. 1b - Set. When written by software, the trigger for this channel is set immediately. This feature should not be used with level triggering when TRIGBURST = 0.
1 RELOAD	Control Structure Reload Indicates whether the channel s control structure will be reloaded when the current descriptor is exhausted. Reloading allows ping-pong and linked transfers. 0b - Disabled. Do not reload the channels control structure when the current descriptor is exhausted. 1b - Enabled. Reload the channels control structure when the current descriptor is exhausted.
0 CFGVALID	Configuration Valid Flag This bit indicates whether the current channel descriptor is valid and can potentially be acted upon, if all other activation criteria are fulfilled. 0b - Not valid. The channel descriptor is not considered valid until validated by an associated SETVALID0 setting. 1b - Valid. The current channel descriptor is considered valid.

Chapter 9
Pulse Width Modulation (PWM)

9.1 PWM register descriptions

9.1.1 PWM memory map

PWM base address: 4000_C000h

Offset	Register	Width (In bits)	Access	Reset value
0h	PWM Control Register 0 (CTRL0)	32	RW	See description
4h	PWM Control Register 1 (CTRL1)	32	RW	See description
8h	PWM Channels 0 and 1 Prescalers Register (PSCL01)	32	RW	See description
Ch	PWM Channels 2 and 3 Prescalers Register (PSCL23)	32	RW	See description
10h	PWM Channels 4 and 5 Prescalers Register (PSCL45)	32	RW	See description
14h	PWM Channels 6 and 7 Prescalers Register (PSCL67)	32	RW	See description
18h	PWM Channels 8 and 9 Prescalers Register (PSCL89)	32	RW	See description
1Ch	PWM Channel 10 Prescalers Register (PSCL10)	32	RW	See description
20h	PWM Channel 0 Period and Compare register (PCP0)	32	RW	FFFF_FFFFh
24h	PWM Channel 1 Period and Compare register (PCP1)	32	RW	FFFF_FFFFh
28h	PWM Channel 2 Period and Compare register (PCP2)	32	RW	FFFF_FFFFh
2Ch	PWM Channel 3 Period and Compare register (PCP3)	32	RW	FFFF_FFFFh
30h	PWM Channel 4 Period and Compare register (PCP4)	32	RW	FFFF_FFFFh
34h	PWM Channel 5 Period and Compare register (PCP5)	32	RW	FFFF_FFFFh
38h	PWM Channel 6 Period and Compare register (PCP6)	32	RW	FFFF_FFFFh
3Ch	PWM Channel 7 Period and Compare register (PCP7)	32	RW	FFFF_FFFFh
40h	PWM Channel 8 Period and Compare register (PCP8)	32	RW	FFFF_FFFFh
44h	PWM Channel 9 Period and Compare register (PCP9)	32	RW	FFFF_FFFFh
48h	PWM Channel 10 Period and Compare register (PCP10)	32	RW	FFFF_FFFFh

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
4Ch	PWM Status Register 0 (Channel 0 to Channel 3) (PST0)	32	RW	See description
50h	PWM Status Register 1 (Channel 4 to Channel 7) (PST1)	32	RW	See description
54h	PWM Status Register 2 (Channel 8 to Channel 10) (PST2)	32	RW	See description
FFCh	PWM Module Identifier Register ('PW' in ASCII) (MODULE_ID)	32	RO	5057_0000h

9.1.2 PWM Control Register 0 (CTRL0)

Offset

Register	Offset
CTRL0	0h

Function

PWM Control Register 0 is used to enable channels and channel interrupts (channel 0 to channel 10). Note if all interrupts are enabled with short period timings it is not possible to manage all the interrupts.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved					INT_EN_...	INT_EN_9	INT_EN_8	INT_EN_7	INT_EN_6	INT_EN_5	INT_EN_4	INT_EN_3	INT_EN_2	INT_EN_1	INT_EN_0
W																
Reset	u	u	u	u	u	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved					PWM_EN_...	PWM_EN_9	PWM_EN_8	PWM_EN_7	PWM_EN_6	PWM_EN_5	PWM_EN_4	PWM_EN_3	PWM_EN_2	PWM_EN_1	PWM_EN_0
W																
Reset	u	u	u	u	u	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-27 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
26-16 INT_EN_n	PWM Channel n Interrupt Enable 0b - The channel interrupt is disabled. 1b - The channel interrupt is enabled.
15-11 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
10-0 PWM_EN_n	PWM Channel n Enable Note, PWM_EN_10 enables the common PWM mode where PWM10 will be routed to all PWM channels. 0b - The channel is disabled. 1b - The channel is enabled.

9.1.3 PWM Control Register 1 (CTRL1)

Offset

Register	Offset
CTRL1	4h

Function

PWM Control Register 1 is used to control waveform polarity and output level for Channel 0 to Channel 10.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved						DIS_LEV...	DIS_LEV...	DIS_LEV...	DIS_LEV...	DIS_LEV...	DIS_LEV...	DIS_LEV...	DIS_LEV...	DIS_LEV...	DIS_LEV...
Reset	u	u	u	u	u	u	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved					POL_10	POL_9	POL_8	POL_7	POL_6	POL_5	POL_4	POL_3	POL_2	POL_1	POL_0
Reset	u	u	u	u	u	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-26 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
25-16 DIS_LEVEL_n	PWM Channel n Output Level when PWM channel n is disabled 0b - Low Level. 1b - High Level.
15-11 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
10-0 POL_n	PWM Channel n Waveform Polarity Control 0b - Set high on compare match, set low at the end of PWM period. 1b - Set low on compare match, set high at the end of PWM period

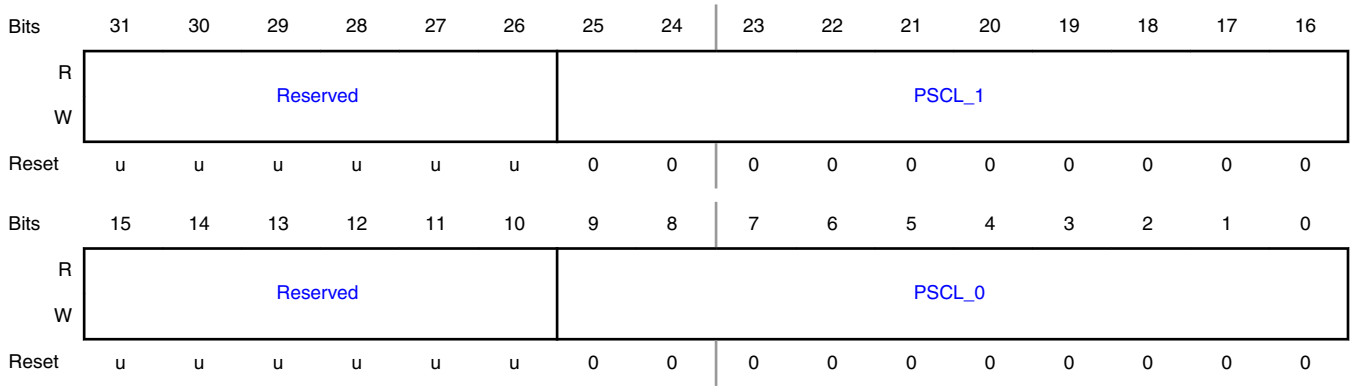
9.1.4 PWM Channels 0 and 1 Prescalers Register (PSCL01)

Offset

Register	Offset
PSCL01	8h

Function
Each PWM has its own prescaler. This register sets the prescale value for PWM channels 0 and 1.

Diagram



Fields

Field	Function
31-26 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
25-16 PSCL_1	PWM Channel 1 Prescaler The output frequency equals to clk/(PSCL_1 + 1)
15-10 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
9-0 PSCL_0	PWM Channel 0 Prescaler The output frequency equals to clk/(PSCL_0 + 1)

9.1.5 PWM Channels 2 and 3 Prescalers Register (PSCL23)

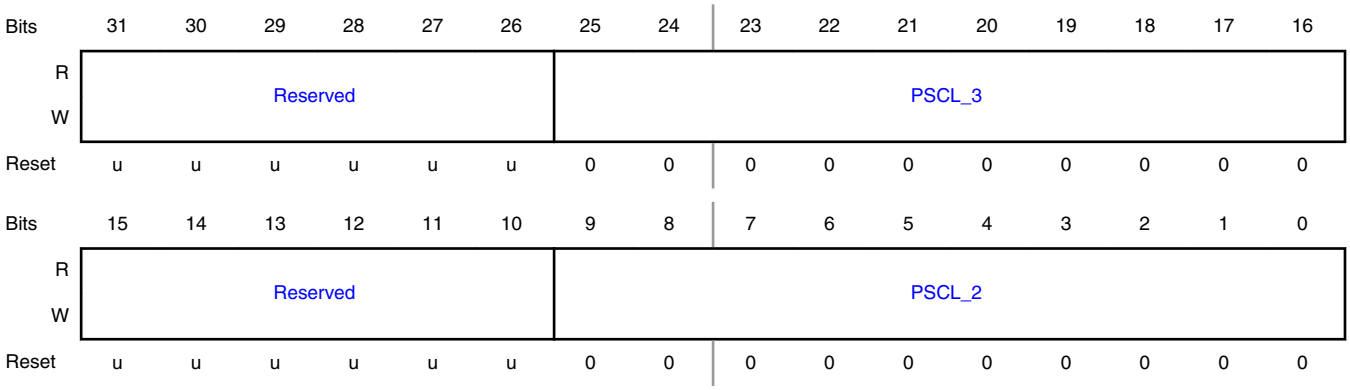
Offset

Register	Offset
PSCL23	Ch

Function

Each PWM has its own prescaler. This register sets the prescale value for PWM channels 2 and 3.

Diagram



Fields

Field	Function
31-26 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
25-16 PSCL_3	PWM Channel 3 Prescaler The output frequency equals to clk/(PSCL_3 + 1)
15-10 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
9-0 PSCL_2	PWM Channel 2 Prescaler The output frequency equals to clk/(PSCL_2 + 1)

9.1.6 PWM Channels 4 and 5 Prescalers Register (PSCL45)

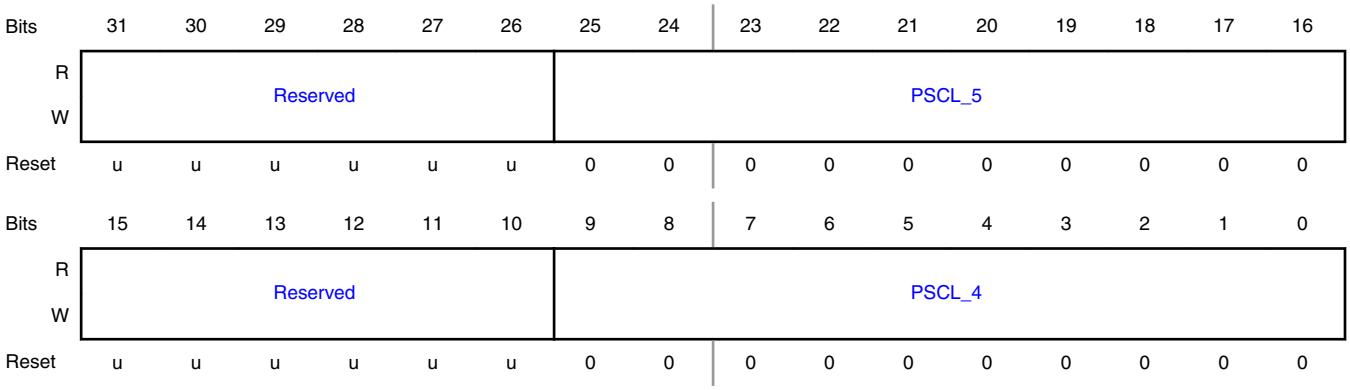
Offset

Register	Offset
PSCL45	10h

Function

Each PWM has its own prescaler. This register sets the prescale value for PWM channels 4 and 5.

Diagram



Fields

Field	Function
31-26 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
25-16 PSCL_5	PWM Channel 5 Prescaler The output frequency equals to clk/(PSCL_5 + 1)
15-10 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
9-0 PSCL_4	PWM Channel 4 Prescaler The output frequency equals to clk/(PSCL_4 + 1)

9.1.7 PWM Channels 6 and 7 Prescalers Register (PSCL67)

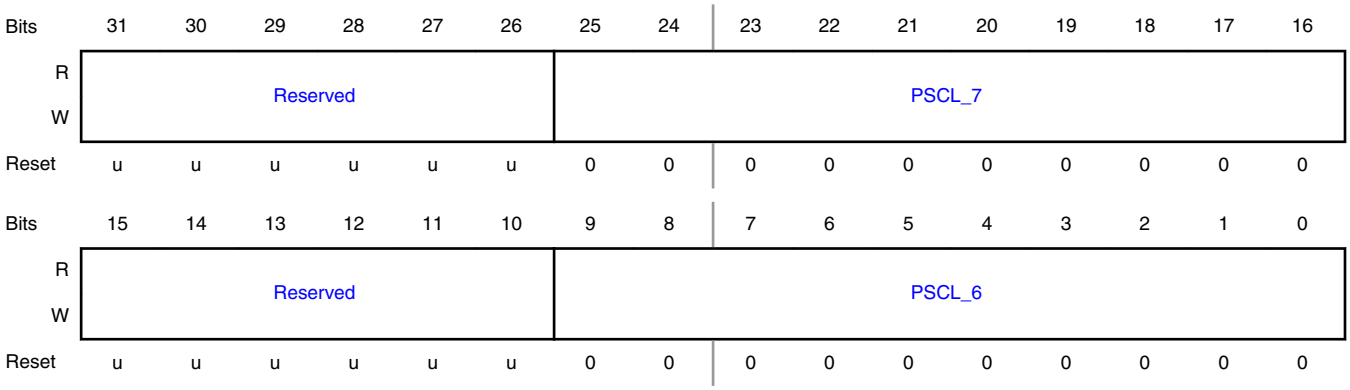
Offset

Register	Offset
PSCL67	14h

Function

Each PWM has its own prescaler. This register sets the prescale value for PWM channels 6 and 7.

Diagram



Fields

Field	Function
31-26 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
25-16 PSCL_7	PWM Channel 7 Prescaler The output frequency equals to clk/(PSCL_7 + 1)
15-10 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
9-0 PSCL_6	PWM Channel 6 Prescaler The output frequency equals to clk/(PSCL_6 + 1)

9.1.8 PWM Channels 8 and 9 Prescalers Register (PSCL89)

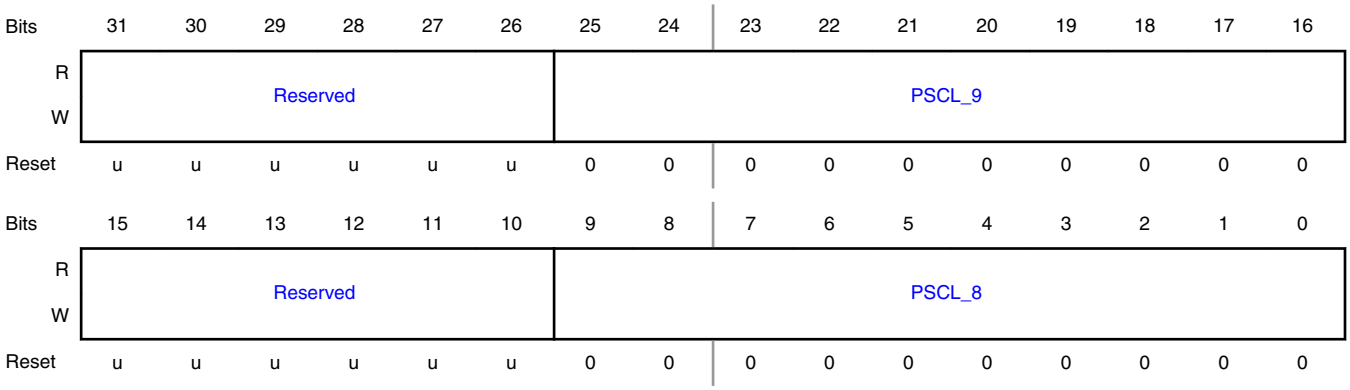
Offset

Register	Offset
PSCL89	18h

Function

Each PWM has its own prescaler. This register sets the prescale value for PWM channels 8 and 9.

Diagram



Fields

Field	Function
31-26 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
25-16 PSCL_9	PWM Channel 9 Prescaler The output frequency equals to clk/(PSCL_9 + 1)
15-10 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
9-0 PSCL_8	PWM Channel 8 Prescaler The output frequency equals to clk/(PSCL_8 + 1)

9.1.9 PWM Channel 10 Prescalers Register (PSCL10)

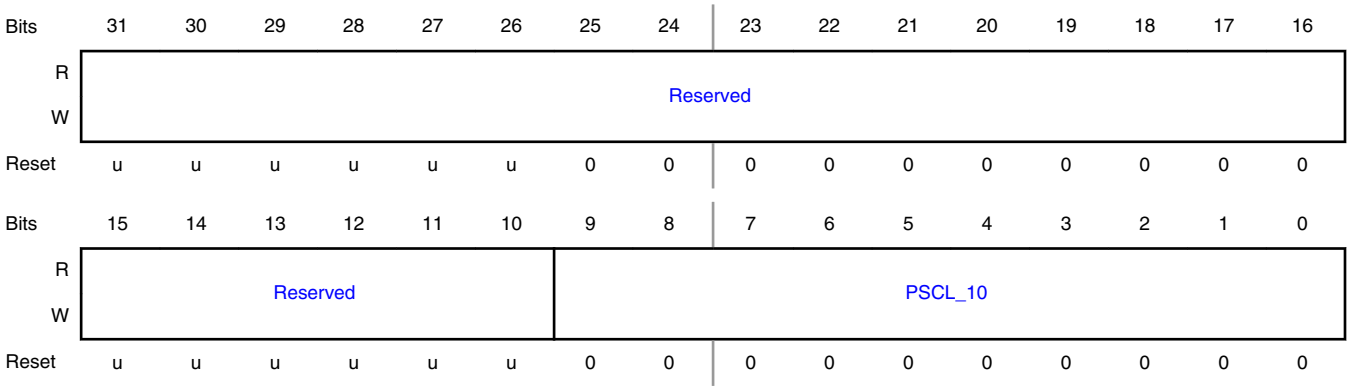
Offset

Register	Offset
PSCL10	1Ch

Function

Each PWM has its own prescaler. This register sets the prescale value for PWM channel 10.

Diagram



Fields

Field	Function
31-10 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
9-0 PSCL_10	PWM Channel 10 Prescaler The output frequency equals to clk/(PSCL_10 + 1)

9.1.10 PWM Channel 0 Period and Compare register (PCP0)

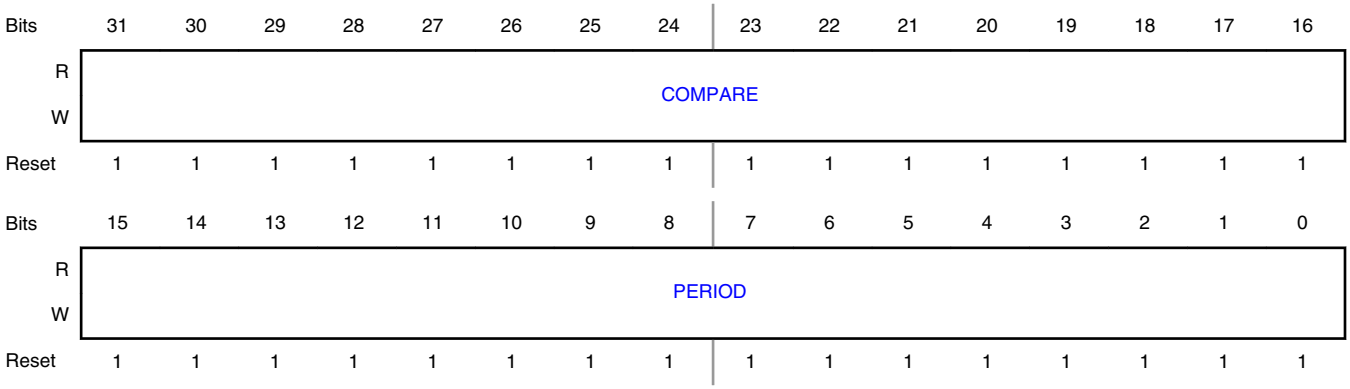
Offset

Register	Offset
PCP0	20h

Function

PERIOD and COMPARE setting for PWM channel 0. Each channel has a counter that counts down from PERIOD to 0. When COMPARE value is reached, PWM output will change on next counter decrement and be stable from 'COMPARE-1' to 0.

Diagram



Fields

Field	Function
31-16 COMPARE	PWM channel 0 Compare 'COMPARE' must not be 0x0.
15-0 PERIOD	PWM Channel 0 period The actual period equals to [PERIOD + 1]. 'PERIOD' must not be 0x0.

9.1.11 PWM Channel 1 Period and Compare register (PCP1)

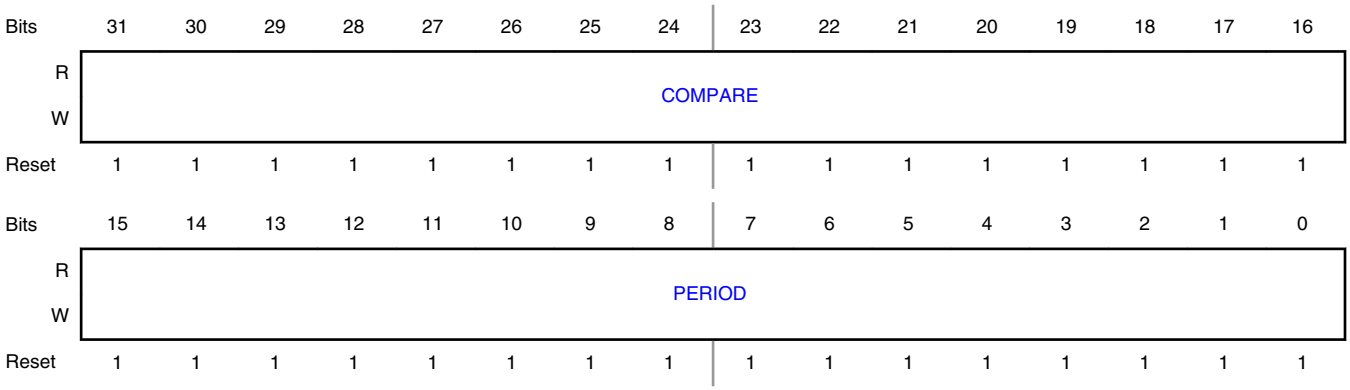
Offset

Register	Offset
PCP1	24h

Function

PERIOD and COMPARE setting for PWM channel 1. Each channel has a counter that counts down from PERIOD to 0. When COMPARE value is reached, PWM output will change on next counter decrement and be stable from 'COMPARE-1' to 0.

Diagram



Fields

Field	Function
31-16 COMPARE	PWM channel 1 Compare 'COMPARE' must not be 0x0.
15-0 PERIOD	PWM Channel 1 period The actual period equals to [PERIOD + 1]. 'PERIOD' must not be 0x0.

9.1.12 PWM Channel 2 Period and Compare register (PCP2)

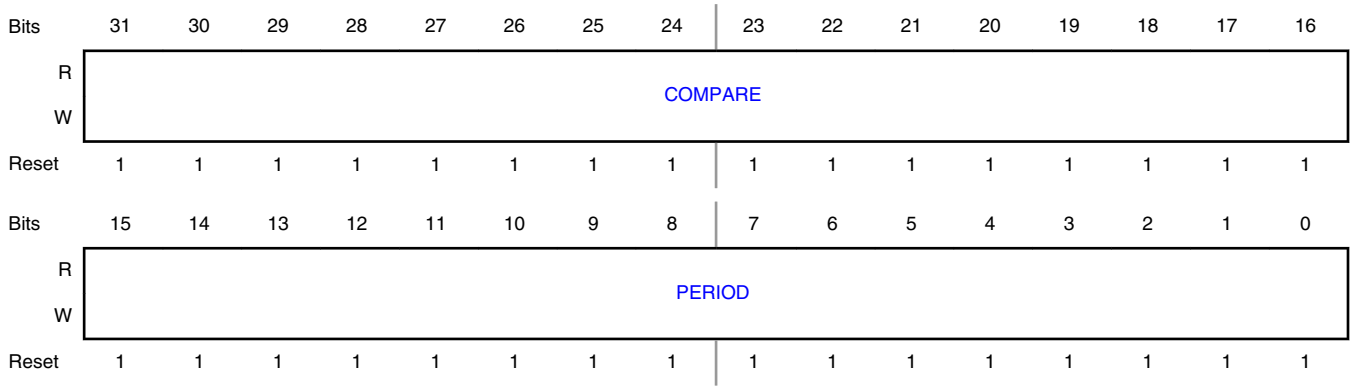
Offset

Register	Offset
PCP2	28h

Function

PERIOD and COMPARE setting for PWM channel 2. Each channel has a counter that counts down from PERIOD to 0. When COMPARE value is reached, PWM output will change on next counter decrement and be stable from 'COMPARE-1' to 0.

Diagram



Fields

Field	Function
31-16 COMPARE	PWM channel 2 Compare 'COMPARE' must not be 0x0.
15-0 PERIOD	PWM Channel 2 period The actual period equals to [PERIOD + 1]. 'PERIOD' must not be 0x0.

9.1.13 PWM Channel 3 Period and Compare register (PCP3)

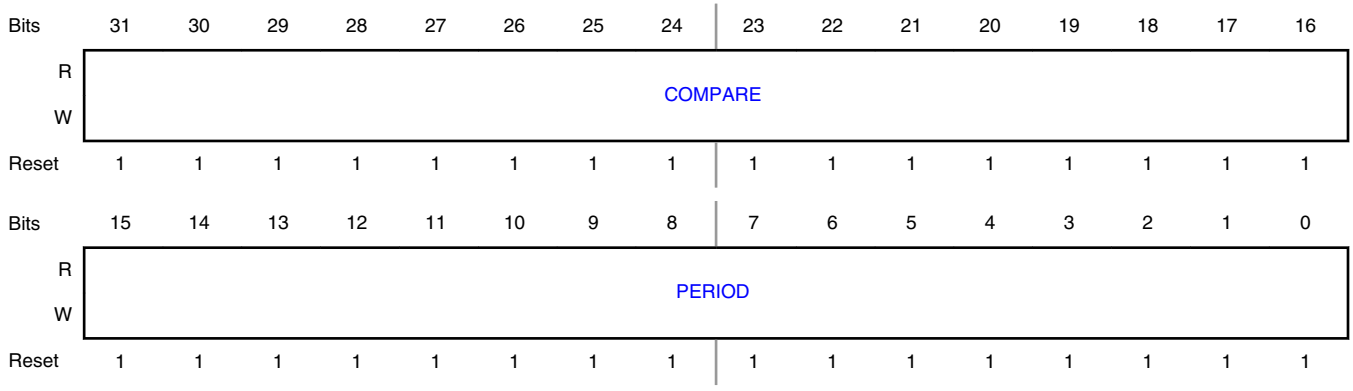
Offset

Register	Offset
PCP3	2Ch

Function

PERIOD and COMPARE setting for PWM channel 3. Each channel has a counter that counts down from PERIOD to 0. When COMPARE value is reached, PWM output will change on next counter decrement and be stable from 'COMPARE-1' to 0.

Diagram



Fields

Field	Function
31-16 COMPARE	PWM channel 3 Compare 'COMPARE' must not be 0x0.
15-0 PERIOD	PWM Channel 3 period The actual period equals to [PERIOD + 1]. 'PERIOD' must not be 0x0.

9.1.14 PWM Channel 4 Period and Compare register (PCP4)

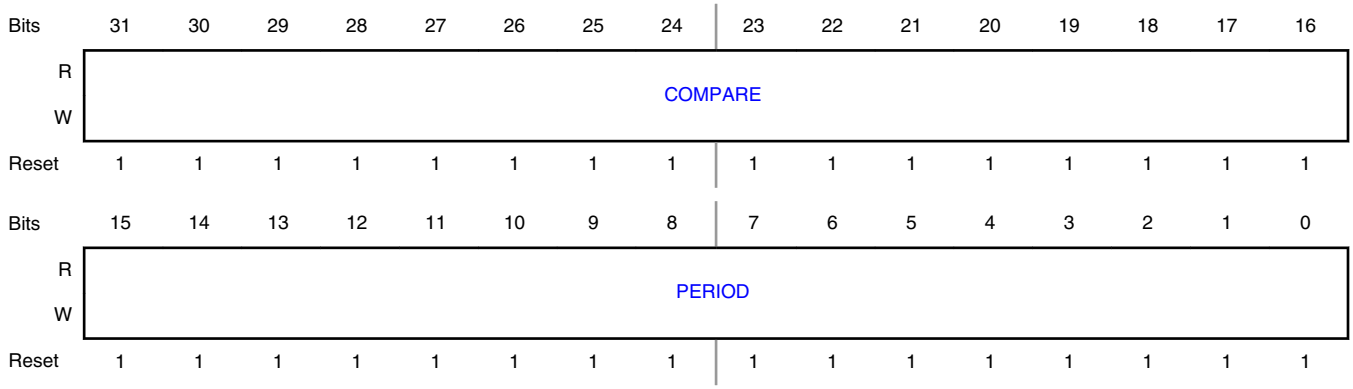
Offset

Register	Offset
PCP4	30h

Function

PERIOD and COMPARE setting for PWM channel 4. Each channel has a counter that counts down from PERIOD to 0. When COMPARE value is reached, PWM output will change on next counter decrement and be stable from 'COMPARE-1' to 0.

Diagram



Fields

Field	Function
31-16 COMPARE	PWM channel 4 Compare 'COMPARE' must not be 0x0.
15-0 PERIOD	PWM Channel 4 period The actual period equals to [PERIOD + 1]. 'PERIOD' must not be 0x0.

9.1.15 PWM Channel 5 Period and Compare register (PCP5)

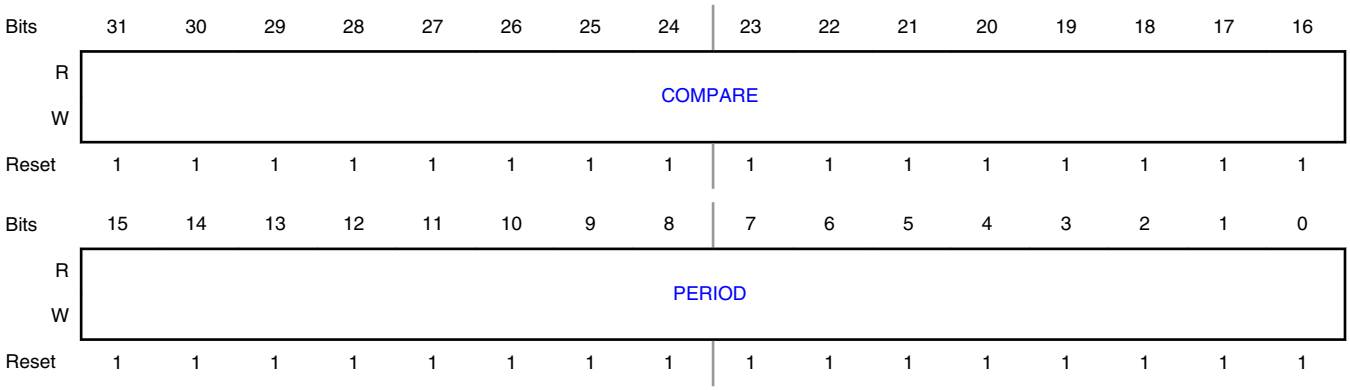
Offset

Register	Offset
PCP5	34h

Function

PERIOD and COMPARE setting for PWM channel 5. Each channel has a counter that counts down from PERIOD to 0. When COMPARE value is reached, PWM output will change on next counter decrement and be stable from 'COMPARE-1' to 0.

Diagram



Fields

Field	Function
31-16 COMPARE	PWM channel 5 Compare 'COMPARE' must not be 0x0.
15-0 PERIOD	PWM Channel 5 period The actual period equals to [PERIOD + 1]. 'PERIOD' must not be 0x0.

9.1.16 PWM Channel 6 Period and Compare register (PCP6)

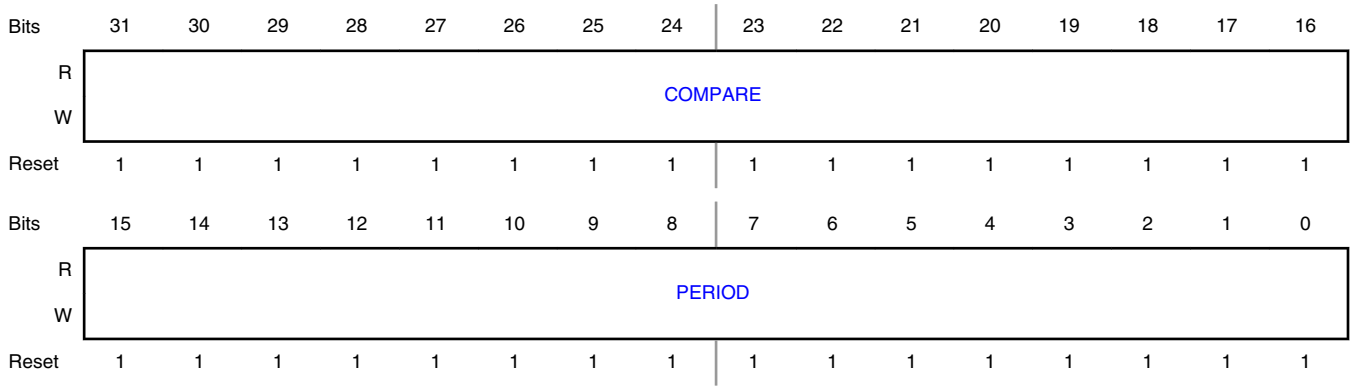
Offset

Register	Offset
PCP6	38h

Function

PERIOD and COMPARE setting for PWM channel 6. Each channel has a counter that counts down from PERIOD to 0. When COMPARE value is reached, PWM output will change on next counter decrement and be stable from 'COMPARE-1' to 0.

Diagram



Fields

Field	Function
31-16 COMPARE	PWM channel 6 Compare 'COMPARE' must not be 0x0.
15-0 PERIOD	PWM Channel 6 period The actual period equals to [PERIOD + 1]. 'PERIOD' must not be 0x0.

9.1.17 PWM Channel 7 Period and Compare register (PCP7)

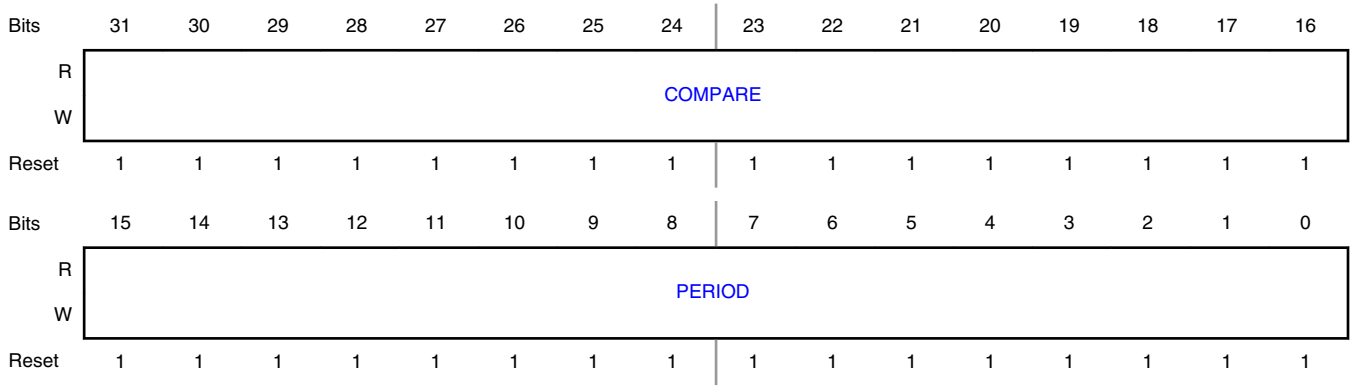
Offset

Register	Offset
PCP7	3Ch

Function

PERIOD and COMPARE setting for PWM channel 7. Each channel has a counter that counts down from PERIOD to 0. When COMPARE value is reached, PWM output will change on next counter decrement and be stable from 'COMPARE-1' to 0.

Diagram



Fields

Field	Function
31-16 COMPARE	PWM channel 7 Compare 'COMPARE' must not be 0x0.
15-0 PERIOD	PWM Channel 7 period The actual period equals to [PERIOD + 1]. 'PERIOD' must not be 0x0.

9.1.18 PWM Channel 8 Period and Compare register (PCP8)

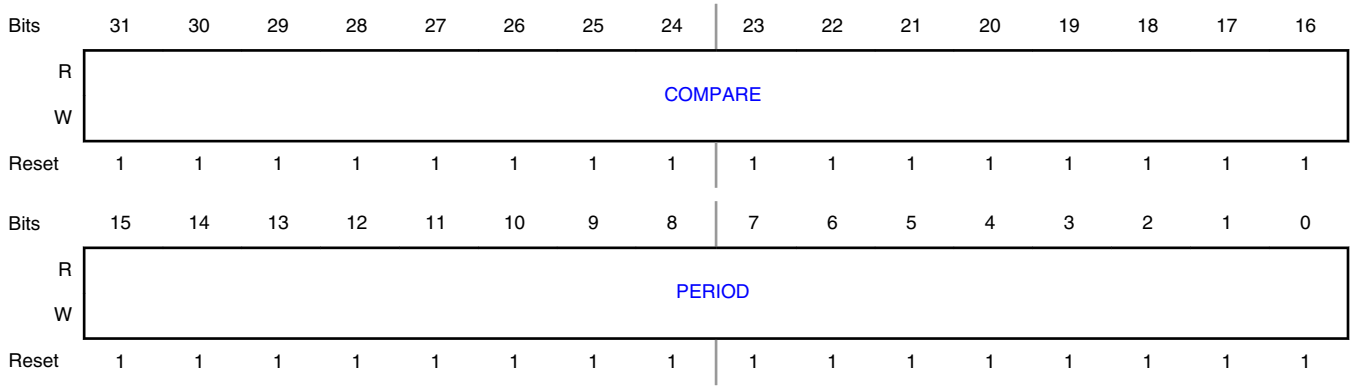
Offset

Register	Offset
PCP8	40h

Function

PERIOD and COMPARE setting for PWM channel 8. Each channel has a counter that counts down from PERIOD to 0. When COMPARE value is reached, PWM output will change on next counter decrement and be stable from 'COMPARE-1' to 0.

Diagram



Fields

Field	Function
31-16 COMPARE	PWM channel 8 Compare 'COMPARE' must not be 0x0.
15-0 PERIOD	PWM Channel 8 period The actual period equals to [PERIOD + 1]. 'PERIOD' must not be 0x0.

9.1.19 PWM Channel 9 Period and Compare register (PCP9)

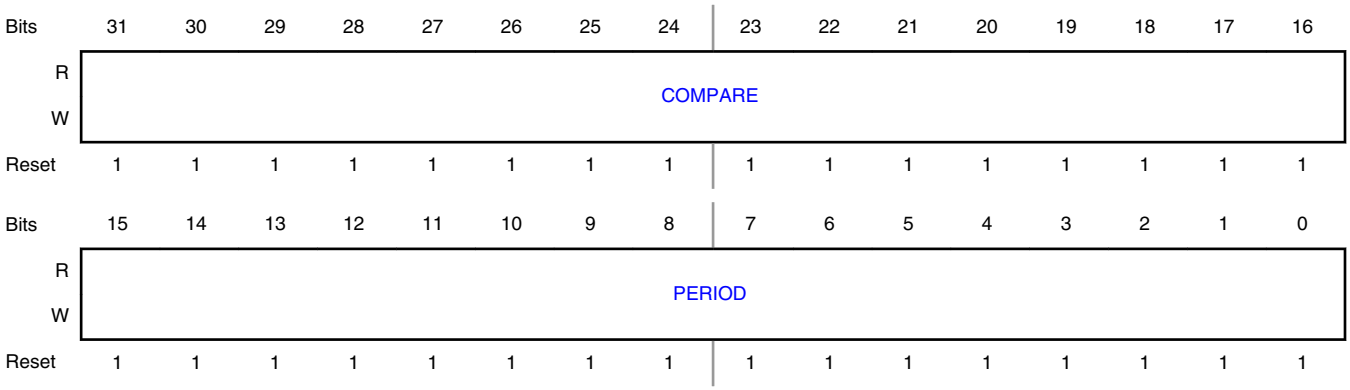
Offset

Register	Offset
PCP9	44h

Function

PERIOD and COMPARE setting for PWM channel 9. Each channel has a counter that counts down from PERIOD to 0. When COMPARE value is reached, PWM output will change on next counter decrement and be stable from 'COMPARE-1' to 0.

Diagram



Fields

Field	Function
31-16 COMPARE	PWM channel 9 Compare 'COMPARE' must not be 0x0.
15-0 PERIOD	PWM Channel 9 period The actual period equals to [PERIOD + 1]. 'PERIOD' must not be 0x0.

9.1.20 PWM Channel 10 Period and Compare register (PCP10)

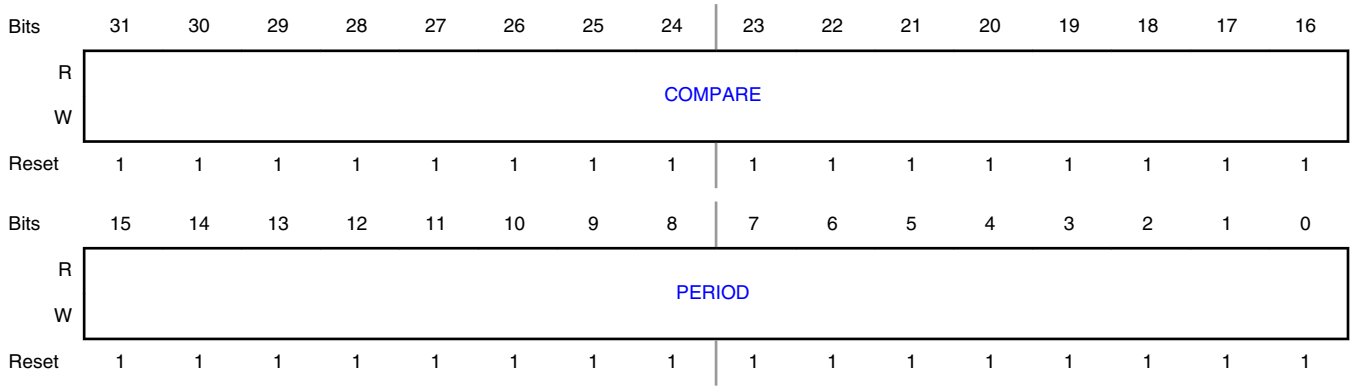
Offset

Register	Offset
PCP10	48h

Function

PERIOD and COMPARE setting for PWM channel 10. Each channel has a counter that counts down from PERIOD to 0. When COMPARE value is reached, PWM output will change on next counter decrement and be stable from 'COMPARE-1' to 0.

Diagram



Fields

Field	Function
31-16 COMPARE	PWM channel 10 Compare 'COMPARE' must not be 0x0.
15-0 PERIOD	PWM Channel 10 period The actual period equals to [PERIOD + 1]. 'PERIOD' must not be 0x0.

9.1.21 PWM Status Register 0 (Channel 0 to Channel 3) (PST0)

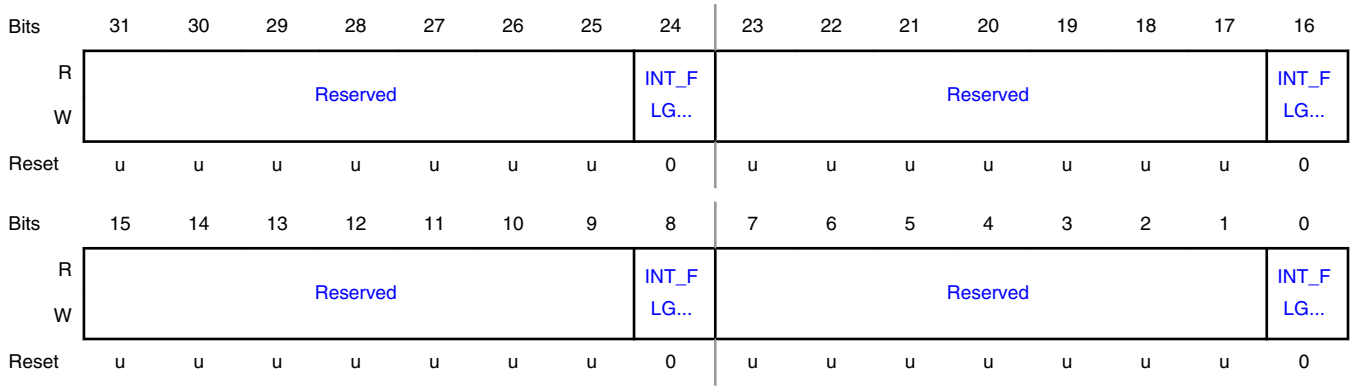
Offset

Register	Offset
PST0	4Ch

Function

This register shows the interrupt status for channels 0 to channel 3. The interrupt status can also be cleared with this register.

Diagram



Fields

Field	Function
31-25 —	RESERVED Reserved. The value read from a reserved bit is not defined.
24 INT_FLG_3	PWM Channel 3 Interrupt Flag Write 1 to clear the interrupt. 0b - No interrupt pending. 1b - Interrupt pending.
23-17 —	RESERVED Reserved. The value read from a reserved bit is not defined.
16 INT_FLG_2	PWM Channel 2 Interrupt Flag Write 1 to clear the interrupt. 0b - No interrupt pending. 1b - Interrupt pending.
15-9 —	RESERVED Reserved. The value read from a reserved bit is not defined.
8 INT_FLG_1	PWM Channel 1 Interrupt Flag Write 1 to clear the interrupt. 0b - No interrupt pending. 1b - Interrupt pending.
7-1 —	RESERVED Reserved. The value read from a reserved bit is not defined.
0 INT_FLG_0	PWM Channel 0 Interrupt Flag Write 1 to clear the interrupt. 0b - No interrupt pending. 1b - Interrupt pending.

9.1.22 PWM Status Register 1 (Channel 4 to Channel 7) (PST1)

Offset

Register	Offset
PST1	50h

Function

This register shows the interrupt status for channels 4 to channel 7. The interrupt status can also be cleared with this register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved							INT_FLG...	Reserved							INT_FLG...
W																
Reset	u	u	u	u	u	u	u	0	u	u	u	u	u	u	u	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved							INT_FLG...	Reserved							INT_FLG...
W																
Reset	u	u	u	u	u	u	u	0	u	u	u	u	u	u	u	0

Fields

Field	Function
31-25 —	RESERVED Reserved. The value read from a reserved bit is not defined.
24 INT_FLG_7	PWM Channel 7 Interrupt Flag Write 1 to clear the interrupt. 0b - No interrupt pending. 1b - Interrupt pending.
23-17 —	RESERVED Reserved. The value read from a reserved bit is not defined.
16 INT_FLG_6	PWM Channel 6 Interrupt Flag Write 1 to clear the interrupt. 0b - No interrupt pending. 1b - Interrupt pending.
15-9 —	RESERVED Reserved. The value read from a reserved bit is not defined.
8 INT_FLG_5	PWM Channel 5 Interrupt Flag Write 1 to clear the interrupt. 0b - No interrupt pending. 1b - Interrupt pending.

Table continues on the next page...

Table continued from the previous page...

Field	Function
7-1 —	RESERVED Reserved. The value read from a reserved bit is not defined.
0 INT_FLG_4	PWM Channel 4 Interrupt Flag Write 1 to clear the interrupt. 0b - No interrupt pending. 1b - Interrupt pending.

9.1.23 PWM Status Register 2 (Channel 8 to Channel 10) (PST2)

Offset

Register	Offset
PST2	54h

Function

This register shows the interrupt status for channels 8 to Channel 10. The interrupt status can also be cleared with this register.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								Reserved							INT_FLG...
Reset	u	u	u	u	u	u	u	0	u	u	u	u	u	u	u	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved							INT_FLG...	Reserved							INT_FLG...
Reset	u	u	u	u	u	u	u	0	u	u	u	u	u	u	u	0

Fields

Field	Function
31-24 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

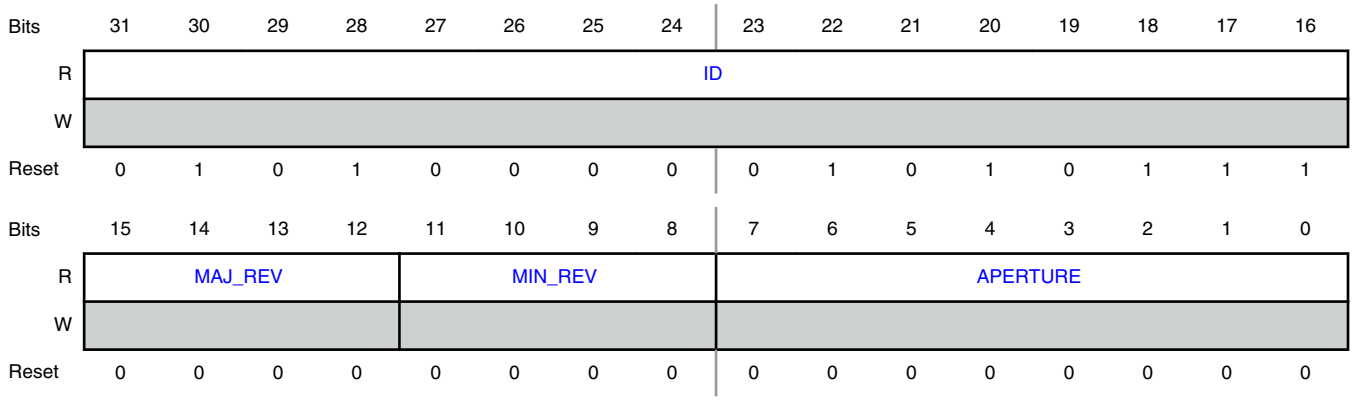
Field	Function
23-17 —	RESERVED Reserved. The value read from a reserved bit is not defined.
16 INT_FLG_10	PWM Channel 10 Interrupt Flag Write 1 to clear the interrupt. 0b - No interrupt pending. 1b - Interrupt pending.
15-9 —	RESERVED Reserved. The value read from a reserved bit is not defined.
8 INT_FLG_9	PWM Channel 9 Interrupt Flag Write 1 to clear the interrupt. 0b - No interrupt pending. 1b - Interrupt pending.
7-1 —	RESERVED Reserved. The value read from a reserved bit is not defined.
0 INT_FLG_8	PWM Channel 8 Interrupt Flag Write 1 to clear the interrupt. 0b - No interrupt pending. 1b - Interrupt pending.

9.1.24 PWM Module Identifier Register ('PW' in ASCII) (MODULE_ID)

Offset

Register	Offset
MODULE_ID	FFCh

Diagram



Fields

Field	Function
31-16 ID	Identifier This is the unique identifier of the module.
15-12 MAJ_REV	Major Revision Major revision implies software modifications.
11-8 MIN_REV	Minor Revision Minor revision without software consequences.
7-0 APERTURE	Aperture Aperture number minus 1 of consecutive packets 4 KB reserved for this IP

Chapter 10
Standard Counter/Timers (CT32B)

10.1 CT32B register descriptions

10.1.1 CT32B memory map

CT32B0 base address: 4002_1000h

CT32B1 base address: 4002_2000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Interrupt Register (IR)	32	RW	See description
4h	Timer Control Register (TCR)	32	RW	See description
8h	Timer Counter Register (TC)	32	RW	0000_0000h
Ch	Prescale Register (PR)	32	RW	0000_0000h
10h	Prescale Counter Register (PC)	32	RW	0000_0000h
14h	Match Control Register (MCR)	32	RW	See description
18h	Match Register 0 (MR0)	32	RW	0000_0000h
1Ch	Match Register 1 (MR1)	32	RW	0000_0000h
20h	Match Register 2 (MR2)	32	RW	0000_0000h
24h	Match Register 3 (MR3)	32	RW	0000_0000h
28h	Capture Control Register (CCR)	32	RW	See description
2Ch	Capture Register 0 (CR0)	32	RO	0000_0000h
30h	Capture Register 1 (CR1)	32	RO	0000_0000h
3Ch	External Match Register (EMR)	32	RW	See description
70h	Count Control Register (CTCR)	32	RW	See description
74h	PWM Control Register (PWMC)	32	RW	See description

10.1.2 Interrupt Register (IR)

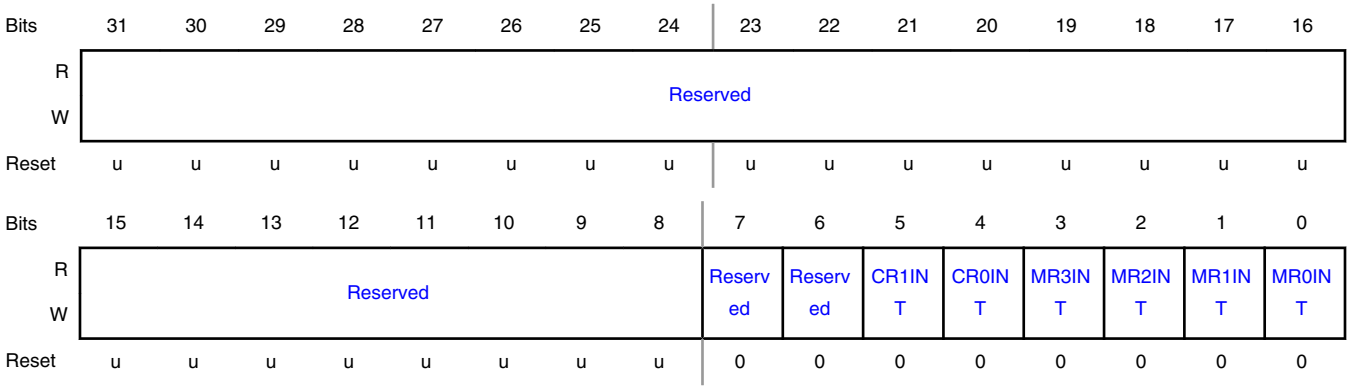
Offset

Register	Offset
IR	0h

Function

The Interrupt Register consists of 4 bits for the match interrupts and 4 bits for the capture interrupts. If an interrupt is generated then the corresponding bit in the IR will be high. Otherwise, the bit will be low. Writing a logic one to the corresponding IR bit will reset the interrupt. Writing a zero has no effect. The act of clearing an interrupt for a timer match also clears any corresponding DMA request. Writing a zero has no effect.

Diagram



Fields

Field	Function
31-8 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
7 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
6 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
5 CR1INT	Interrupt Flag for Capture Channel 1 Event

Table continues on the next page...

Table continued from the previous page...

Field	Function
4 CR0INT	Interrupt Flag for Capture Channel 0 Event
3-0 MRnINT	Interrupt Flag for Match Channel n

10.1.3 Timer Control Register (TCR)

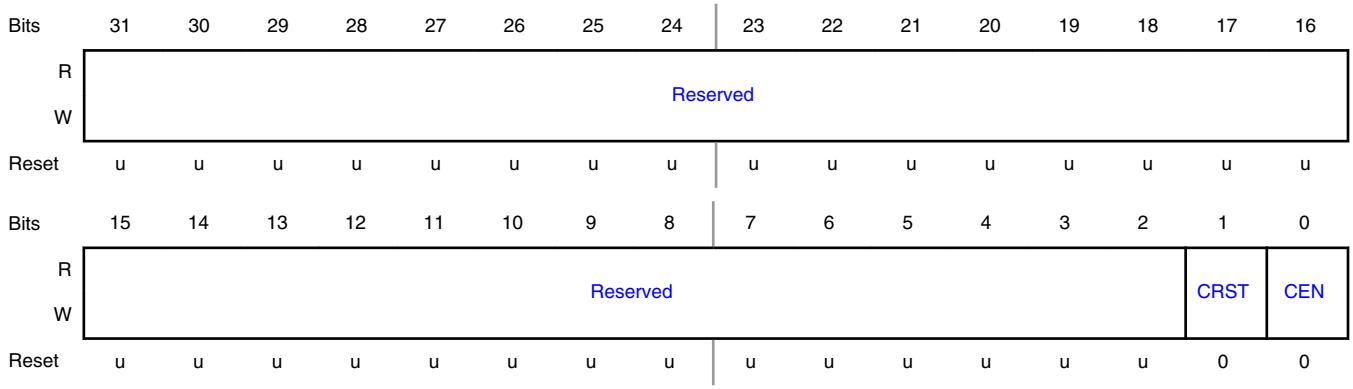
Offset

Register	Offset
TCR	4h

Function

The TCR is used to control the Timer Counter functions. The Timer Counter can be disabled or reset through the TCR.

Diagram



Fields

Field	Function
31-2 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
1 CRST	Counter Reset 0b - Disabled. Do nothing. 1b - Enabled. The Timer Counter and the Prescale Counter are synchronously reset on the next positive edge of the APB bus clock. The counters remain reset until TCR[1] is returned to zero.
0 CEN	Counter Enable 0b - Disabled.The counters are disabled. 1b - Enabled. The Timer Counter and Prescale Counter are enabled.

10.1.4 Timer Counter Register (TC)

Offset

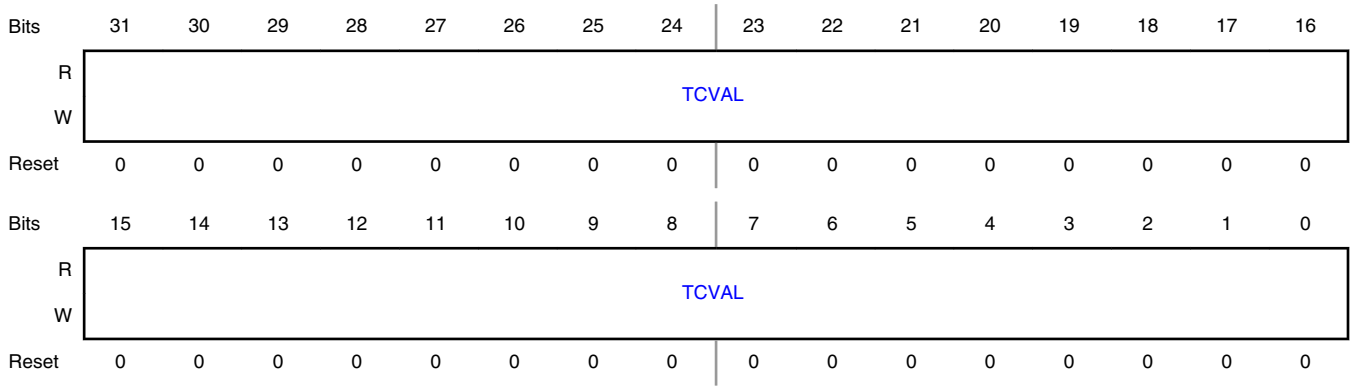
Register	Offset
TC	8h

Function

The 32-bit Timer Counter register is incremented when the prescale counter reaches its terminal count; PR+1 cycles of the APB bus clock. Unless it is reset before reaching its upper limit, the Timer Counter will count up through the value 0xFFFF FFFF and then wrap back to the value 0x0000 0000. This event does not cause an interrupt, but a match register can be used to detect an overflow if needed.

The TC is controlled through the TCR register.

Diagram



Fields

Field	Function
31-0 TCVAL	Timer Counter Value

10.1.5 Prescale Register (PR)

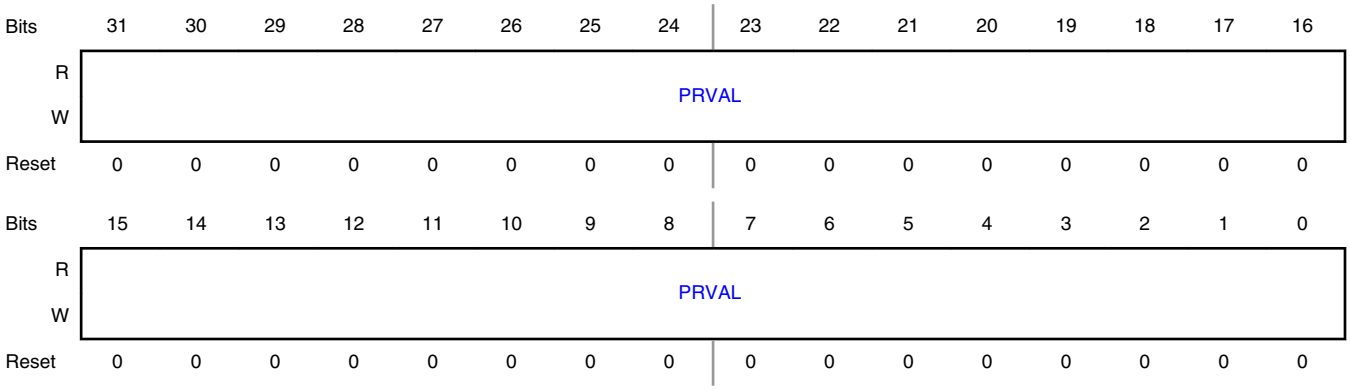
Offset

Register	Offset
PR	Ch

Function

The 32-bit Prescale register specifies the maximum value for the Prescale Counter. When the Prescale Counter (PC) is equal to this value, the next clock increments the TC and clears the PC.

Diagram



Fields

Field	Function
31-0 PRVAL	Prescale Counter Value

10.1.6 Prescale Counter Register (PC)

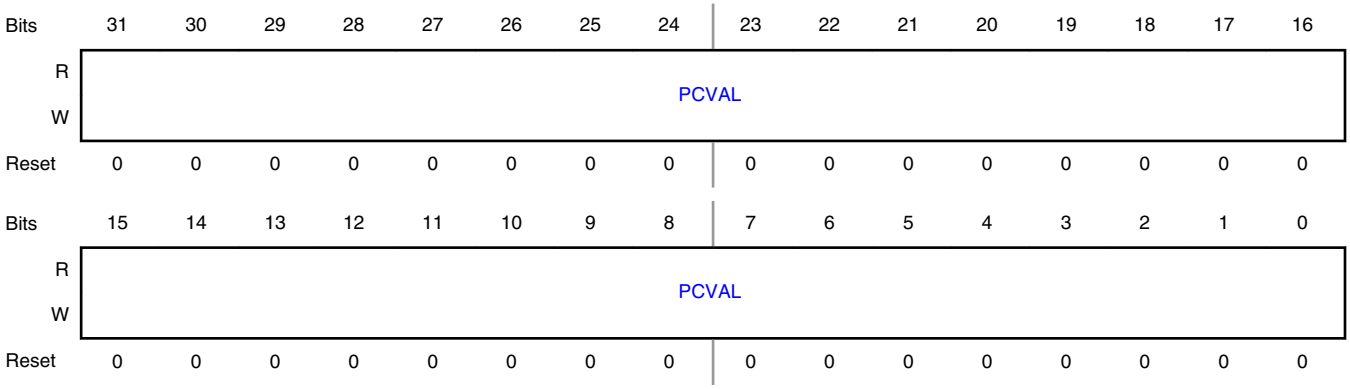
Offset

Register	Offset
PC	10h

Function

This register controls division of the APB bus clock by some constant value before it is applied to the Timer Counter. This allows control of the relationship of the resolution of the timer versus the maximum time before the timer overflows. The Prescale Counter is incremented on every APB bus clock. When it reaches the value stored in the Prescale register, the Timer Counter is incremented and the Prescale Counter is reset on the next APB bus clock. This causes the Timer Counter to increment on every APB bus clock when PR = 0, every 2 APB bus clocks when PR = 1, etc.

Diagram



Fields

Field	Function
31-0 PCVAL	Prescale Counter Value

10.1.7 Match Control Register (MCR)

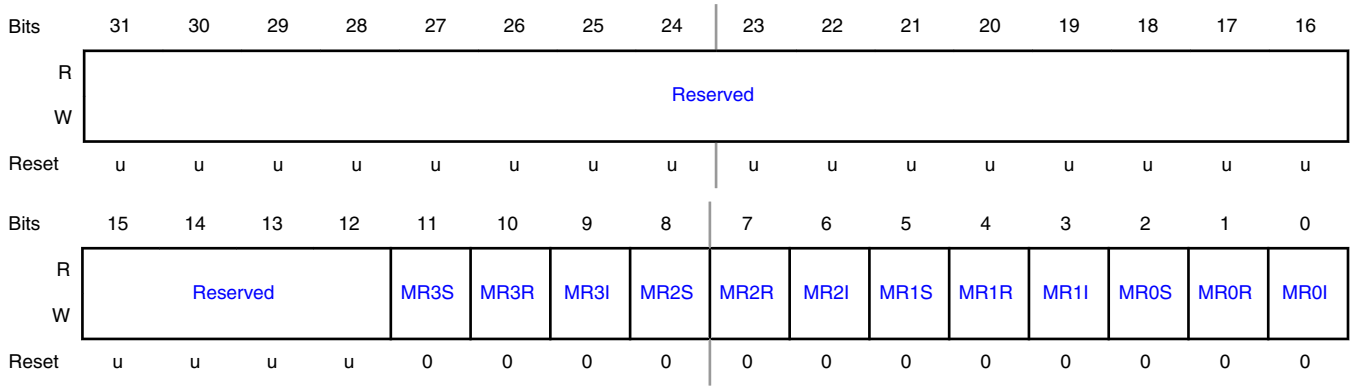
Offset

Register	Offset
MCR	14h

Function

The Match Control Register is used to control what operations are performed when one of the Match Registers matches the Timer Counter.

Diagram



Fields

Field	Function
31-12 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
11 MR3S	Stop on MR3 The TC and PC will be stopped and TCR[0] will be set to 0 if MR3 matches the TC. 0b - Disabled. 1b - Enabled.
10 MR3R	Reset on MR3 The TC will be reset if MR3 matches it. 0b - Disabled. 1b - Enabled.
9 MR3I	Interrupt on MR3 An interrupt is generated when MR3 matches the value in the TC. 0b - Disabled. 1b - Enabled.
8 MR2S	Stop on MR2 The TC and PC will be stopped and TCR[0] will be set to 0 if MR2 matches the TC. 0b - Disabled. 1b - Enabled.

Table continues on the next page...

Table continued from the previous page...

Field	Function
7 MR2R	Reset on MR2 The TC will be reset if MR2 matches it. 0b - Disabled. 1b - Enabled.
6 MR2I	Interrupt on MR2 An interrupt is generated when MR2 matches the value in the TC. 0b - Disabled. 1b - Enabled.
5 MR1S	Stop on MR1 The TC and PC will be stopped and TCR[0] will be set to 0 if MR1 matches the TC. 0b - Disabled. 1b - Enabled.
4 MR1R	Reset on MR1 The TC will be reset if MR1 matches it. 0b - Disabled. 1b - Enabled.
3 MR1I	Interrupt on MR1 An interrupt is generated when MR1 matches the value in the TC. 0b - Disabled. 1b - Enabled.
2 MR0S	Stop on MR0 The TC and PC will be stopped and TCR[0] will be set to 0 if MR0 matches the TC. 0b - Disabled. 1b - Enabled.
1 MR0R	Reset on MR0 The TC will be reset if MR0 matches it. 0b - Disabled. 1b - Enabled.

Table continues on the next page...

Table continued from the previous page...

Field	Function
0 MR0I	Interrupt on MR0 An interrupt is generated when MR0 matches the value in the TC. 0b - Disabled. 1b - Enabled.

10.1.8 Match Register 0 (MR0)

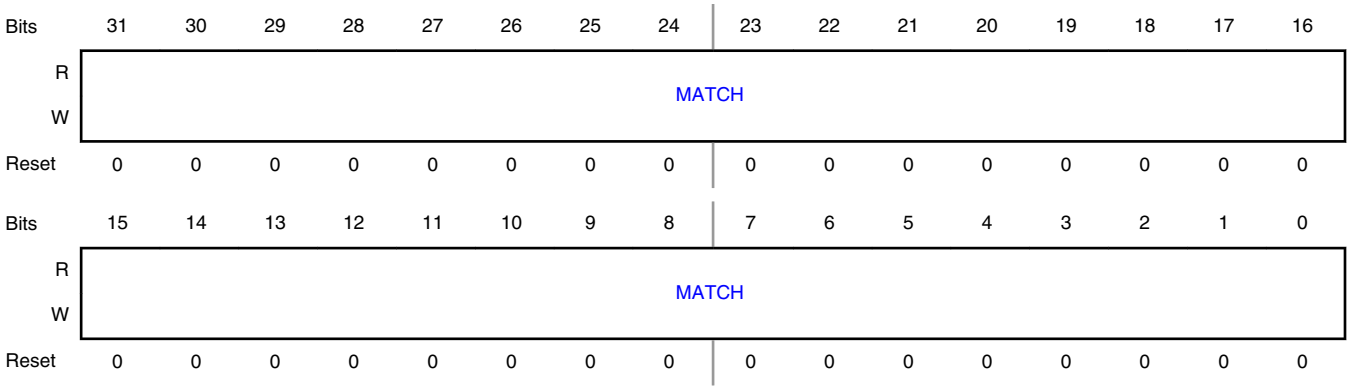
Offset

Register	Offset
MR0	18h

Function

The Match register value, MR0, is continuously compared to the Timer Counter (TC) value. When the two values are equal, actions can be triggered automatically. The action possibilities are to generate an interrupt, reset the Timer Counter, or stop the timer. Actions are controlled by the settings in the MCR register.

Diagram



Fields

Field	Function
31-0 MATCH	Timer counter match value.

10.1.9 Match Register 1 (MR1)

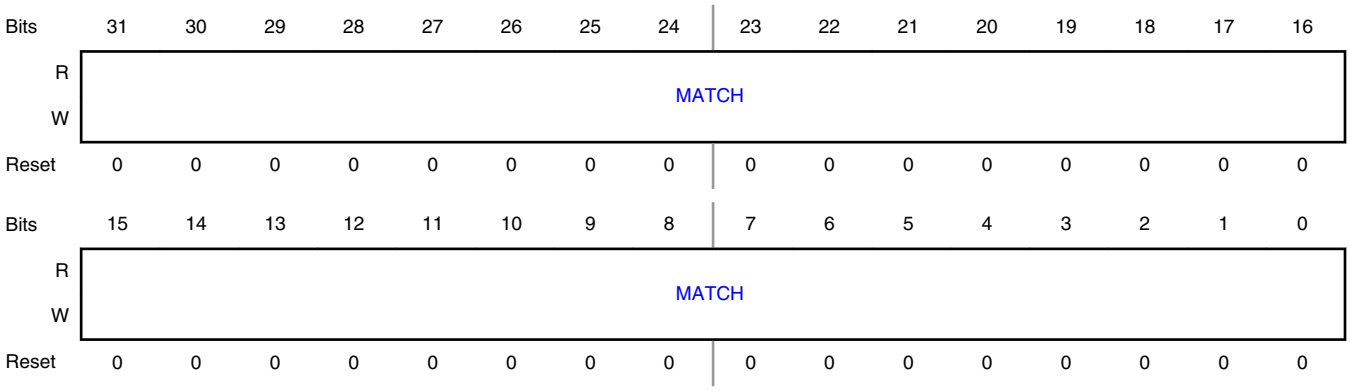
Offset

Register	Offset
MR1	1Ch

Function

The Match register value, MR1, is continuously compared to the Timer Counter (TC) value. When the two values are equal, actions can be triggered automatically. The action possibilities are to generate an interrupt, reset the Timer Counter, or stop the timer. Actions are controlled by the settings in the MCR register.

Diagram



Fields

Field	Function
31-0 MATCH	Timer counter match value.

10.1.10 Match Register 2 (MR2)

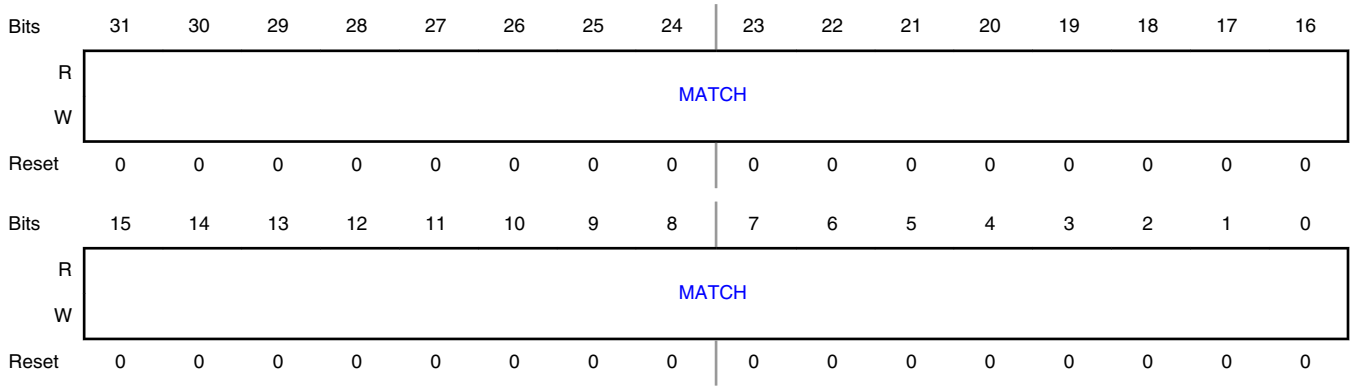
Offset

Register	Offset
MR2	20h

Function

The Match register value, MR2, is continuously compared to the Timer Counter (TC) value. When the two values are equal, actions can be triggered automatically. The action possibilities are to generate an interrupt, reset the Timer Counter, or stop the timer. Actions are controlled by the settings in the MCR register.

Diagram



Fields

Field	Function
31-0 MATCH	Timer counter match value.

10.1.11 Match Register 3 (MR3)

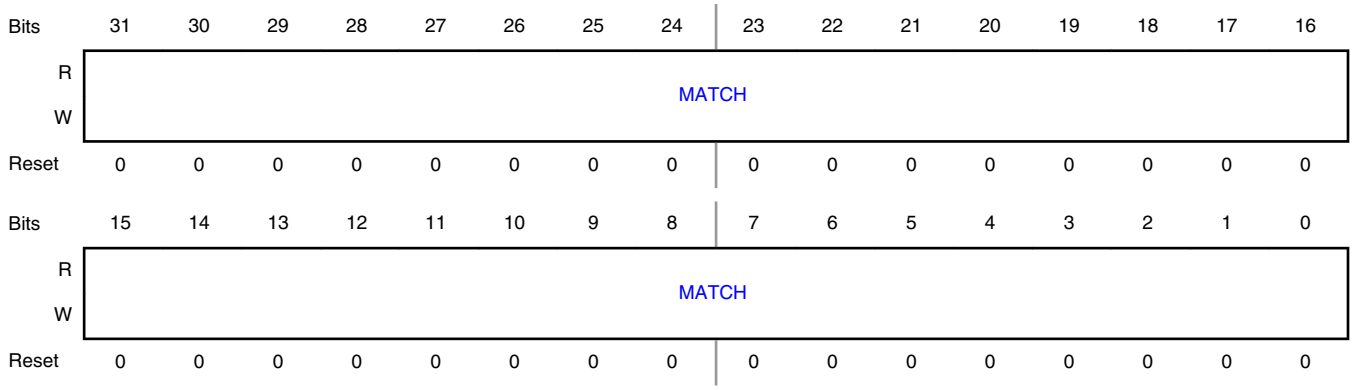
Offset

Register	Offset
MR3	24h

Function

The Match register value, MR3, is continuously compared to the Timer Counter (TC) value. When the two values are equal, actions can be triggered automatically. The action possibilities are to generate an interrupt, reset the Timer Counter, or stop the timer. Actions are controlled by the settings in the MCR register.

Diagram



Fields

Field	Function
31-0 MATCH	Timer counter match value.

10.1.12 Capture Control Register (CCR)

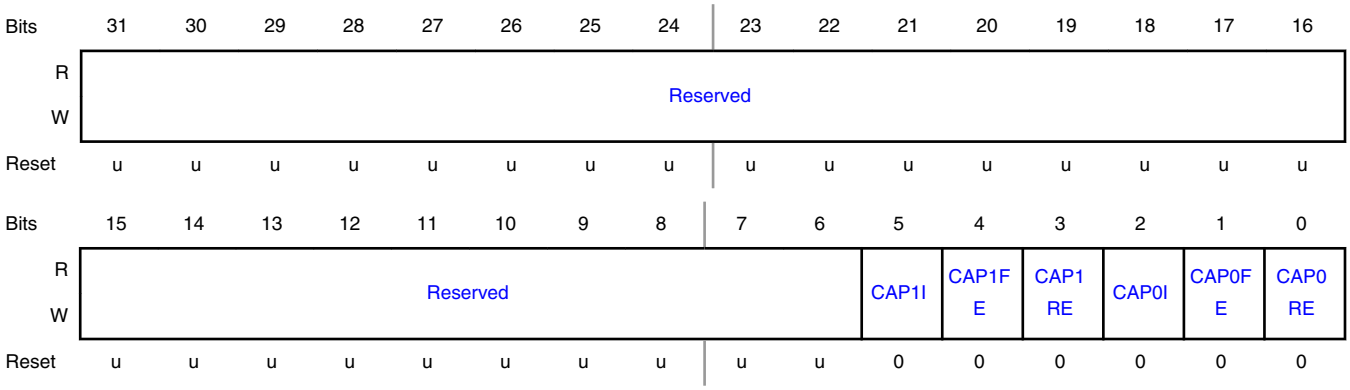
Offset

Register	Offset
CCR	28h

Function

The Capture Control Register is used to control whether one of the two Capture Registers is loaded with the value in the Timer Counter when the capture event occurs, and whether an interrupt is generated by the capture event. Setting both the rising and falling bits at the same time is a valid configuration, resulting in a capture event for both edges. Note: If Counter mode is selected for a particular CAP input in the CTCR, the 3 bits for that input in this register should be programmed as 000b, but capture and/or interrupt can be selected for the other CAP inputs.

Diagram



Fields

Field	Function
31-6 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
5 CAP1I	Generate Interrupt on Channel 1 Capture Event If set, a CR1 load generates an interrupt.

Table continues on the next page...

Table continued from the previous page...

Field	Function
4 CAP1FE	Falling Edge of Capture Channel 1 A sequence of 1 then 0 causes CR1 to be loaded with the contents of TC. 0b - Disabled. 1b - Enabled.
3 CAP1RE	Rising Edge of Capture Channel 1 A sequence of 0 then 1 causes CR1 to be loaded with the contents of TC. 0b - Disabled. 1b - Enabled.
2 CAP0I	Generate Interrupt on Channel 0 Capture Event If set, a CR0 load generates an interrupt.
1 CAP0FE	Falling Edge of Capture Channel 0 A sequence of 1 then 0 causes CR0 to be loaded with the contents of TC. 0b - Disabled. 1b - Enabled.
0 CAP0RE	Rising Edge of Capture Channel 0 A sequence of 0 then 1 causes CR0 to be loaded with the contents of TC. 0b - Disabled. 1b - Enabled.

10.1.13 Capture Register 0 (CR0)

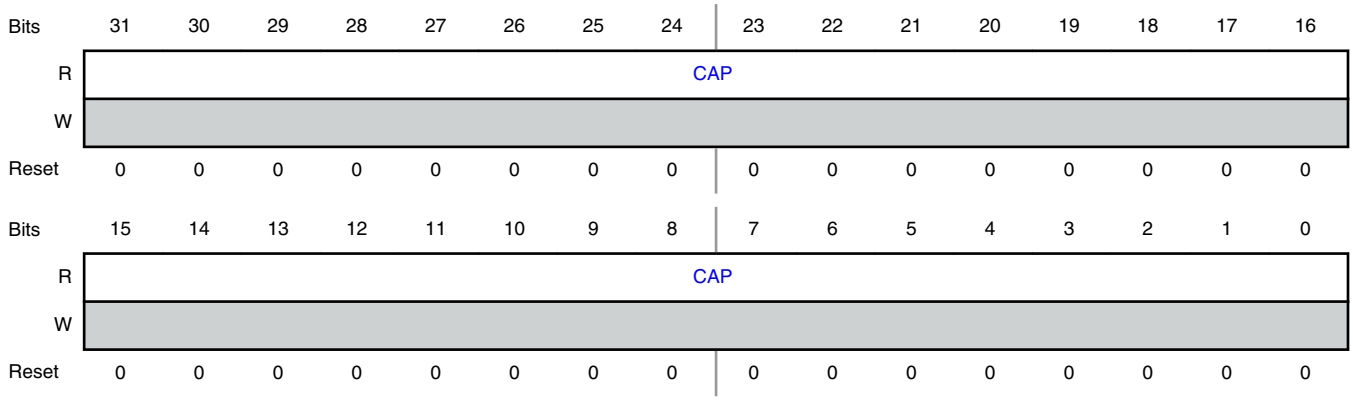
Offset

Register	Offset
CR0	2Ch

Function

Capture register 0 is associated with capture channel 0 and may be loaded with the counter/timer value when a specified event occurs on the signal defined for capture channel 0. The signal could originate from an external pin or from an internal source. The settings in the Capture Control Register (CCR) determine whether the capture function is enabled, and whether a capture event happens on the rising edge, the falling edge, or on both edges of the associated signal.

Diagram



Fields

Field	Function
31-0 CAP	Timer Counter Capture Value

10.1.14 Capture Register 1 (CR1)

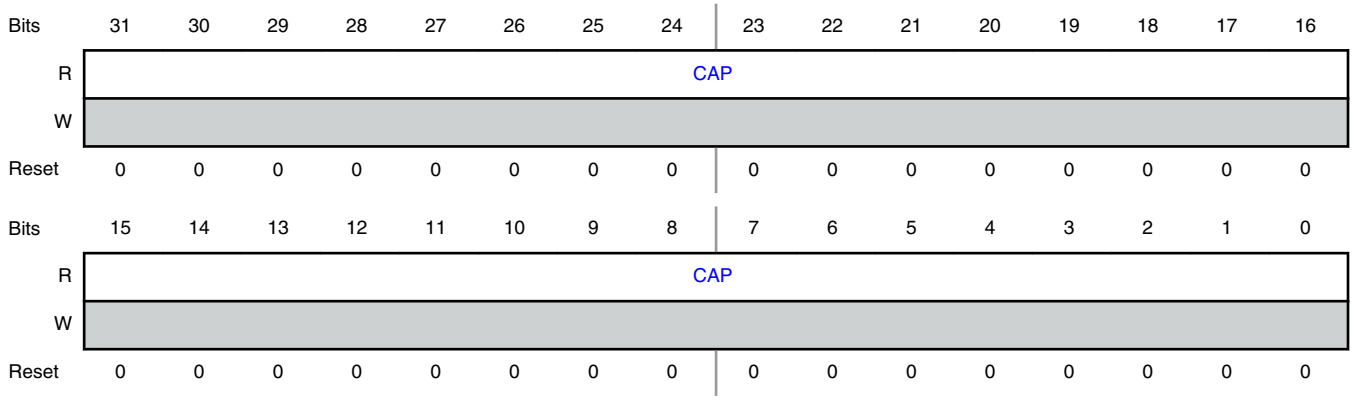
Offset

Register	Offset
CR1	30h

Function

Capture register 1 is associated with capture channel 1 and may be loaded with the counter/timer value when a specified event occurs on the signal defined for capture channel 1. The signal could originate from an external pin or from an internal source. The settings in the Capture Control Register (CCR) determine whether the capture function is enabled, and whether a capture event happens on the rising edge, the falling edge, or on both edges of the associated signal.

Diagram



Fields

Field	Function
31-0 CAP	Timer Counter Capture Value

10.1.15 External Match Register (EMR)

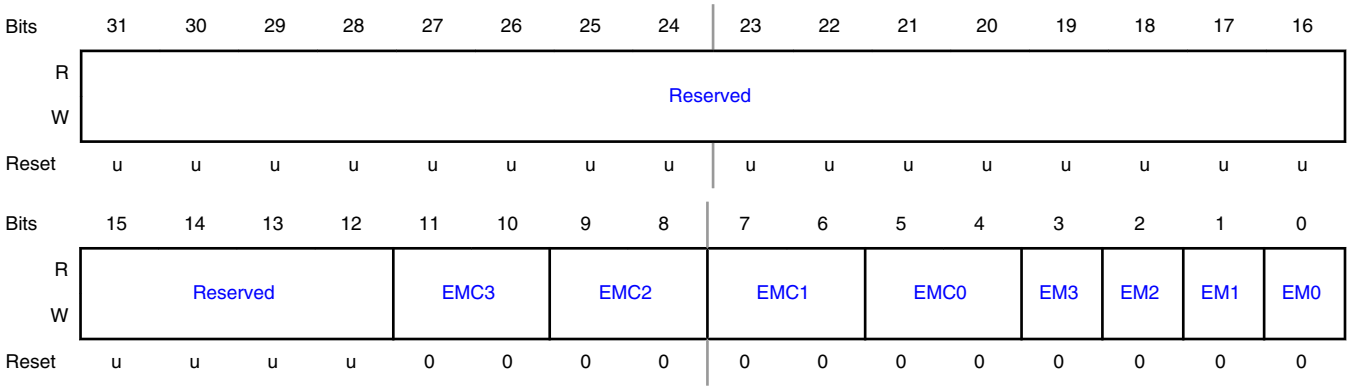
Offset

Register	Offset
EMR	3Ch

Function

The External Match Register provides both control and status of the external match pins. Match events for Match 0 and Match 1 in each timer can cause a DMA request. Match 0 and Match 1 outputs can also be output to a device pin with the necessary configuration.

Diagram



Fields

Field	Function
31-12 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
11-10: EMC3 9-8: EMC2 7-6: EMC1 5-4: EMC0	External Match Control n Determines the functionality of External Match n. 00b - Do Nothing. 01b - Clear. Clear the corresponding External Match bit/output to 0 (MAT0 pin is LOW if pinned out). 10b - Set. Set the corresponding External Match bit/output to 1 (MAT0 pin is HIGH if pinned out). 11b - Toggle. Toggle the corresponding External Match bit/output.
3-0 EMn	External Match n This bit reflects the state of output MATn, whether or not this output is connected to a pin. When a match occurs between the TC and MRn, this bit can either toggle, go LOW, go HIGH, or do nothing, as selected by EMCn. This bit is driven to the MAT pins if the match function is selected via IOCON. 0b - LOW. 1b - HIGH.

10.1.16 Count Control Register (CTCR)

Offset

Register	Offset
CTCR	70h

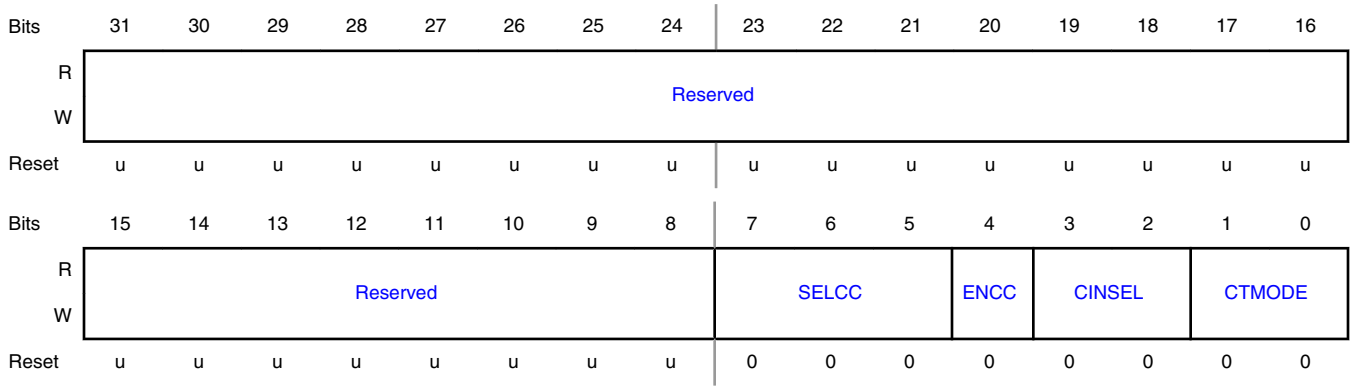
Function

When Counter Mode is chosen as a mode of operation, the CAP input (selected by the CINSEL bit) is sampled on every rising edge of the APB bus clock. After comparing two consecutive samples of this CAP input, one of the following four events is recognized: rising edge, falling edge, either of edges or no changes in the level of the selected CAP input. Only if the identified event occurs and the event corresponds to the one selected by CTMODE field, will the Timer Counter register be incremented.

Effective processing of the externally supplied clock to the counter has some limitations. Since two successive rising edges of the APB bus clock are used to identify only one edge on the CAP selected input, the frequency of the CAP input cannot exceed one half of the APB bus clock. Consequently, duration of the HIGH/LOW levels on the same CAP input in this case cannot be shorter than 1/APB bus clock.

Bits 7:4 of this register are also used to enable and configure the capture-clears-timer feature. This feature allows for a designated edge on a particular CAP input to reset the timer to all zeros. Using this mechanism to clear the timer on the leading edge of an input pulse and performing a capture on the trailing edge, permits direct pulse-width measurement using a single capture input without the need to perform a subtraction operation in software.

Diagram



Fields

Field	Function
31-8 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
7-5 SELCC	Edge Select When bit 4 is 1, these bits select which capture input edge will cause the timer and prescaler to be cleared. These bits have no effect when bit 4 is low. 000b - Channel 0 Rising Edge. Rising edge of the signal on capture channel 0 clears the timer (if bit 4 is set). 001b - Channel 0 Falling Edge. Falling edge of the signal on capture channel 0 clears the timer (if bit 4 is set). 010b - Channel 1 Rising Edge. Rising edge of the signal on capture channel 1 clears the timer (if bit 4 is set). 011b - Channel 1 Falling Edge. Falling edge of the signal on capture channel 1 clears the timer (if bit 4 is set). 100b - Reserved. 101b - Reserved. 110b - Reserved. 111b - Reserved.
4 ENCC	Enable Clearing Timer and Prescaler Setting this bit to 1 enables clearing of the timer and the prescaler when the capture-edge event specified in bits 7:5 occurs.

Table continues on the next page...

Table continued from the previous page...

Field	Function
3-2 CINSEL	Count Input Select Count Input Select When bits 1:0 in this register are not 00, these bits select which CAP pin is sampled for clocking. Note: If Counter mode is selected for a particular CAPn input in the CTCR, the 3 bits for that input in the Capture Control Register (CCR) must be programmed as 000. However, capture and/or interrupt can be selected for the other CAPn input in the same timer. 00b - Channel 0. CAPn.0 for CT32Bn. 01b - Channel 1. CAPn.1 for CT32Bn. 10b - Reserved. 11b - Reserved.
1-0 CTMODE	Counter/Timer Mode This field selects which rising APB bus clock edges can increment Timer s Prescale Counter (PC), or clear PC and increment Timer Counter (TC). Timer Mode: the TC is incremented when the Prescale Counter matches the Prescale Register. 00b - Timer Mode. Incremented every rising APB bus clock edge. 01b - Counter Mode rising edge. TC is incremented on rising edges on the CAP input selected by bits 3:2. 10b - Counter Mode falling edge. TC is incremented on falling edges on the CAP input selected by bits 3:2. 11b - Counter Mode dual edge. TC is incremented on both edges on the CAP input selected by bits 3:2.

10.1.17 PWM Control Register (PWMC)

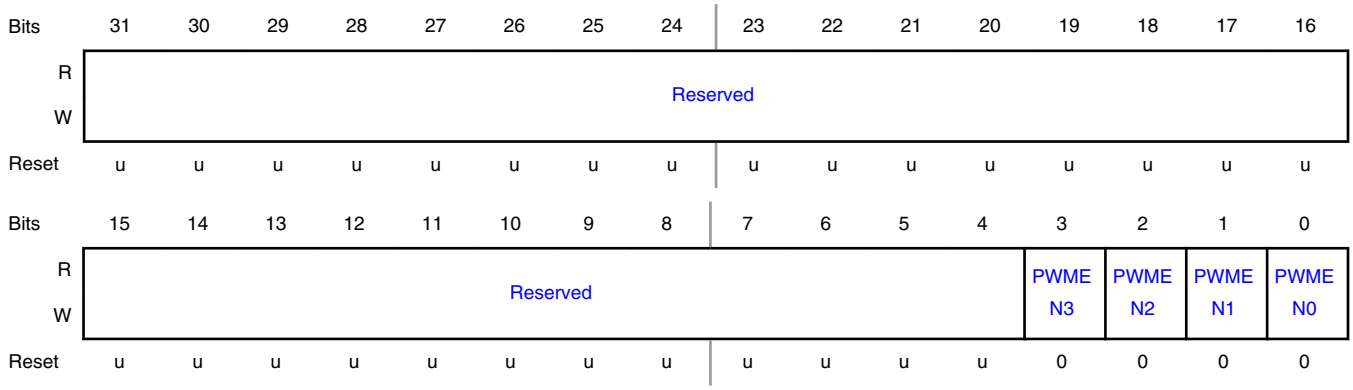
Offset

Register	Offset
PWMC	74h

Function

The PWM Control Register is used to configure the match outputs as PWM outputs. Each match output can be independently set to perform either as PWM output or as match output whose function is controlled by the External Match Register (EMR). For each timer, a maximum of two single edge controlled PWM outputs can be selected on the MATn.1:0 outputs. One additional match register determines the PWM cycle length. When a match occurs in any of the other match registers, the PWM output is set to HIGH. The timer is reset by the match register that is configured to set the PWM cycle length. When the timer is reset to zero, all currently HIGH match outputs configured as PWM outputs are cleared.

Diagram



Fields

Field	Function
31-4 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
3-0 PWMENn	PWM Mode Enable for channel n **NOTE** It is recommended to use match channel 3 to set the PWM cycle. 0b - Match. MATn is controlled by EMn. 1b - PWM. PWM mode is enabled for MATn.

Chapter 11
Windowed Watchdog Timer (WWDT)

11.1 WWDT register descriptions

11.1.1 WWDT memory map

WWDT base address: 4000_A000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Watchdog Mode Register (MOD)	32	RW	See description
4h	Watchdog Timer Constant Register (TC)	32	RW	See description
8h	Watchdog Feed Sequence Register (FEED)	32	WO	See description
Ch	Watchdog Timer Value Register (TV)	32	RO	See description
14h	Watchdog Warning Interrupt Compare Value Register (WARNINT)	32	RW	See description
18h	Watchdog Timer Window Register (WINDOW)	32	RW	See description

11.1.2 Watchdog Mode Register (MOD)

Offset

Register	Offset
MOD	0h

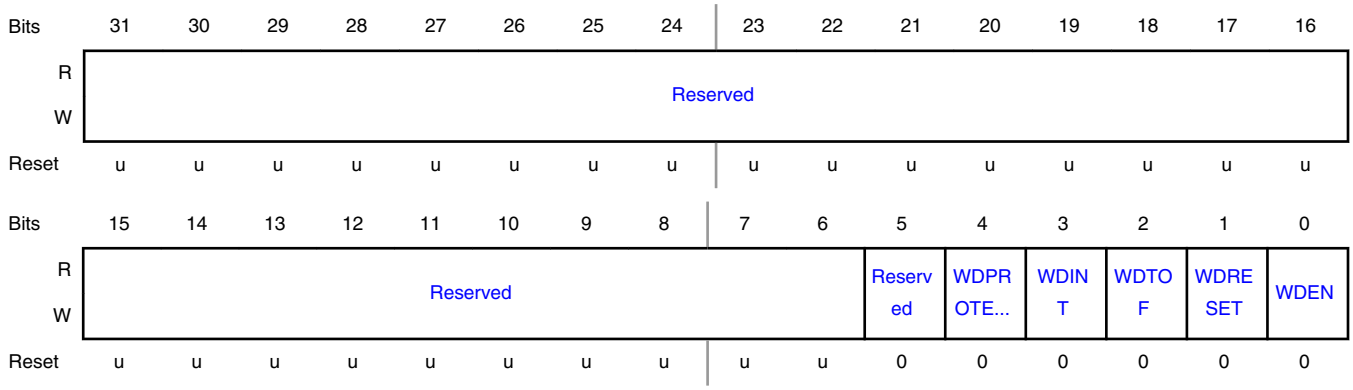
Function

This register contains the basic mode and status of the Watchdog Timer.

NOTE

A watchdog feed must be performed before any changes to the MOD register take effect.

Diagram



Fields

Field	Function
31-6 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
5 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
4 WDPROTECT	Watchdog Update Mode This bit can be set once by software and is only cleared by a reset. 0b - Flexible. A feed sequence can be performed at any time; ie. the watchdog timer can be reloaded with time-out value (TC) at any time. 1b - Threshold. A feed sequence can be performed only after the counter is below the value of WDWARNINT and WDWINDOW; ie. the watchdog timer can be reloaded with time-out value (TC) only when the counter timer value is below the value of WDWARNINT and WDWINDOW, otherwise a 'feed error' is created.
3 WDINT	Warning Interrupt Flag Set when the timer reaches the value in WDWARNINT. Cleared by software writing a 1 to this bit position. Note that this bit cannot be cleared while the WARNINT value is equal to the value of the TV register. This can occur if the value of WARNINT is 0 and the WDRESET bit is 0 when TV decrements to 0.
2 WDTOF	Watchdog Time-out Flag Set when the watchdog timer times out, by a feed error, or by events associated with WDPROTECT. Cleared by software writing a 0 to this bit position. Causes a chip reset if WDRESET = 1.

Table continues on the next page...

Table continued from the previous page...

Field	Function
1 WDRESET	Watchdog Reset Enable Once this bit has been written with a 1 it cannot be re-written with a 0, and will only be cleared by an external reset or a watchdog timer reset. 0b - Interrupt. A watchdog time-out will not cause a chip reset. It will cause an interrupt of the watchdog. 1b - Reset. A watchdog time-out will cause a chip reset.
0 WDEN	Watchdog Enable Once this bit is set to one and a watchdog feed is performed, the watchdog timer will run permanently. 0b - The watchdog timer is stopped. 1b - The watchdog timer is running.

11.1.3 Watchdog Timer Constant Register (TC)

Offset

Register	Offset
TC	4h

Function

The TC register determines the time-out value. A feed sequence is required to transfer the TC value into the Watchdog counter. The TC resets to 0x00_00FF. Writing a value below 0xFF will cause 0x00_00FF to be loaded into the TC. Thus the minimum time-out interval is T_{WDCLK} X 256 x 4.

If the MOD[WDPROTECT] = 1, an attempt to change the value of TC before the watchdog counter is below the values of WARNINT[WARNINT] and WINDOW[WINDOW] will cause a watchdog reset and set the MOD[WDTOF] flag.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								COUNT							
Reset	u	u	u	u	u	u	u	u	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	COUNT															
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Fields

Field	Function
31-24 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
23-0 COUNT	Watchdog Time-out Value If MOD_WDPROTECT is set, changing this value may cause an error.

11.1.4 Watchdog Feed Sequence Register (FEED)

Offset

Register	Offset
FEED	8h

Function

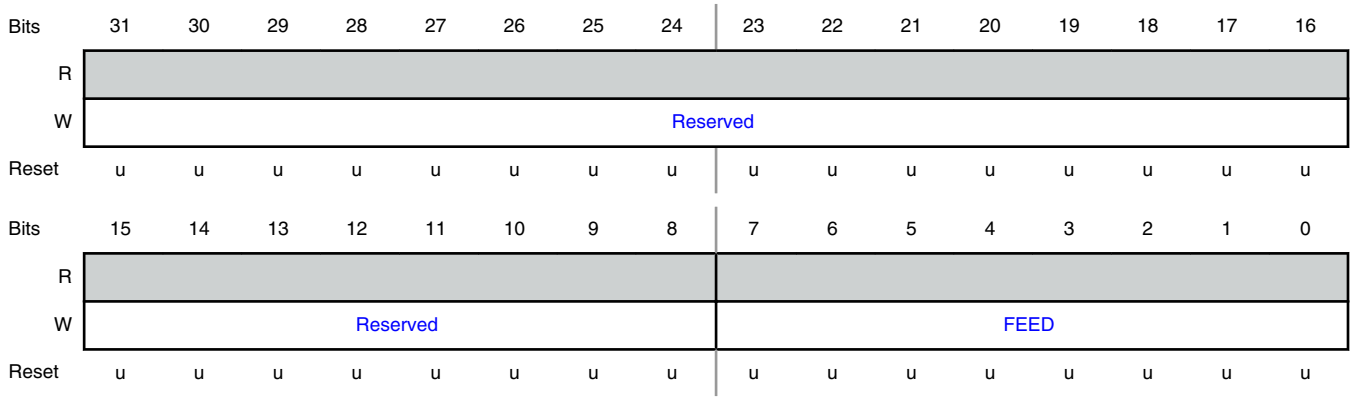
Writing 0xAA followed by 0x55 to this register reloads the Watchdog timer with the value contained in TC. This operation will also start the Watchdog if it is enabled via the MOD register. Setting the MOD[WDEN] bit is not sufficient to enable the Watchdog. A valid feed sequence must be completed after setting MOD[WDEN] before the Watchdog is capable of generating a reset. Until then, the Watchdog will ignore feed errors.

Note that a value in the WINDOW register that is smaller than the default may limit the time when a watchdog feed is allowed.

After writing 0xAA to FEED, access to any Watchdog register other than writing 0x55 to FEED causes an immediate reset/interrupt when the Watchdog is enabled, and sets the MOD[WDTOF] flag. The reset will be generated during the second APB bus clock following an incorrect access to a Watchdog register during a feed sequence.

It is good practice to disable interrupts around a feed sequence, if the application is such that an interrupt might result in rescheduling processor control away from the current task in the middle of the feed, and then lead to some other access to the WDT before control is returned to the interrupted task.

Diagram



Fields

Field	Function
31-8 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
7-0 FEED	Feed Value Feed value should be 0xAA followed by 0x55. Writing 0xAA followed by 0x55 to this register will reload the Watchdog timer with the TC value. This operation will also start the Watchdog if it is enabled via the WDMOD register. Setting the WDEN bit in the WDMOD register is not sufficient to enable the Watchdog. A valid feed sequence must be completed after setting WDEN before the Watchdog is capable of generating a reset. Until then, the Watchdog will ignore feed errors.

11.1.5 Watchdog Timer Value Register (TV)

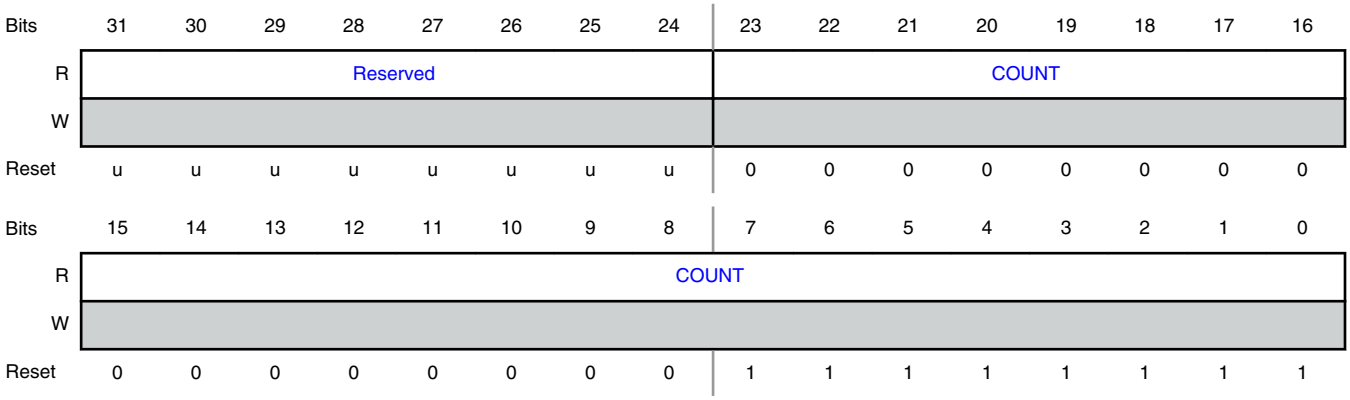
Offset

Register	Offset
TV	Ch

Function

This 24-bit register reads out the current value of the Watchdog timer. When reading the value of the 24-bit counter, the lock and synchronization procedure takes up to 6 WDCLK cycles plus 6 APB bus clock cycles, so the value of TV is older than the actual value of the timer when it's being read by the CPU.

Diagram



Fields

Field	Function
31-24 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
23-0 COUNT	Counter Timer Value The TV register is used to read the current value of Watchdog timer counter.

11.1.6 Watchdog Warning Interrupt Compare Value Register (WARNINT)

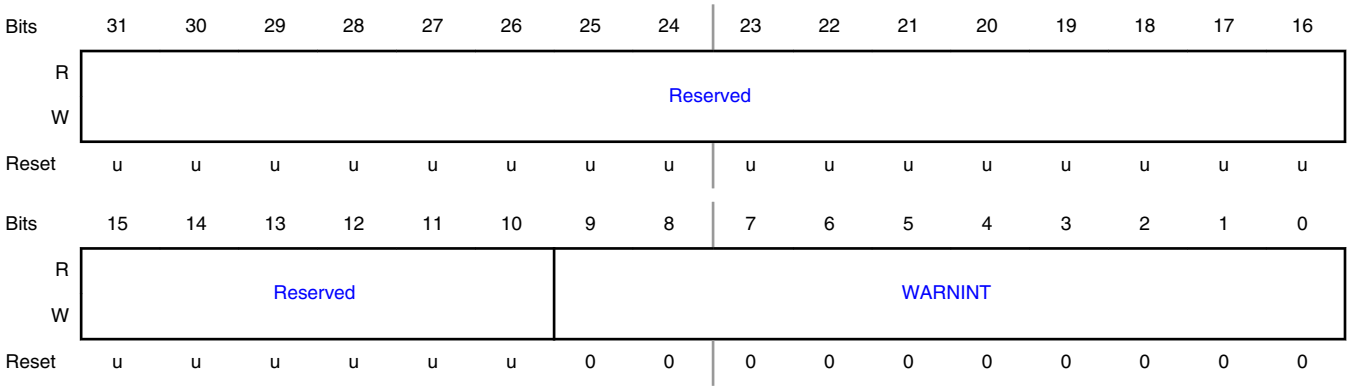
Offset

Register	Offset
WARNINT	14h

Function

The WARNINT register determines the watchdog timer counter value that will generate a watchdog interrupt. When the watchdog timer counter is no longer greater than the value defined by WARNINT, an interrupt will be generated after the subsequent WDCLK.

Diagram



Fields

Field	Function
31-10 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
9-0 WARNINT	Watchdog Warning Interrupt Compare Value WA match of the watchdog timer counter to WARNINT occurs when the bottom 10 bits of the counter have the same value as the 10 bits of WARNINT, and the remaining upper bits of the counter are all 0. This gives a maximum time of 1,023 watchdog timer counts (4,096 watchdog clocks) for the interrupt to occur prior to a watchdog event. If WARNINT is 0, the interrupt will occur at the same time as the watchdog event.

11.1.7 Watchdog Timer Window Register (WINDOW)

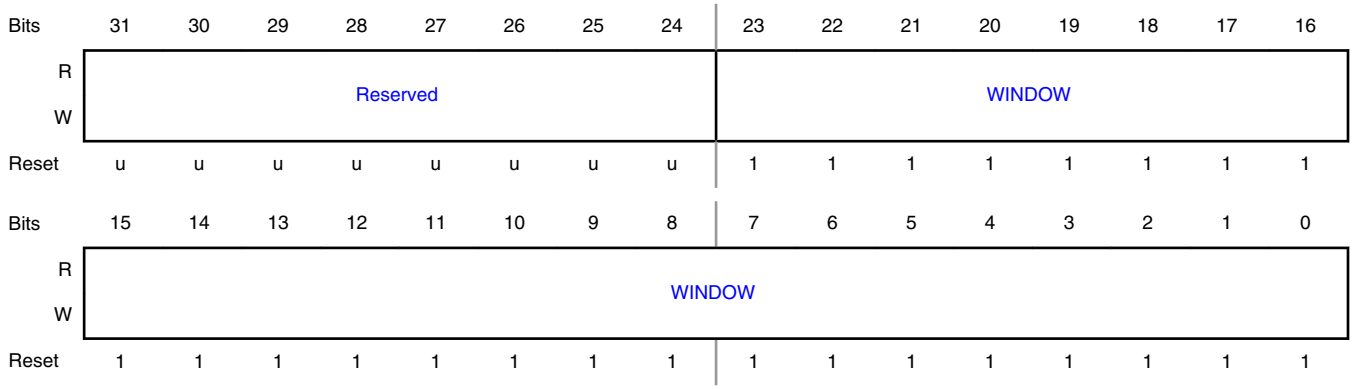
Offset

Register	Offset
WINDOW	18h

Function

This register determines the highest TV value allowed when a watchdog feed is performed.

Diagram



Fields

Field	Function
31-24 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
23-0 WINDOW	Watchdog Window Value The WINDOW register determines the highest TV value allowed when a watchdog feed is performed. If a feed sequence occurs when TV is greater than the value in WINDOW, a watchdog event will occur. WINDOW resets to the maximum possible TV value, so windowing is not in effect.

Chapter 12
Real-Time Clock (RTC)

12.1 RTC register descriptions

12.1.1 RTC memory map

RTC base address: 4000_B000h

Offset	Register	Width (In bits)	Access	Reset value
0h	RTC Control Register (CTRL)	32	RW	See description
4h	RTC 32-bit Counter Match Register (MATCH)	32	RW	FFFF_FFFFh
8h	RTC 32-bit Counter Register (COUNT)	32	RW	0000_0000h
Ch	16-bit RTC Timer Register (WAKE)	32	RW	See description

12.1.2 RTC Control Register (CTRL)

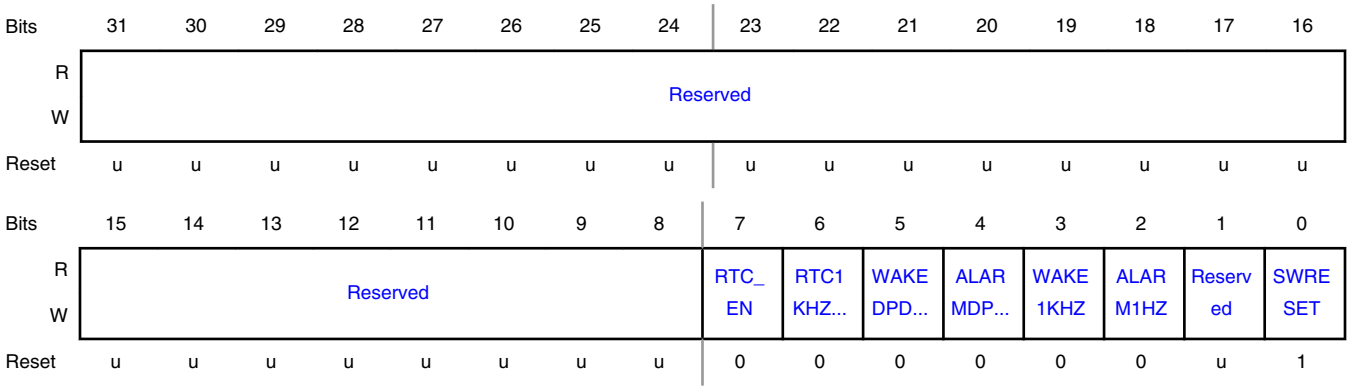
Offset

Register	Offset
CTRL	0h

Function

This register controls which clock the RTC uses (1 kHz or 1 Hz) and enables the two RTC interrupts to wake up the part from deep power-down. To wake up the part from deep-sleep mode, enable the RTC interrupts in the system control block SYSCON_STARTER0 register.

Diagram



Fields

Field	Function
31-8 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
7 RTC_EN	RTC Enable 0b - Disable. The RTC 32-bit timer and 16-bit timer clocks are shut down and the RTC operation is disabled. This bit should be 0 when writing to load a value in the RTC counter register. 1b - Enable. The 32-bit RTC clock is running and RTC operation is enabled. This bit must be set to initiate operation of the RTC. To also enable the 16-bit timer clock, set bit 6 in this register.
6 RTC1KHZ_EN	RTC 16-bit/1-kHz Timer Clock Enable This bit can be set to 0 to conserve power if the 16-bit/1-kHz timer is not used. This bit has no effect when the RTC is disabled (bit 7 of this register is 0). 0b - Disable. A match on the 16-bit/1-kHz RTC timer will not bring the part out of Deep power-down mode. 1b - Enable. The 16-bit/1-kHz RTC timer is enabled.
5 WAKEDPD_EN	RTC 16-bit Timer Wake-up enable for Power-down Modes 0b - Disable. A match on the 16-bit RTC timer will not bring the part out of power-down modes. 1b - Enable. A match on the 16-bit RTC timer will bring the part out of power-down modes.
4 ALARMDPD_EN	RTC 32-bit Timer Alarm Enable for Low Power Mode 0b - Disable. A match on the 32-bit RTC timer will not bring the part out of power-down modes. 1b - Enable. A match on the 32-bit RTC timer will bring the part out of power-down modes.
3 WAKE1KHZ	RTC 16-bit/1-kHz Timer Wake-up Flag Status 0b - The RTC 16-bit/1-kHz timer is running. Writing a 0 has no effect. 1b - The 16-bit/1-kHz timer has timed out. This flag generates an RTC wake-up interrupt request RTC-WAKE which can also wake up the part from low power modes (excluding deep power down mode). Writing a 1 clears this bit.
2 ALARM1HZ	RTC 32-bit/1-Hz Timer Alarm Flag Status 0b - No match has occurred on the 32-bit/1-Hz RTC timer. Writing a 0 has no effect. 1b - A match condition has occurred on the 32-bit/1-Hz RTC timer. This flag generates an RTC alarm interrupt request. RTC_ALARM which can also wake up the part from low power modes (excluding deep power down mode). Writing a 1 clears this bit.
1 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
0 SWRESET	Software Reset Control 0b - Not in reset. The RTC is not held in reset. This bit must be cleared prior to configuring or initiating any operation of the RTC. 1b - In reset. The RTC is held in reset. All register bits within the RTC will be forced to their reset value except the OFD bit. This bit must be cleared before writing to any register in the RTC - including writes to set any of the other bits within this register. Do not attempt to write to any bits of this register at the same time that the reset bit is being cleared.

12.1.3 RTC 32-bit Counter Match Register (MATCH)

Offset

Register	Offset
MATCH	4h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	MATVAL															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MATVAL															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fields

Field	Function
31-0 MATVAL	Match Value Contains the match value against which the 1 Hz RTC timer will be compared to generate the alarm flag RTC_ALARM and generate an alarm interrupt/wake-up if enabled.

12.1.4 RTC 32-bit Counter Register (COUNT)

Offset

Register	Offset
COUNT	8h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	VAL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	VAL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

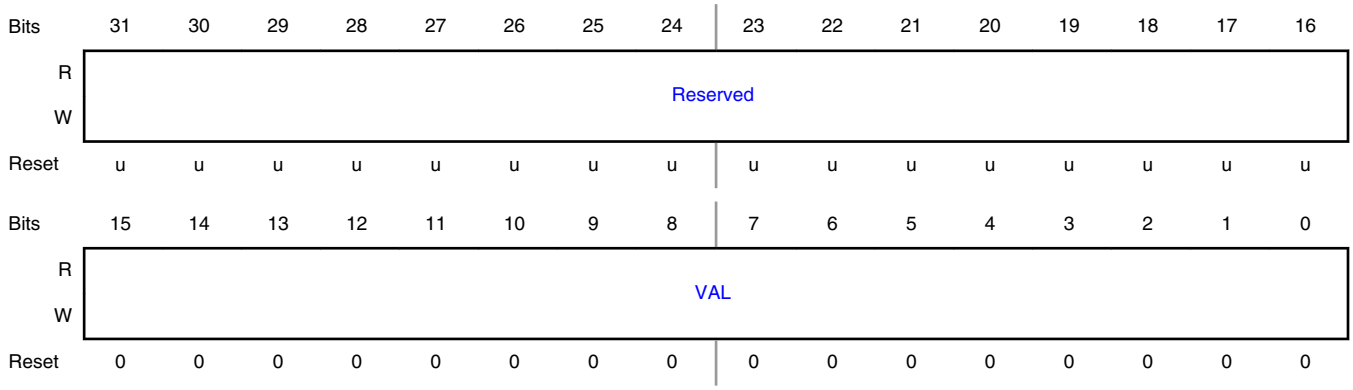
Field	Function
31-0 VAL	32-bit RTC Timer Value A read reflects the current value of the main, 32-bit, RTC timer. A write loads a new initial value into the timer. The RTC 32-bit counter will count up continuously at the 32-bit timer clock rate once the RTC Software Reset is removed (by clearing bit 0 of the CTRL register). **NOTE** No synchronization is provided to prevent a read of the counter register during a count transition. The suggested method to read a counter is to read the location twice and compare the results. If the values match, the time can be used. If they do not match, then the read should be repeated until two consecutive reads produce the same result. **NOTE** Only write to this register when the CTRL[RTC_EN] bit is 0. The counter increments one second after the CTRL[RTC_EN] bit is set.

12.1.5 16-bit RTC Timer Register (WAKE)

Offset

Register	Offset
WAKE	Ch

Diagram



Fields

Field	Function
31-16 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
15-0 VAL	Current Value of 16-bit Timer A read reflects the current value of 16-bit timer. A write pre-loads a start count value into the 16-bit timer and initializes a count-down sequence. Do not write to this register while counting is in progress.

Chapter 13
Universal Synchronous/Asynchronous Receiver/ Transmitter (USART)

13.1 USART register descriptions

13.1.1 USART memory map

USART0 base address: 4008_B000h

USART1 base address: 4008_C000h

Offset	Register	Width (In bits)	Access	Reset value
0h	USART Configuration Register (CFG)	32	RW	See description
4h	USART Control Register (CTL)	32	RW	See description
8h	USART Status Register (STAT)	32	RW	See description
Ch	USART (not FIFO) Status Interrupt Enable Read and Set Register (INTENSET)	32	RW	See description
10h	Interrupt Enable Clear Register (INTENCLR)	32	WO	See description
20h	Baud Rate Generator Register (BRG)	32	RW	See description
24h	Interrupt Status Register (INTSTAT)	32	RO	See description
28h	Asynchronous Communication Oversample Selection Register (OSR)	32	RW	See description
2Ch	Automatic Address Matching Register (ADDR)	32	RW	See description
E00h	FIFO Configuration and Enable Register (FIFOCFG)	32	RW	See description
E04h	FIFO Status Register (FIFOSTAT)	32	RW	See description
E08h	FIFO Trigger Settings for Interrupt and DMA Request Register (FIFOTRIG)	32	RW	See description
E10h	FIFO Interrupt Enable Set (Enable) and Read Register (FIFOINTENSET)	32	RW	See description

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
E14h	FIFO Interrupt Enable Clear (Disable) and Read Register (FIFOINTENCLR)	32	WO	See description
E18h	FIFO Interrupt Status Register (FIFOINTSTAT)	32	RO	See description
E20h	FIFO Write Data Register (FIFOWR)	32	WO	See description
E30h	FIFO Read Data Register (FIFORD)	32	RO	See description
E40h	FIFO Data Read Without FIFO Pop Register (FIFORDNOPOP)	32	RO	See description
FF8h	Flexcomm ID and Peripheral Function Select Register (PSELID)	32	RW	See description
FFCh	USART Module Identifier Register (ID)	32	RO	E010_2100h

13.1.2 USART Configuration Register (CFG)

Offset

Register	Offset
CFG	0h

Function

The CFG register contains communication and mode settings for aspects of the USART that would normally be configured once in an application.

NOTE

Only the CFG register can be written when the ENABLE bit = 0. CFG can be set up by software with ENABLE = 1, then the rest of the USART can be configured.

NOTE

If software needs to change configuration values, the following sequence should be used:

1. Make sure the USART is not currently sending or receiving data.
2. Disable the USART by writing a 0 to the Enable bit (0 may be written to the entire register).
3. Write the new configuration value, with the ENABLE bit set to 1.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								TXPOL	RXPOL	OEPOL	OESEL	AUTOADDR	OETA	Reserved	
Reset	u	u	u	u	u	u	u	u	0	0	0	0	0	0	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	LOOP	SYNCMST	Reserved	CLKPOL	SYNCEN	Reserved	CTSEN	LINMODE	MODE32K	STOPLEN	PARITYSEL		DATALEN		Reserved	ENABLE
Reset	0	0	u	0	0	u	0	0	0	0	0	0	0	0	u	0

Fields

Field	Function
31-24 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
23 TXPOL	Transmit Data Polarity 0b - Standard. The TX signal is sent out without change. This means that the TX reset value is 1, start bit is 0, data is not inverted, and the stop bit is 1. 1b - Inverted. The TX signal is inverted by the USART before being sent out. This means that the TX rest value is 0, start bit is 1, data is inverted, and the stop bit is 0.
22 RXPOL	Receive Data Polarity 0b - Standard. The RX signal is used as it arrives from the pin. This means that the RX rest value is 1, start bit is 0, data is not inverted, and the stop bit is 1. 1b - Inverted. The RX signal is inverted before being used by the USART. This means that the RX rest value is 0, start bit is 1, data is inverted, and the stop bit is 0.
21 OEPOL	Output Enable Polarity 0b - Low. If selected by OESEL, the output enable is active low. 1b - High. If selected by OESEL, the output enable is active high.
20 OESEL	Output Enable Selection 0b - Standard. The RTS signal is used as the standard flow control function. 1b - RS-485. The RTS signal configured to provide an output enable signal to control an RS-485 transceiver.
19 AUTOADDR	Automatic Address Matching Enable 0b - Disabled. When addressing is enabled by ADDRDET, address matching is done by software. This provides the possibility of versatile addressing (e.g. respond to more than one address). 1b - Enabled. When addressing is enabled by ADDRDET, address matching is done by hardware, using the value in the ADDR register as the address to match.

Table continues on the next page...

Table continued from the previous page...

Field	Function
18 OETA	RS-485 Operation Output Enable Turnaround Time Enable 0b - Disabled. If selected by OESEL, the Output Enable signal deasserted at the end of the last stop bit of a transmission. 1b - Enabled. If selected by OESEL, the Output Enable signal remains asserted for one character time after the end of the last stop bit of a transmission. OE will also remain asserted if another transmit begins before it is deasserted.
17-16 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
15 LOOP	Data Loopback Mode Selection This field selects data loopback mode. This provides a mechanism to perform diagnostic loopback testing for USART data. Serial data from the transmitter (Un_TXD) is connected internally to serial input of the receiver (Un_RXD). Un_TXD and Un_RTS activity will also appear on external pins if these functions are configured to appear on device pins. The receiver RTS signal is also looped back to CTS and performs flow control if enabled by CTSEN bit. 0b - Normal operation. 1b - Loopback mode.
14 SYNCMST	Synchronous Mode Master Selection 0b - Slave. When synchronous mode is enabled, the USART is a slave. 1b - Master. When synchronous mode is enabled, the USART is a master.
13 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
12 CLKPOL	Received Data Clock Polarity and Sampling Edge Selection This field selects the clock polarity and sampling edge of received data in synchronous mode. 0b - Falling edge. Un_RXD is sampled on the falling edge of SCLK. 1b - Rising edge. Un_RXD is sampled on the rising edge of SCLK.
11 SYNCEN	Synchronous or Asynchronous Operation Selection 0b - Asynchronous mode. 1b - Synchronous mode.
10 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
9 CTSEN	CTS Enable Determine whether CTS is used for flow control. CTS can be from the input pin, or from the USART own RTS output if loopback mode is enabled. 0b - No flow control. The transmitter does not receive any automatic flow control signal. 1b - Flow control enabled. The transmitter uses the CTS input (or RTS output in loopback mode) for flow control purposes.
8 LINMODE	LIN Bus Break Mode Enable 0b - Disabled. Break detection and generation are configured for normal operation. 1b - Enabled. Break detection and generation are configured for LIN bus operation.
7 MODE32K	Standard or 32 kHz Clocking Mode Selection This field selects standard or 32 kHz clocking mode. 0b - Disabled. USART uses standard clocking. 1b - Enabled. USART uses the 32 kHz clock from the RTC oscillator as the clock source to the BRG (baud rate generator), and uses a special bit clocking scheme.
6 STOPLEN	Stop Bits Number This field configures number of stop bits appended to transmitted data. Only a single stop bit is required for received data. 0b - 1 stop bit. 1b - 2 stop bits. This setting should only be used for asynchronous communication.
5-4 PARITYSEL	USART Parity Type This field selects what type of parity is used by the USART. 00b - No parity. 01b - Reserved. 10b - Even Parity. Add a bit to each character such that the number of 1s in a transmitted character is even, and the number of 1s in a received character is expected to be even. 11b - Odd parity. Add a bit to each character such that the number of 1s in a transmitted character is odd, and the number of 1s in a received character is expected to be odd.
3-2 DATALEN	USART Data Size This field selects the data size for the USART. 00b - 7-bit Data Length 01b - 8-bit Data Length 10b - 9 bit data length. The 9th bit is commonly used for addressing in multidrop mode. See the ADDRDET bit in the CTRL regsiter. 11b - Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Function
1 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
0 ENABLE	USART Enable 0b - Disabled. The USART is disabled and the internal state machine and counters are reset. While ENABLE = 0, all USART interrupts and DMA transfers are disabled. When this field is set again, CFG and most other control bits remain unchanged. When re-enabled, the USART will immediately be ready to transmit because the transmitter has been reset and is therefore available. 1b - Eanbled. The USART is enabled for operation.

13.1.3 USART Control Register (CTL)

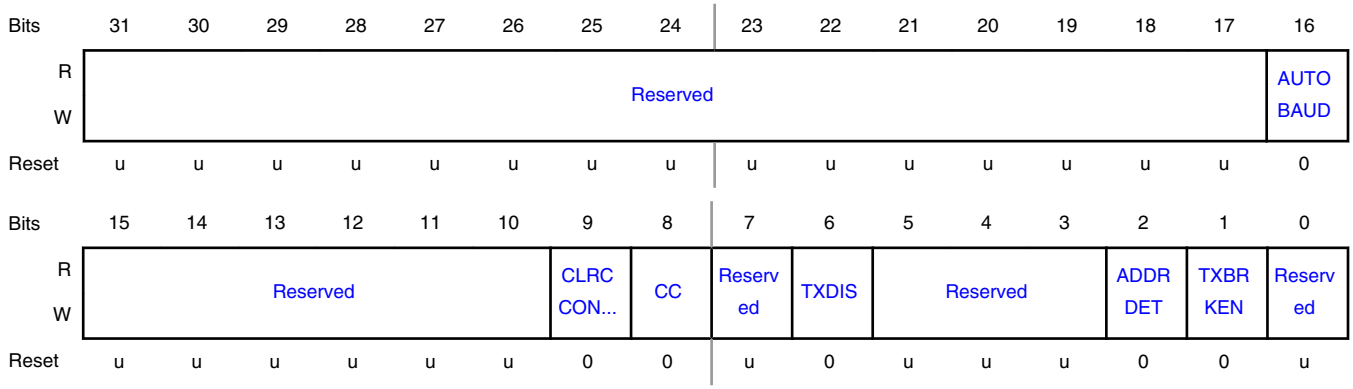
Offset

Register	Offset
CTL	4h

Function

USART control settings are more likely to be changed during operation.

Diagram



Fields

Field	Function
31-17 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
16 AUTOBAUD	Autobaud Enable 0b - Disabled. USART is in normal operating mode. 1b - Enabled. USART is in auto-baud mode. This bit should be set only when the USART receiver is idle. The first start bit of RX is measured and used to update the BRG register to match the received data rate. AUTOBAUD is cleared once this process is complete, or if there is an AERR.
15-10 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
9 CLRCCONRX	Clear Continuous Clock 0b - No effect. No effect on the CC bit. 1b - Auto-clear. The CC bit is automatically cleared when a complete character has been received. This bit is cleared at the same time.
8 CC	Continuous Clock Generation By default, SCLK is output only while data is transmitted in synchronous mode. 0b - Clock on character. In synchronous mode, SCLK cycles only when characters are being sent on Un_TXD or to complete a character that is being received. 1b - Continuous clock. SCLK runs continuously in synchronous mode, allowing characters to be received on Un_RxD independently from transmission on Un_TXD.
7 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
6 TXDIS	Transmit Disable 0b - Not disabled. USART transmitter is not disabled. 1b - Disabled. USART transmitter is disabled after any character currently being transmitted is complete. This feature can be used to facilitate software flow control.
5-3 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
2 ADDRDET	Enable Address Detect Mode 0b - Disabled. The USART presents all incoming data. 1b - Enabled. The USART receiver ignores incoming data that does not have the most significant bit of the data (typically the 9th bit) = 1. When the data MSB bit = 1, the receiver treats the incoming data normally, generating a received data interrupt. Software can then check whether the data is an address that should be handled. If it is, the ADDRDET bit is cleared by software and further incoming data is handled normally.
1 TXBRKEN	Break Enable 0b - Normal Operation. 1b - Continuous break. Continuous break is sent immediately when this bit is set, and remains until this bit is cleared. A break may be sent without danger of corrupting any currently transmitting character if the transmitter is first disabled (TXDIS field is set) and then waiting for the transmitter to be disabled (STAT[TXDISSTAT] = 1) before writing 1 to TXBRKEN.
0 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

13.1.4 USART Status Register (STAT)

Offset

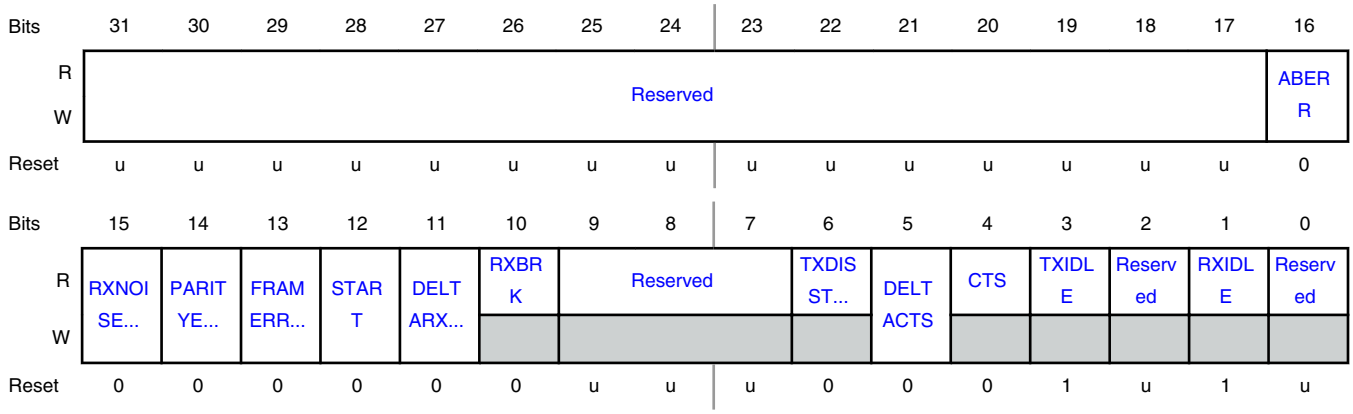
Register	Offset
STAT	8h

Function

The STAT register primarily provides a set of USART status flags (not including FIFO status) for software to read. Flags other than read-only flags may be cleared by writing ones to corresponding bits of STAT. Interrupt status flags that are read only and cannot be cleared by software, can be masked using the INTENCLR register.

The error flags for received noise, parity error, and framing error are set immediately upon detection and remain set until cleared by software action in STAT.

Diagram



Fields

Field	Function
31-17 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
16 ABERR	Auto Baud Error When performing an auto baud measurement on a start bit, an auto baud error can occur if the BRG counts to its limit before the end of the start bit. This flag will be set if this condition occurs. Essentially it is an auto baud time-out error flag.
15 RXNOISEINT	Received Noise Interrupt Flag Three samples of received data are taken in order to determine the value of each received data bit, except in synchronous mode. This acts as a noise filter if one sample disagrees. This flag is set when a received data bit contains one disagreeing sample. This could indicate line noise, a baud rate or character format mismatch, or loss of synchronization during data reception.
14 PARITYERRINT	Parity Error Interrupt Flag This flag is set when a parity error is detected in a received character.
13 FRAMERRINT	Framing Error Interrupt Flag This flag is set when a character is received with a missing stop bit at the expected location. This could be an indication of a baud rate or configuration mismatch with the transmitting source.
12 START	Receiver Input Start Detection This bit is set when a start is detected on the receiver input. Its purpose is primarily to allow wake-up from Deep sleep or Power-down mode immediately when a start is detected. Cleared by software.
11 DELTARXBRK	Receiver Break Detection State Change This bit is set when a change in the state of receiver break detection occurs. Cleared by software.

Table continues on the next page...

Table continued from the previous page...

Field	Function
10 RXBRK	Received Break This bit reflects the current state of the receiver break detection logic. It is set when the Un_RXD pin remains low for 16-bit times. Note that FRAMERRINT bit will also be set when this condition occurs because the stop bit(s) for the character would be missing. RXBRK is cleared when the Un_RXD pin goes high.
9-7 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
6 TXDISSTAT	Transmitter Disabled Status Flag 0b - Reserved 1b - Indicates that the USART transmitter is fully idle after being disabled via the TXDIS bit (CTL[TXDIS] = 1).
5 DELTACTS	CTS Flag State Change This bit is set when a change in the state is detected for the CTS flag above. This bit is cleared by software.
4 CTS	CTS Signal State This bit reflects the current state of the CTS signal, regardless of the setting of the CFG[CTSEN] bit. This will be the value of the CTS input pin unless loopback mode is enabled. Hence, reset value is not applicable.
3 TXIDLE	Transmitter Idle 0b - Indicate that the transmitter is currently in the process of sending data 1b - Indicate that the transmitter is not currently in the process of sending data.
2 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
1 RXIDLE	Receiver Idle 0b - Indicate that the receiver is currently in the process of receiving data. 1b - Indicate that the receiver is not currently in the process of receiving data.
0 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

13.1.5 USART (not FIFO) Status Interrupt Enable Read and Set Register (INTENSET)

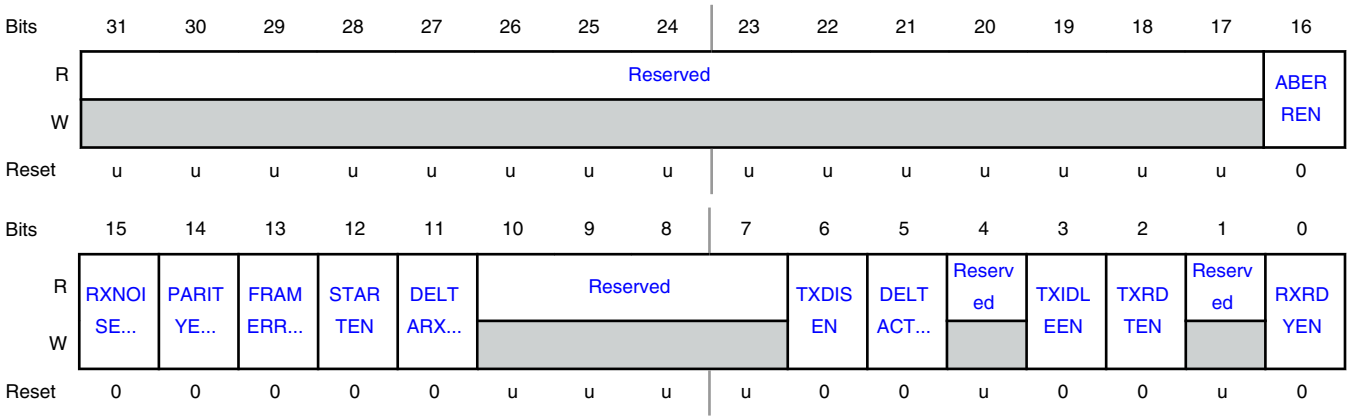
Offset

Register	Offset
INTENSET	Ch

Function

The INTENSET register is used to enable various USART interrupt sources (not including FIFO interrupts). Enable bits in INTENSET are mapped in locations that correspond to the flags in the STAT register. Interrupt enables may also be read back from this register. Writing ones to implemented bits in this register causes those bits to be set. The INTENCLR register is used to clear bits in this register.

Diagram



Fields

Field	Function
31-17 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
16 ABERREN	Auto Baud Error Interrupt Enable When 1, it enables an interrupt when an auto baud error occurs.
15 RXNOISEEN	Noise Detect Interrupt Enable When 1, it enables an interrupt when noise is detected.
14 PARITYERREN	Parity Error Detect Interrupt Enable When 1, it enables an interrupt when a parity error has been detected.

Table continues on the next page...

Table continued from the previous page...

Field	Function
13 FRAMERREN	Frame Error Detect Interrupt Enable When 1, it enables an interrupt when a framing error has been detected.
12 STARTEN	Received Start Bit Detect Interrupt Enalbe When 1, it enables an interrupt when a received start bit has been detected.
11 DELTARXBRKEN	Break Condition Receive Detection State Change Interrupt Enable When 1, it enables an interrupt when a change of state has occurred in the detection of a received break condition (break condition asserted or deasserted).
10-7 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
6 TXDISEN	Transmitter Disabled Interrupt Enable When 1, enables an interrupt when the transmitter is fully disabled as indicated by the STAT[TXDISSTAT] flag. See description of the STAT[TXDISSTAT] for details.
5 DELTACTSEN	CTS Change Interrupt Enable When 1, it enables an interrupt when the CTS flag changes state.
4 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
3 TXIDLEEN	Transmitter Idle Interrupt Enable When 1, it enables an interrupt when the transmitter becomes idle (STAT[TXIDLE] = 1).
2 TXRDTEN	TX Ready Interrupt Enable When 1, it enables an interrupt when TX becomes ready
1 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
0 RXRDYEN	RX Ready Interrupt Enable When 1, it enables an interrupt when RX becomes ready

13.1.6 Interrupt Enable Clear Register (INTENCLR)

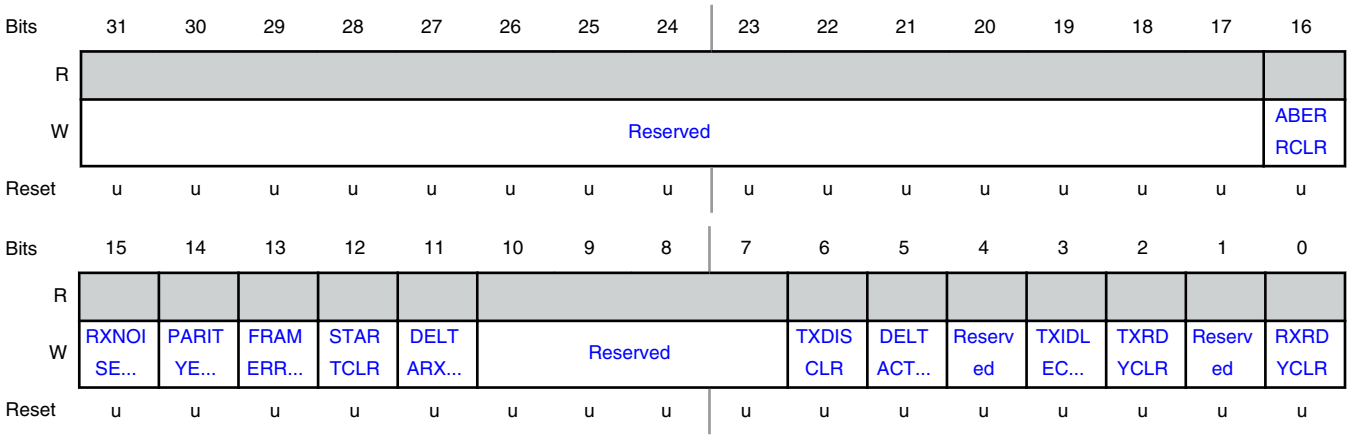
Offset

Register	Offset
INTENCLR	10h

Function

This register allows clearing any combination of bits in the INTENSET register. Writing a 1 to any implemented bit position causes the corresponding bit to be cleared.

Diagram



Fields

Field	Function
31-17 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
16 ABERRCLR	ABERR Clear Writing 1 clears the ABERR bit in the INTENSET register.
15 RXNOISECLR	RXNOISE Clear Writing 1 clears the RXNOISE bit in the INTENSET register.
14 PARITYERRCLR	PARITYERR Clear Writing 1 clears the PARITYERR bit in the INTENSET register.
13 FRAMERRCLR	FRAMERR Clear Writing 1 clears the FRAMERR bit in the INTENSET register.

Table continues on the next page...

Table continued from the previous page...

Field	Function
12 STARTCLR	START Clear Writing 1 clears the START bit in the INTENSET register.
11 DELTARXBRKCLR	DELTARXBRK Clear Writing 1 clears the DELTARXBRK bit in the INTENSET register.
10-7 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
6 TXDISCLR	TXDIS Clear Writing 1 clears the TXDIS bit in the INTENSET register.
5 DELTACTSCLR	DELTACTS Clear Writing 1 clears the DELTACTS bit in the INTENSET register.
4 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
3 TXIDLECLR	TXIDLE Clear Writing 1 clears the TXIDLE bit in the INTENSET register.
2 TXRDYCLR	TXRDY Clear Writing 1 clears the TXRDY bit in the INTENSET register.
1 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
0 RXRDYCLR	RXRDY Clear Writing 1 clears INTENSET[RXRDYEN] bit.

13.1.7 Baud Rate Generator Register (BRG)

Offset

Register	Offset
BRG	20h

Function

The Baud Rate Generator is a simple 16-bit integer divider controlled by the BRG register. The BRG register contains the value used to divide the USARTCLK in order to produce the clock used for USART internal operations.

A 16-bit value allows producing standard baud rates from 300 baud and lower at the highest frequency of the device, up to 921,600 baud from a base clock as low as 14.7456 MHz.

Typically, the baud rate clock is 16 times the actual baud rate. This overclocking allows for centering the data sampling time within a bit cell, and for noise reduction and detection by taking three samples of incoming data. Note that in 32 kHz mode, the baud rate generator is still used and must be set to 0 if 9600 baud is required.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BRGVAL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-16 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
15-0 BRGVAL	USART Input Clock Divider This value is used to divide the USART input clock to determine the baud rate, based on the input clock from the FRG. 0: FCLK is used directly by the USART function. 1: FCLK is divided by 2 before use by the USART function. 2: FCLK is divided by 3 before use by the USART function. ... 0xFFFF = FCLK is divided by 65,536 before use by the USART function.

13.1.8 Interrupt Status Register (INTSTAT)

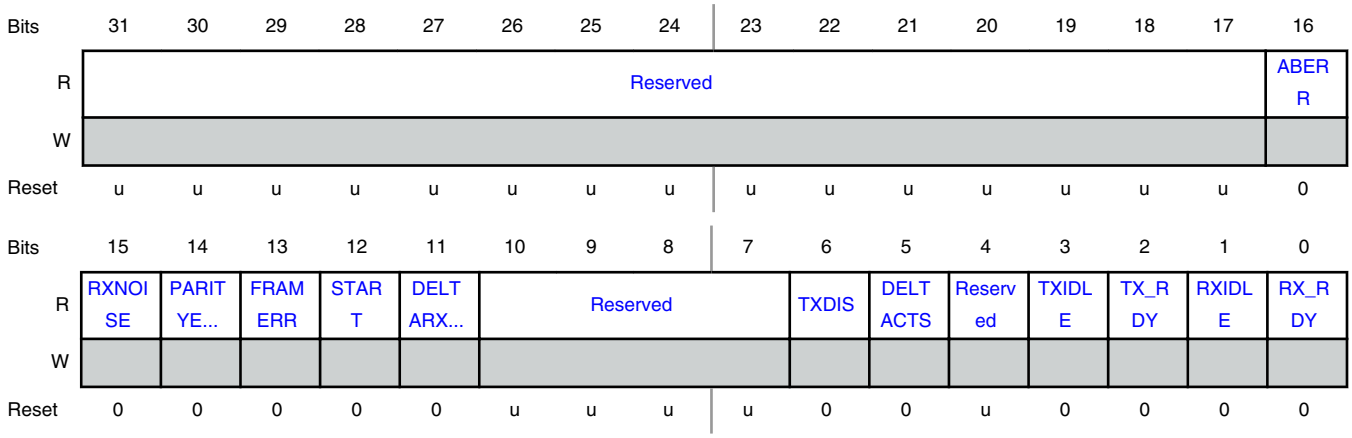
Offset

Register	Offset
INTSTAT	24h

Function

The read-only INTSTAT register provides a view of those interrupt flags that are currently enabled. This can simplify software handling of interrupts.

Diagram



Fields

Field	Function
31-17 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
16 ABERR	Auto baud Error Interrupt Flag
15 RXNOISE	Received Noise Interrupt Flag
14 PARITYERR	Parity Error Interrupt Flag
13 FRAMERR	Framing Error Interrupt Flag
12 START	Receiver Input Start Detect This bit is set when a start is detected on the receiver input.
11 DELTARXBRK	Receiver Break Detection State Change This bit is set when a change in the state of receiver break detection occurs.
10-7 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
6 TXDIS	Transmitter Disabled Interrupt Flag

Table continues on the next page...

Table continued from the previous page...

Field	Function
5 DELTACTS	CTS Input State Change Detect This bit is set when a change in the state of the CTS input is detected.
4 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
3 TXIDLE	Transmitter Idle Status
2 TX_RDY	Transmitter Ready Status
1 RXIDLE	Receiver Idle Status
0 RX_RDY	Receiver Ready Status

13.1.9 Asynchronous Communication Oversample Selection Register (OSR)

Offset

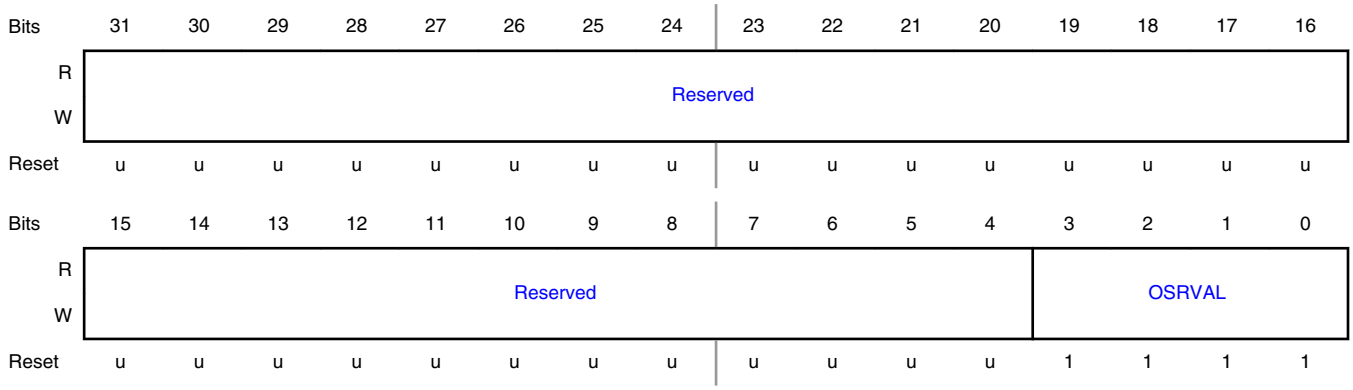
Register	Offset
OSR	28h

Function

The OSR register allows selection of oversampling in asynchronous modes. The oversample value is the number of BRG clocks used to receive one data bit. The default is industry standard 16x oversampling.

Changing the oversampling can sometimes allow better matching of baud rates in cases where the function clock rate is not a multiple of 16 times the expected maximum baud rate. For all modes where the OSR setting is used, the USART receiver takes three consecutive samples of input data in the approximate middle of the bit time. Smaller values of OSR can make the sampling position within a data bit less accurate and may potentially cause more noise errors or incorrect data.

Diagram



Fields

Field	Function
31-4 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
3-0 OSRVAL	Oversample Selection Value 0000b to 0011b: not supported. 0100b: 5 function clocks are used to transmit and receive each data bit. 0101b: 6 function clocks are used to transmit and receive each data bit. ... 0x1111b: 16 function clocks are used to transmit and receive each data bit.

13.1.10 Automatic Address Matching Register (ADDR)

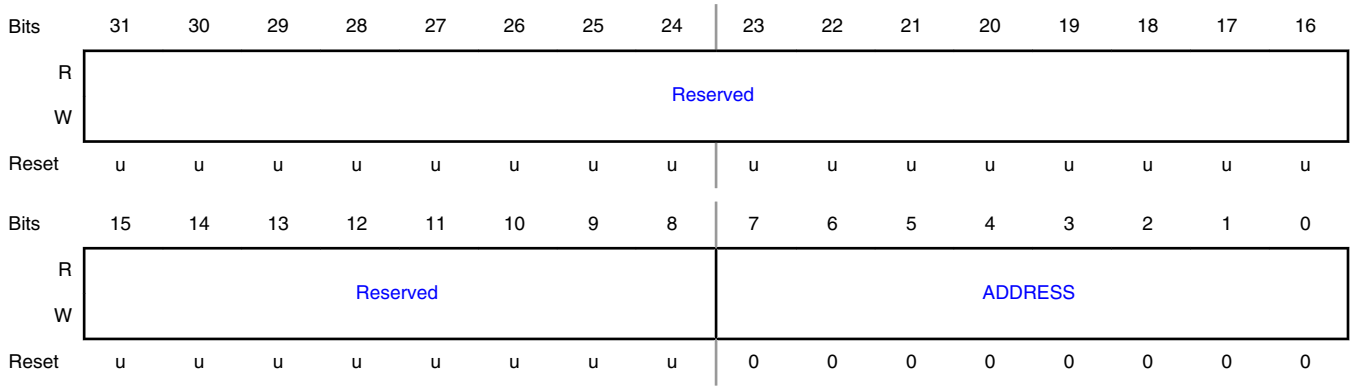
Offset

Register	Offset
ADDR	2Ch

Function

The ADDR register holds the address for hardware address matching in address detect mode with automatic address matching enabled.

Diagram



Fields

Field	Function
31-8 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
7-0 ADDRESS	8-bit Address Used with Automatic Address Matching. Used when address detection is enabled (ADDRDET in CTL = 1) and automatic address matching is enabled (AUTOADDR in CFG = 1).

13.1.11 FIFO Configuration and Enable Register (FIFOCFG)

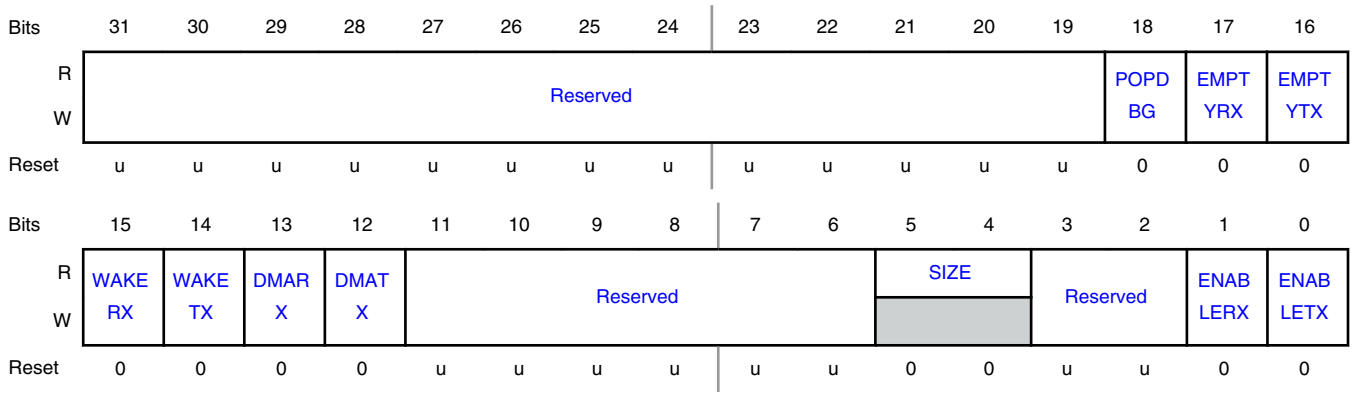
Offset

Register	Offset
FIFOCFG	E00h

Function

This register configures FIFO usage. The configuration of PSELID must be performed prior to configuring the FIFO.

Diagram



Fields

Field	Function
31-19 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
18 POPDBG	Pop FIFO for Debug Reads
17 EMPTYRX	Empty Command for Receive FIFO When a 1 is written to this bit, the RX FIFO is emptied.
16 EMPTYTX	Empty Command for Transmit FIFO When a 1 is written to this bit, the TX FIFO is emptied.
15 WAKERX	Wakeup for Rceive FIFO Level This allows the device to be woken from reduced power modes (up to power-down, as long as the peripheral function works in that power mode) without enabling the TXLVL interrupt. Only DMA wakes up, processes data, and goes back to sleep. The CPU will remain stopped until woken by another cause, such as DMA completion. 0b - Only enabled interrupts will wake up the device form reduced power modes. 1b - A device wake-up for DMA will occur if the receive FIFO level reaches the value specified by RXLVL in FIFOTRIG, even when the RXLVL interrupt is not enabled.
14 WAKETX	Wakeup for Transmit FIFO Level This allows the device to be woken from reduced power modes (up to power-down, as long as the peripheral function works in that power mode) without enabling the TXLVL interrupt. Only DMA wakes up, processes data, and goes back to sleep. The CPU will remain stopped until woken by another cause, such as DMA completion. 0b - Only enabled interrupts will wake up the device form reduced power modes. 1b - A device wake-up for DMA will occur if the transmit FIFO level reaches the value specified by TXLVL in FIFOTRIG, even when the TXLVL interrupt is not enabled.
13 DMARX	DMA Configuration for Receive 0b - DMA is not used for the receive function. 1b - Generate a DMA request for the receive function if the FIFO is not empty. Generally, data interrupts would be disabled if DMA is enabled.
12 DMATX	DMA Configuration for Transmit 0b - DMA is not used for the transmit function. 1b - Generate a DMA request for the transmit function if the FIFO is not full. Generally, data interrupts would be disabled if DMA is enabled

Table continues on the next page...

Table continued from the previous page...

Field	Function
11-6 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
5-4 SIZE	FIFO Size Configuration 01b, 10b, 11b are not applicable. 00b - FIFO is configured as 4 entries of 8 bits.
3-2 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
1 ENABLERX	Enable Receive FIFO 0b - The receive FIFO is not enabled. 1b - The receive FIFO is enabled. This is automatically enabled when PSELID.PERSEL is set to 1 to configure the USART functionality.
0 ENABLETX	Enable Transmit FIFO 0b - The transmit FIFO is not enabled. 1b - The transmit FIFO is enabled. This is automatically enabled when PSELID.PERSEL is set to 1 to configure the USART functionality.

13.1.12 FIFO Status Register (FIFOSTAT)

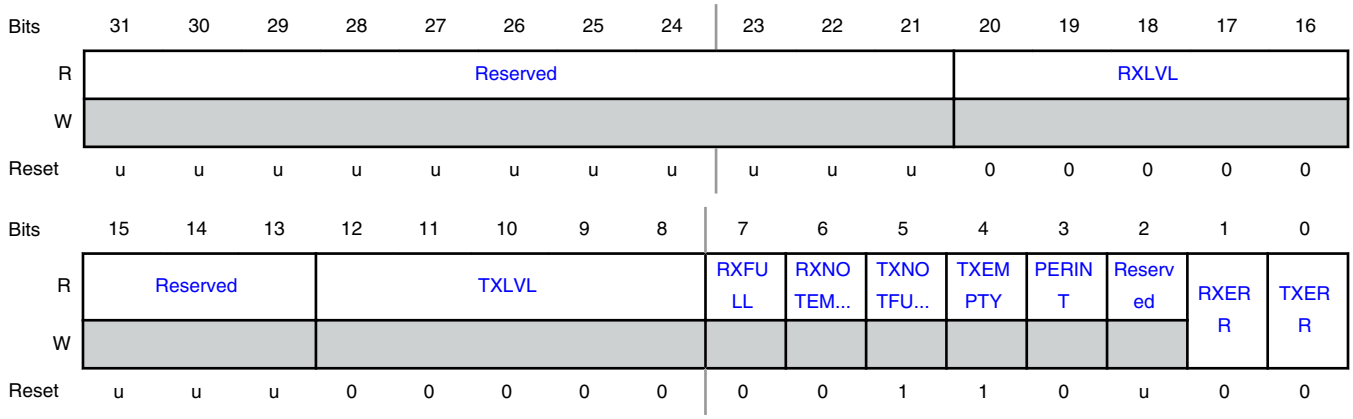
Offset

Register	Offset
FIFOSTAT	E04h

Function

This register provides status information for the FIFO and also indicates an interrupt from the peripheral function.

Diagram



Fields

Field	Function
31-21 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
20-16 RXLVL	Receive FIFO Current Level A 0 means the RX FIFO is currently empty, and the RXFULL and RXNOTEMPTY flags will be 0. Other values tell how much data is actually in the RX FIFO at the point where the read occurs. If the RX FIFO is full, the RXFULL and RXNOTEMPTY flags will be 1.
15-13 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
12-8 TXLVL	Transmit FIFO Current Level A 0 means the TX FIFO is currently empty, and the TXEMPTY and TXNOTFULL flags will be 1. Other values tell how much data is actually in the TX FIFO at the point where the read occurs. If the TX FIFO is full, the TXEMPTY and TXNOTFULL flags will be 0.
7 RXFULL	Receive FIFO Full When 1, the receive FIFO is full. Data needs to be read out to prevent the peripheral from causing an overflow.
6 RXNOTEMPTY	Receive FIFO not Empty When 1, the receive FIFO is not empty, so data can be read. When 0, the receive FIFO is empty.
5 TXNOTFULL	Transmit FIFO not Full When 1, the transmit FIFO is not full, so more data can be written. When 0, the transmit FIFO is full and another write would cause it to overflow.

Table continues on the next page...

Table continued from the previous page...

Field	Function
4 TXEMPTY	Transmit FIFO Empty When 1, the transmit FIFO is empty. The peripheral may still be processing the last piece of data.
3 PERINT	Peripheral Interrupt When 1, this indicates that the peripheral function has asserted an interrupt. The details can be found by reading the peripheral s STAT register.
2 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
1 RXERR	RX FIFO Error RX FIFO error. Will be set if a receive FIFO overflow occurs, caused by software or DMA not emptying the FIFO fast enough. Cleared by writing a 1 to this bit.
0 TXERR	TX FIFO Error Will be set if a transmit FIFO error occurs. This could be an overflow caused by pushing data into a full FIFO, or by an underflow if the FIFO is empty when data is needed. Cleared by writing a 1 to this bit.

13.1.13 FIFO Trigger Settings for Interrupt and DMA Request Register (FIFOTRIG)

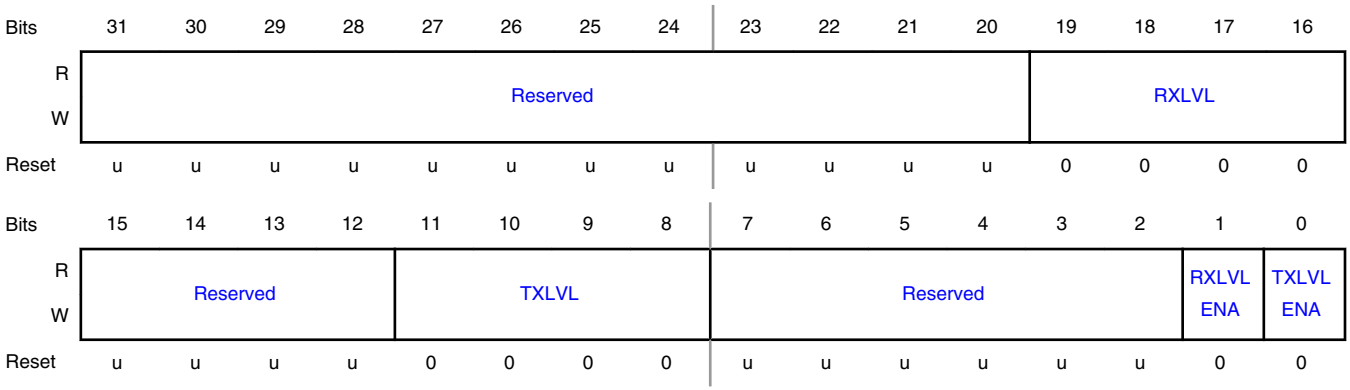
Offset

Register	Offset
FIFOTRIG	E08h

Function

This register allows selecting when FIFO-level related interrupts occur.

Diagram



Fields

Field	Function
31-20 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
19-16 RXLVL	Receive FIFO Level Trigger Point The RX FIFO level is checked when a new piece of data is received. This field is used only when RXLVLENA = 1. 0000b - Trigger when the RX FIFO has received one entry (is no longer empty). 0001b - Trigger when the RX FIFO has received two entries. 0010b - Trigger when the RX FIFO has received three entries. 0011b - Trigger when the RX FIFO has received four entries (has become full). others - Reserved
15-12 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
11-8 TXLVL	Transmit FIFO level trigger point. This field is used only when TXLVLENA = 1. 0000b - Trigger when the TX FIFO becomes empty. 0001b - Trigger when the TX FIFO level decreases to one entry. 0010b - Trigger when the TX FIFO level decreases to two entries. 0011b - Trigger when the TX FIFO level decreases to three entries (is no longer full) others - Reserved
7-2 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
1 RXLVLENA	Receive FIFO Level Trigger Enable This trigger will become an interrupt if enabled in FIFOINTENSET, or a DMA trigger if DMARX in FIFOCFG is set. 0b - Receive FIFO level does not generate a FIFO level trigger. 1b - An interrupt will be generated if the receive FIFO level reaches the value specified by the RXLVL field in this register.

Table continues on the next page...

Table continued from the previous page...

Field	Function
0 TXLVLENA	Transmit FIFO level Trigger Enable This trigger will become an interrupt if enabled in FIFOINTENSET, or a DMA trigger if DMATX in FIFOCFG is set. 0b - Transmit FIFO level does not generate a FIFO level trigger. 1b - An interrupt will be generated if the transmit FIFO level reaches the value specified by the TXLVL field in this register.

13.1.14 FIFO Interrupt Enable Set (Enable) and Read Register (FIFOINTENSET)

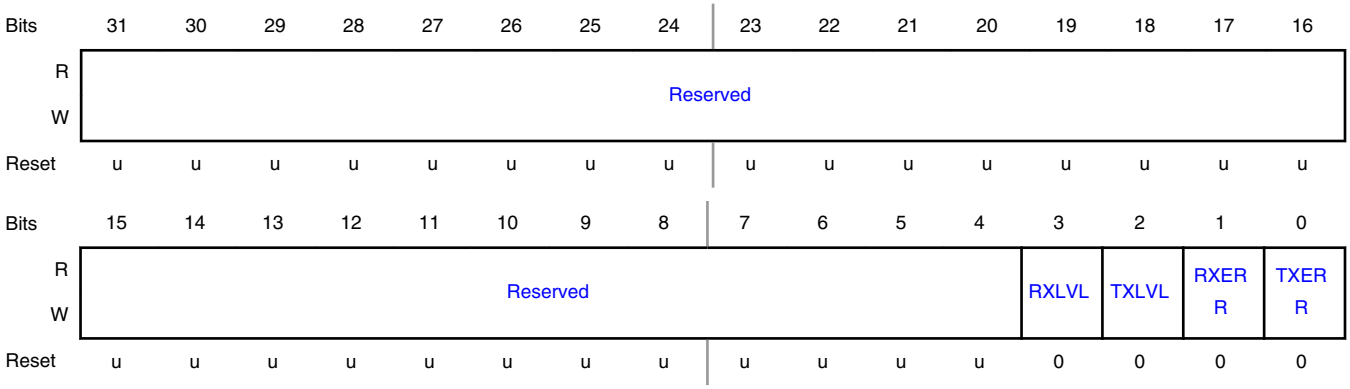
Offset

Register	Offset
FIFOINTENSET	E10h

Function

This register is used to enable various interrupt sources. The complete set of interrupt enables may be read from this register. Writing ones to implemented bits in this register causes those bits to be set. The FIFOINTENCLR register is used to clear bits in this register.

Diagram



Fields

Field	Function
31-4 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
3 RXLVL	Receive FIFO Reaches Level Interrupt This field determines whether an interrupt occurs when a the receive FIFO reaches the level specified by the TXLVL field in the FIFOTRIG register. 0b - No interrupt will be generated based on the RX FIFO level. 1b - If RXLVLENA in the FIFOTRIG register = 1, an interrupt will be generated when the when the RX FIFO level increases to the level specified by RXLVL in the FIFOTRIG register.
2 TXLVL	Transmit FIFO Reaches Level Interrupt This field determines whether an interrupt occurs when a the transmit FIFO reaches the level specified by the TXLVL field in the FIFOTRIG register. 0b - No interrupt will be generated based on the TX FIFO level. 1b - TXLVLENA in the FIFOTRIG register = 1, an interrupt will be generated when the TX FIFO level decreases to the level specified by TXLVL in the FIFOTRIG register.
1 RXERR	Receive Error Interrupt This field determines whether an interrupt occurs when a receive error occurs, based on the RXERR flag in the FIFOSTAT register. 0b - No interrupt will be generated for a receive error. 1b - An interrupt will be generated when a receive error occurs.
0 TXERR	Transmit Error Interrupt This field determines whether an interrupt occurs when a transmit error occurs, based on the TXERR flag in the FIFOSTAT register. 0b - No interrupt will be generated for a transmit error. 1b - An interrupt will be generated when a transmit error occurs.

13.1.15 FIFO Interrupt Enable Clear (Disable) and Read Register (FIFOINTENCLR)

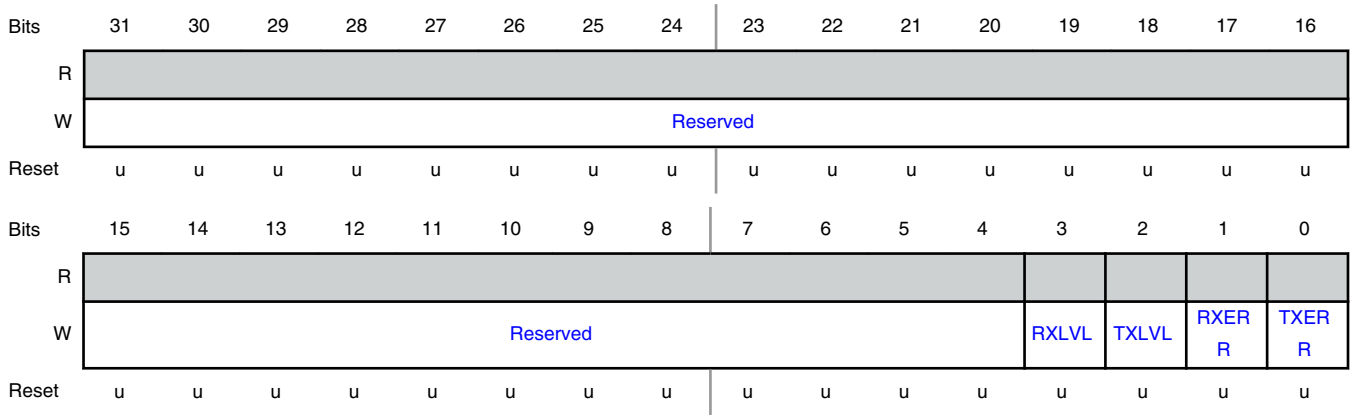
Offset

Register	Offset
FIFOINTENCLR	E14h

Function

The FIFOINTENCLR register is used to clear interrupt enable bits in FIFOINTENSET. The complete set of interrupt enables may also be read from this register as well as FIFOINTENSET.

Diagram



Fields

Field	Function
31-4 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
3 RXLVL	RXLVL Clear Writing 1 clears the RXLVL bit in the FIFOINTENSET register
2 TXLVL	TXLVL Clear Writing 1 clears the TXLVL bit in the FIFOINTENSET register
1 RXERR	RXERR Clear Writing 1 clears the RXERR bit in the FIFOINTENSET register
0 TXERR	TXERR Clear Writing 1 clears the TXERR bit in the FIFOINTENSET register

13.1.16 FIFO Interrupt Status Register (FIFOINTSTAT)

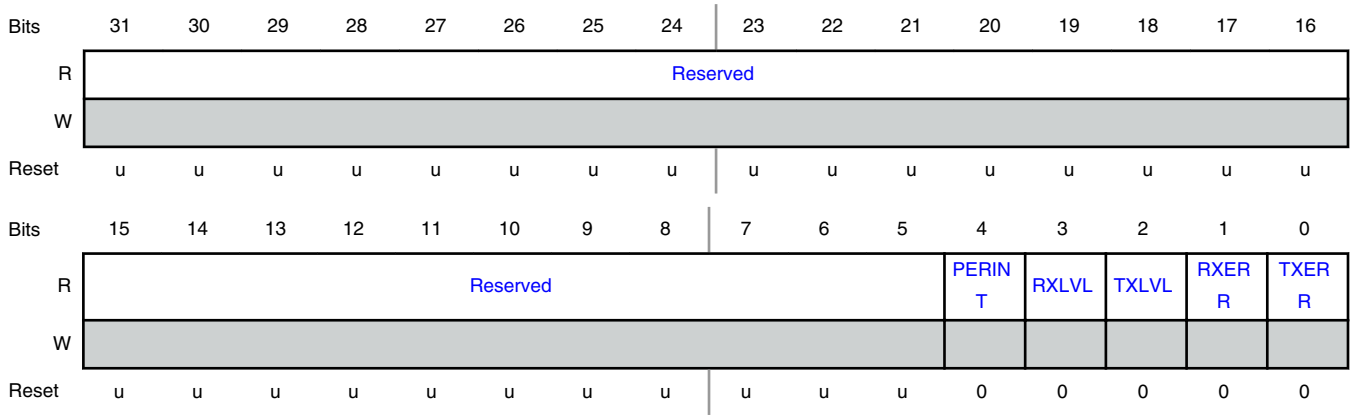
Offset

Register	Offset
FIFOINTSTAT	E18h

Function

The read-only FIFOINTSTAT register provides a view of those interrupt flags that are both pending and currently enabled. This can simplify software handling of interrupts.

Diagram



Fields

Field	Function
31-5 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
4 PERINT	Peripheral Interrupt
3 RXLVL	Receive FIFO Level Interrupt.
2 TXLVL	Transmit FIFO Level Interrupt
1 RXERR	RX FIFO Error
0 TXERR	TX FIFO Error

13.1.17 FIFO Write Data Register (FIFOWR)

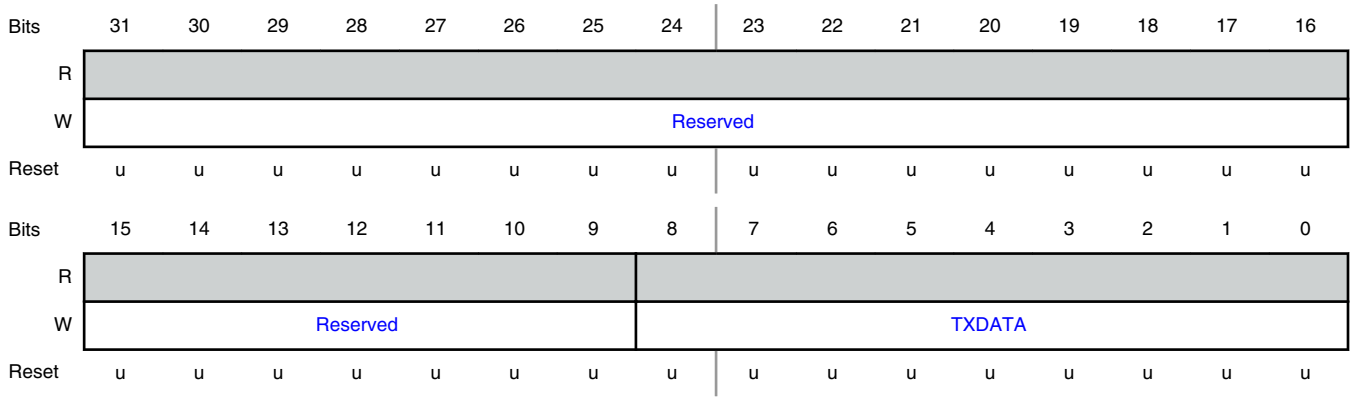
Offset

Register	Offset
FIFOWR	E20h

Function

The FIFOWR register is used to write values to be transmitted to the FIFO.

Diagram



Fields

Field	Function
31-9 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
8-0 TXDATA	Transmit Data to FIFO The number of bits used depends on the DATALEN.

13.1.18 FIFO Read Data Register (FIFORD)

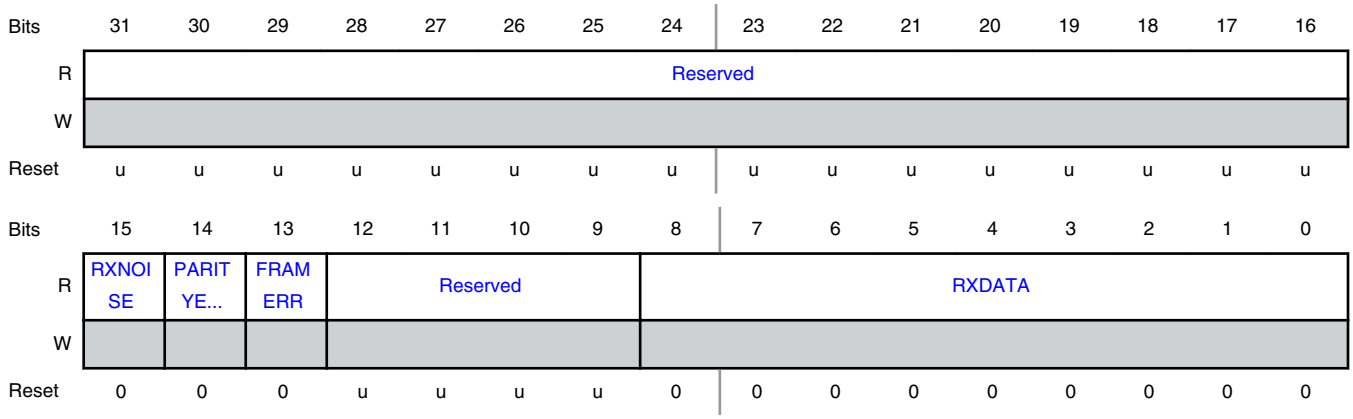
Offset

Register	Offset
FIFORD	E30h

Function

The FIFORD register is used to read values that have been received by the FIFO.

Diagram



Fields

Field	Function
31-16 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
15 RXNOISE	Received Noise Flag
14 PARITYERR	Parity Error Status Flag This bit reflects the status for the data it is read along with from the FIFO. This bit will be set when a parity error is detected in a received character.
13 FRAMERR	Framing Error Status Flag This bit reflects the status for the data it is read along with from the FIFO, and indicates that the character was received with a missing stop bit at the expected location. This could be an indication of a baud rate or configuration mismatch with the transmitting source.
12-9 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
8-0 RXDATA	Received Data from FIFO The number of bits used depends on the DATALEN and PARITYSEL settings.

13.1.19 FIFO Data Read Without FIFO Pop Register (FIFORDNOPOP)

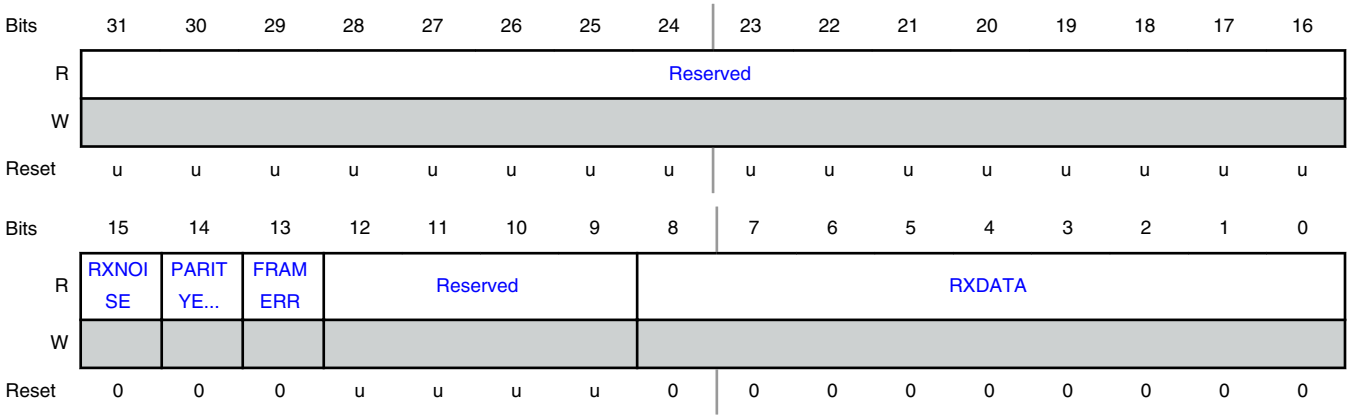
Offset

Register	Offset
FIFORDNOPOP	E40h

Function

This register acts in exactly the same way as FIFORD, except that it supplies data from the top of the FIFO without popping the FIFO (i.e. leaving the FIFO state unchanged). This could be used to allow system software to observe incoming data without interfering with the peripheral driver.

Diagram



Fields

Field	Function
31-16 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
15 RXNOISE	Received Noise Flag
14 PARITYERR	Parity Error Status Flag This bit reflects the status for the data it is read along with from the FIFO. This bit will be set when a parity error is detected in a received character.

Table continues on the next page...

Table continued from the previous page...

Field	Function
13 FRAMERR	Framing Error Status Flag This bit reflects the status for the data it is read along with from the FIFO, and indicates that the character was received with a missing stop bit at the expected location. This could be an indication of a baud rate or configuration mismatch with the transmitting source.
12-9 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
8-0 RXDATA	Received Data from FIFO The number of bits used depends on the DATALEN and PARITYSEL settings.

13.1.20 Flexcomm ID and Peripheral Function Select Register (PSELID)

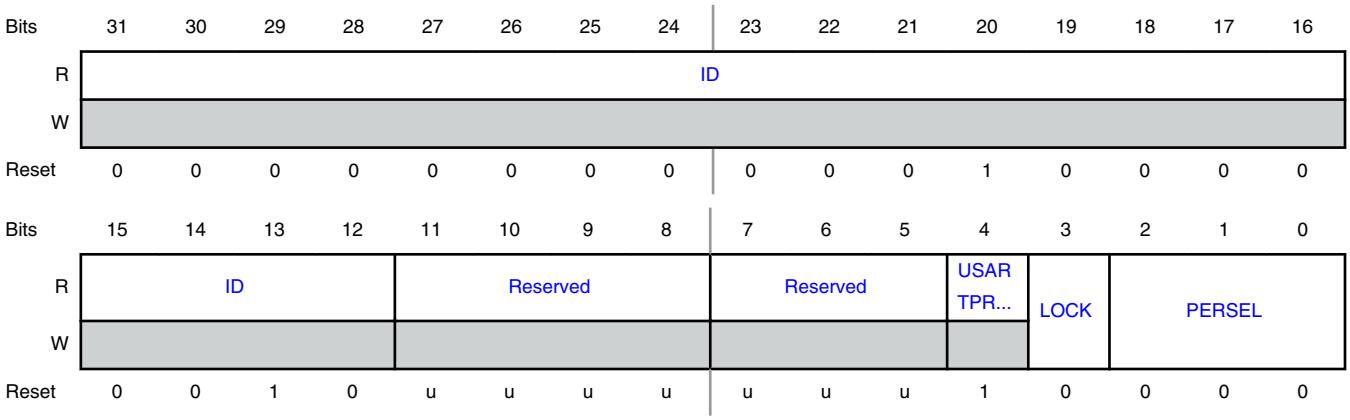
Offset

Register	Offset
PSELID	FF8h

Function

This register is used to enable the USART FIFO operation.

Diagram



Fields

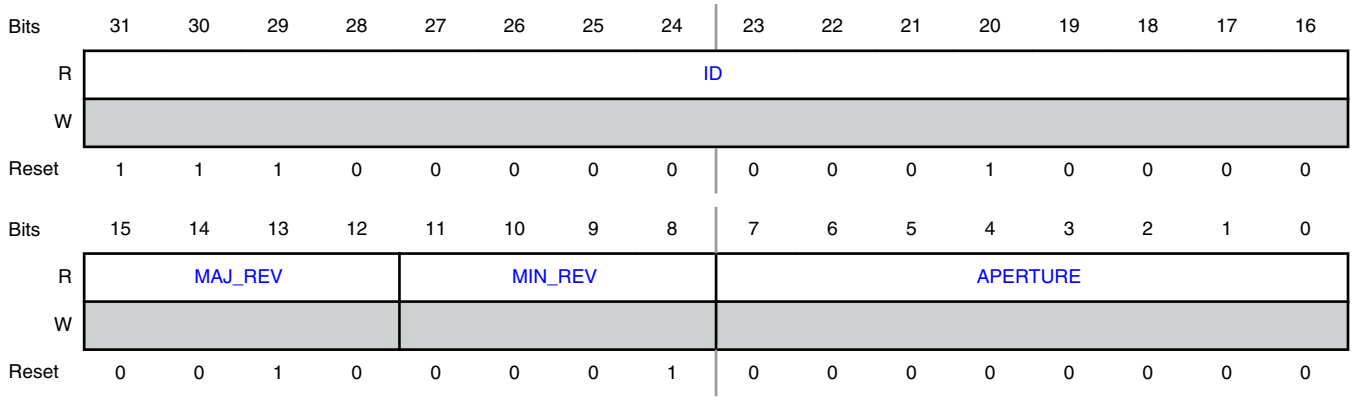
Field	Function
31-12 ID	Flexcomm ID
11-8 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
7-5 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
4 USARTPRESENT	USART Present Indicator 0b - This Flexcomm does not include the USART function. 1b - This Flexcomm includes the USART function.
3 LOCK	Lock Selected Peripheral This field is writable by software. 0b - Peripheral select can be changed by software. 1b - Peripheral select is locked and cannot be changed until this Flexcomm or the entire device is reset.
2-0 PERSEL	Peripheral Select This field is writable by software.010b-111b are reserved. 000b - No peripheral selected. 001b - USART function selected.

13.1.21 USART Module Identifier Register (ID)

Offset

Register	Offset
ID	FFCh

Diagram



Fields

Field	Function
31-16 ID	Identifier This is the unique identifier of the module
15-12 MAJ_REV	Major Revision Major revision implies software modifications
11-8 MIN_REV	Minor Revision Minor revision with no software consequences
7-0 APERTURE	Aperture Aperture number minus 1 of consecutive packets 4 Kbytes reserved for this IP

Chapter 14
Serial Peripheral Interfaces (SPI)

14.1 SPI register descriptions

14.1.1 SPI memory map

SPI0 base address: 4008_D000h

SPI1 base address: 4008_E000h

Offset	Register	Width (In bits)	Access	Reset value
400h	SPI Configuration Register (CFG)	32	RW	See description
404h	SPI Delay Register (DLY)	32	RW	See description
408h	SPI Status Register (STAT)	32	RW	See description
40Ch	SPI Interrupt Enable Read and Set Register (INTENSET)	32	RW	See description
410h	SPI Interrupt Enable Clear Register (INTENCLR)	32	RW	See description
420h	SPI Transmit Control Register (TXCTL)	32	RW	See description
424h	SPI Clock Divider Register (DIV)	32	RW	See description
428h	SPI Interrupt Status Register (INTSTAT)	32	RO	See description
E00h	FIFO Configuration and Enable Register (FIFOCFG)	32	RW	See description
E04h	FIFO Status Register (FIFOSTAT)	32	RW	See description
E08h	FIFO Trigger Settings for Interrupt and DMA Request Register (FIFOTRIG)	32	RW	See description
E10h	FIFO Interrupt Enable Set (enable) and Read Register (FIFOINTENSET)	32	RW	See description
E14h	FIFO Interrupt Enable Clear (Disable) and Read Register (FIFOINTENCLR)	32	RW	See description
E18h	FIFO Interrupt Status Register (FIFOINTSTAT)	32	RO	See description

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
E20h	FIFO Write Data Register (FIFOWR)	32	WO	See description
E30h	FIFO Read Data (FIFORD)	32	RO	See description
E40h	FIFO Data Read with no FIFO Pop (FIFORDNOPOP)	32	RO	See description
FF8h	Peripheral Function Select and ID Register (PSELID)	32	RW	See description
FFCh	SPI Module Identifier (ID)	32	RO	E020_1200h

14.1.2 SPI Configuration Register (CFG)

Offset

Register	Offset
CFG	400h

Function

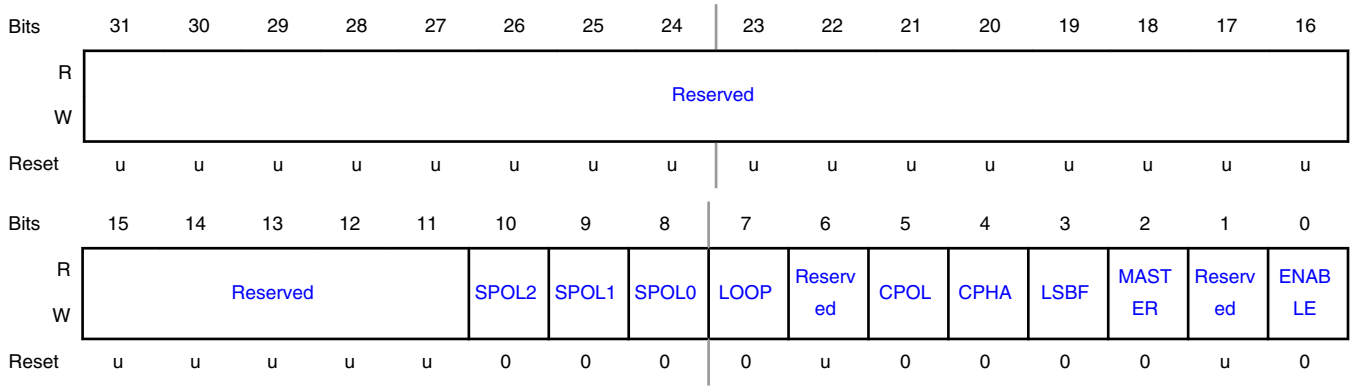
The CFG register contains information for the general configuration of the SPI. Typically, this information is not changed during operation. See the description of the STAT[MSTIDLE] for more information.

NOTE

A setup sequence is recommended for initial SPI setup (after the SPI function has been selected in the PSELID register), and when changes need to be made to settings in the CFG register after the interface has been in use. See the list below. In the case of changing existing settings, the interface should first be disabled by clearing the ENABLE bit once the interface is fully idle. See the description of the STAT[MSTIDLE].

1. Disable the FIFO by clearing the FIFOCFG[ENABLETX] and FIFOCFG[ENABLERX] bits
2. Setup the SPI interface in the CFG register, leaving ENABLE = 0.
3. Enable the FIFO by setting the FIFOCFG[ENABLETX] and FIFOCFG[ENABLERX] bits.
4. Enable the SPI by setting the ENABLE bit in CFG.

Diagram



Fields

Field	Function
31-11 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
10 SPOL2	SSEL2 Polarity Select Valid only for SPI-1 0b - Low. The SSEL2 pin is active low. 1b - High. The SSEL2 pin is active high.
9 SPOL1	SSEL1 Polarity Select Valid only for SPI-1 0b - Low. The SSEL1 pin is active low. 1b - High. The SSEL1 pin is active high.
8 SPOL0	SSEL0 Polarity Select 0b - Low. The SSEL0 pin is active low. 1b - High. The SSEL0 pin is active high.
7 LOOP	Loopback Mode Enable Loopback mode applies only to Master mode, and connects transmit and receive data connected together to allow simple software testing. 0b - Disabled. 1b - Enabled.
6 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
5 CPOL	Clock Polarity Select 0b - Low. The rest state of the clock (between transfers) is low. 1b - High. The rest state of the clock (between transfers) is high.
4 CPHA	Clock Phase Select 0b - Change. The SPI captures serial data on the first clock transition of the transfer (when the clock changes away from the rest state). Data is changed on the following edge. 1b - Capture. The SPI changes serial data on the first clock transition of the transfer (when the clock changes away from the rest state). Data is captured on the following edge.
3 LSBF	LSB First Mode Enable 0b - Standard. Data is transmitted and received in standard MSB first order. 1b - Reverse. Data is transmitted and received in reverse order (LSB first).
2 MASTER	Master Mode Select 0b - Slave Mode. The SPI will operate in slave mode. SCK, MOSI, and the SSEL signals are inputs, MISO is an output. 1b - Master mode. The SPI will operate in master mode. SCK, MOSI, and the SSEL signals are outputs, MISO is an input.
1 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
0 ENABLE	SPI Enable 0b - Disabled. The SPI is disabled and the internal state machine and counters are reset. 1b - Enabled. The SPI is enabled for operation.

14.1.3 SPI Delay Register (DLY)

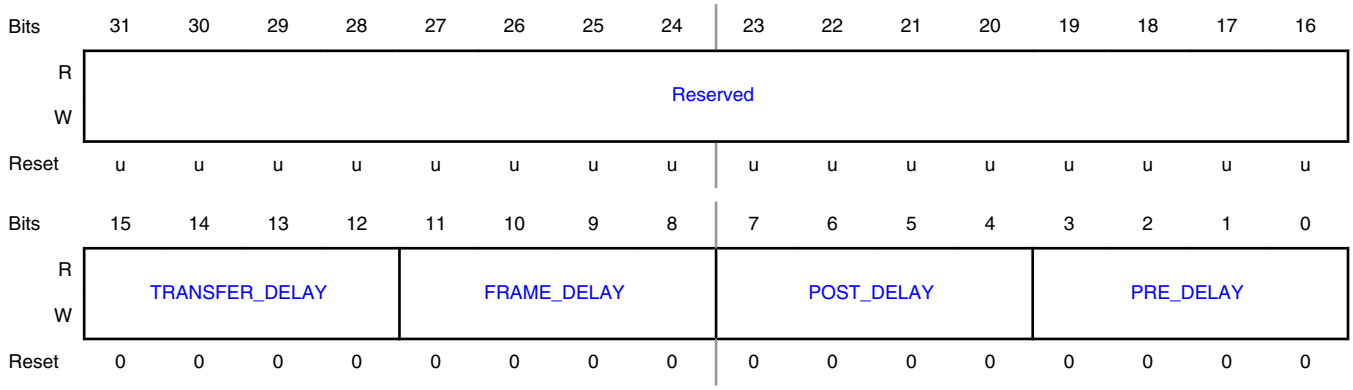
Offset

Register	Offset
DLY	404h

Function

The DLY register controls several programmable delays related to SPI signalling. These delays apply only to master mode, and are all stated in SPI clocks.

Diagram



Fields

Field	Function
31-16 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
15-12 TRANSFER_DELAY	Minimum Time SSEL Deasserted between Transfers This field controls the minimum amount of time that the SSEL is deasserted between transfers. 0000b The minimum time that SSEL is deasserted is 1 SPI clock time. (Zero added time.) 0001b The minimum time that SSEL is deasserted is 2 SPI clock times. 0010b The minimum time that SSEL is deasserted is 3 SPI clock times. ..... 1111b The minimum time that SSEL is deasserted is 16 SPI clock times.
11-8 FRAME_DELAY	Time between Current Frame and Next Frame If the EOFR flag is set, this field controls the minimum amount of time between the current frame and the next frame (or SSEL deassertion if EOTR). 0000b No additional time is inserted. 0001b 1 SPI clock time is inserted. 0010b 2 SPI clock times are inserted. ..... 1111b 15 SPI clock times are inserted.

Table continues on the next page...

Table continued from the previous page...

Field	Function
7-4 POST_DELAY	Time between Data Transfer End and SSEL Deassertion This field controls the amount of time between the end of a data transfer and SSEL deassertion. 0000b No additional time is inserted. 0001b 1 SPI clock time is inserted. 0010b 2 SPI clock times are inserted. ..... 1111b 15 SPI clock times are inserted.
3-0 PRE_DELAY	Time between SSEL Assertion and Data Transfer Start This field controls the amount of time between SSEL assertion and the beginning of a data transfer. There is always one SPI clock time between SSEL assertion and the first clock edge. This is not considered part of the pre-delay. 0000b No additional time is inserted. 0001b 1 SPI clock time is inserted. 0010b 2 SPI clock times are inserted. ..... 1111b 15 SPI clock times are inserted.

14.1.4 SPI Status Register (STAT)

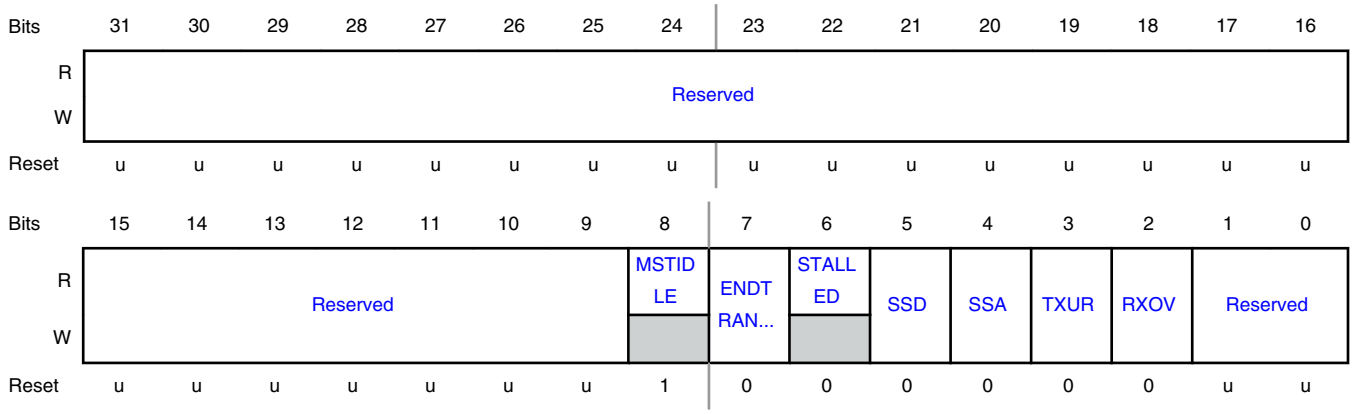
Offset

Register	Offset
STAT	408h

Function

The STAT register provides SPI status flags for software to read, and a control bit for forcing an end of transfer. Flags other than read-only flags may be cleared by writing ones to corresponding bits of STAT.

Diagram



Fields

Field	Function
31-9 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
8 MSTIDLE	Master Idle Status Flag. This bit is 1 whenever the SPI master function is fully idle. This means that the transmit holding register is empty and the transmitter is not in the process of sending data.
7 ENDTRANSFER	End Transfer Control Software can set this bit to force an end to the current transfer when the transmitter finishes any activity already in progress, as if the EOTR flag had been set prior to the last transmission. This capability is included to support cases where it is not known when transmit data is written that it will be the end of a transfer. The bit is cleared when the transmitter becomes idle as the transfer comes to an end. Forcing an end of transfer in this manner causes any specified FRAME_DELAY and TRANSFER_DELAY to be inserted.
6 STALLED	Stalled Status Flag This indicates whether the SPI is currently in a stall condition.
5 SSD	Slave Select Deassert This flag is set whenever any asserted slave selects transition to deasserted, in both master and slave modes. This allows determining when the SPI transmit/receive functions become idle. Write 1 to clear this bit.
4 SSA	Slave Select Assert This flag is set whenever any slave select transitions from deasserted to asserted, in both master and slave modes. This allows determining when the SPI transmit/receive functions become busy, and allows waking up the device from reduced power modes when a slave mode access begins. Write 1 to clear this bit.

Table continues on the next page...

Table continued from the previous page...

Field	Function
3 TXUR	Transmitter Underrun Interrupt Flag This flag applies only to slave mode (Master = 0). In this case, the transmitter must begin sending new data on the next input clock if the transmitter is idle. If that data is not available in the transmitter holding register at that point, there is no data to transmit and the TXUR flag is set. Data transmitted by the SPI should be considered undefined if TXUR is set.
2 RXOV	Receiver Overrun interrupt Flag This flag applies only to slave mode (Master = 0). This flag is set when the beginning of a received character is detected while the receiver buffer is still in use. If this occurs, the receiver buffer contents are preserved, and the incoming data is lost. Data received by the SPI should be considered undefined if RxOv is set.
1-0 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

14.1.5 SPI Interrupt Enable Read and Set Register (INTENSET)

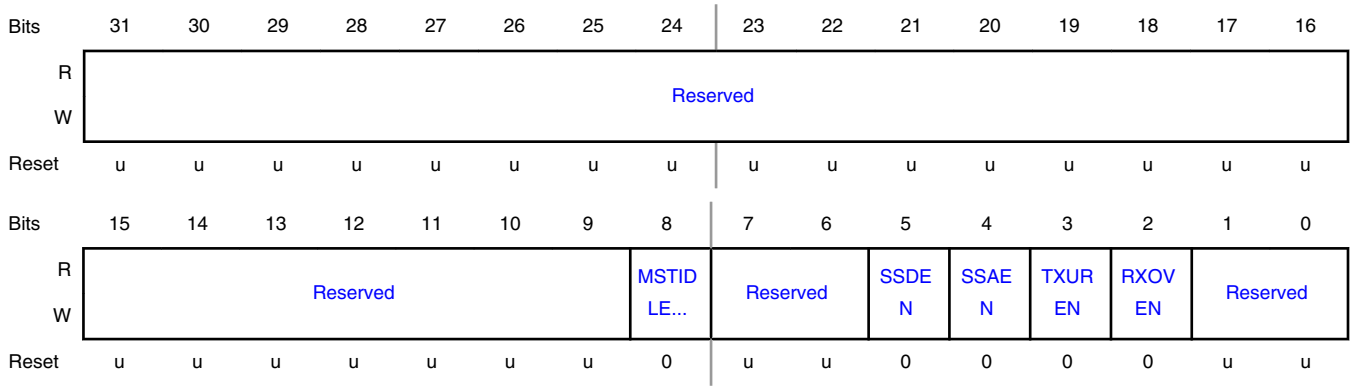
Offset

Register	Offset
INTENSET	40Ch

Function

The INTENSET register is used to enable various SPI interrupt sources. Enable bits in INTENSET are mapped in locations that correspond to the flags in the STAT register. The complete set of interrupt enables may be read from this register. Writing ones to implemented bits in this register causes those bits to be set. The INTENCLR register is used to clear bits in this register. See STAT register for details of the interrupts.

Diagram



Fields

Field	Function
31-9 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
8 MSTIDLEEN	Master Idle Interrupt Enable 0b - No interrupt will be generated when the SPI master function is idle. 1b - An interrupt will be generated when the SPI master function is fully idle.
7-6 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
5 SSDEN	Slave Select Deassert Interrupt Enable Determines whether an interrupt occurs when the Slave Select is deasserted. 0b - Disabled. No interrupt will be generated when all asserted Slave Selects transition to deasserted. 1b - Enabled. An interrupt will be generated when all asserted Slave Selects transition to deasserted.
4 SSAEN	Slave Select Assert Interrupt Enable Determines whether an interrupt occurs when the Slave Select is asserted. 0b - Disabled. No interrupt will be generated when any Slave Select transitions from deasserted to asserted. 1b - Enabled. An interrupt will be generated when any Slave Select transitions from deasserted to asserted.
3 TXUREN	TX Underrun Interrupt Enable Determines whether an interrupt occurs when a transmitter underrun occurs. This happens in slave mode when there is a need to transmit data when none is available. 0b - Disabled. 1b - Enabled.
2 RXOVEN	RX Overrun Interrupt Enable Determines whether an interrupt occurs when a receiver overrun occurs. This happens in slave mode when there is a need for the receiver to move newly received data to the RXDAT register when it is already in use. The interface prevents receiver overrun in Master mode by not allowing a new transmission to begin when a receiver overrun would otherwise occur. 0b - Disabled. 1b - Enabled.
1-0 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

14.1.6 SPI Interrupt Enable Clear Register (INTENCLR)

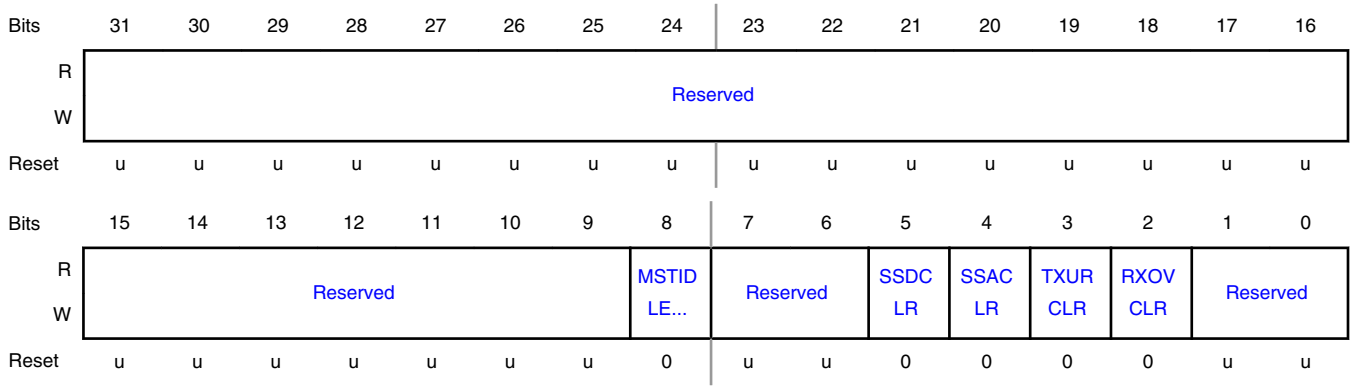
Offset

Register	Offset
INTENCLR	410h

Function

Writing a 1 to any implemented bit position causes the corresponding bit in INTENSET to be cleared.

Diagram



Fields

Field	Function
31-9 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
8 MSTIDLECLR	MSTIDLEEN Clear Writing 1 clears the MSTIDLEEN bit in the INTENSET register.
7-6 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
5 SSDCLR	SSDEN Clear Writing 1 clears the SSDEN bit in the INTENSET register.
4 SSACLR	SSAEN Clear Writing 1 clears the SSAEN bit in the INTENSET register.

Table continues on the next page...

Table continued from the previous page...

Field	Function
3 TXURCLR	TXUREN Clear Writing 1 clears the TXUREN bit in the INTENSET register.
2 RXOVCLR	RXOVEN Clear Writing 1 clears the RXOVEN bit in the INTENSET register.
1-0 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

14.1.7 SPI Transmit Control Register (TXCTL)

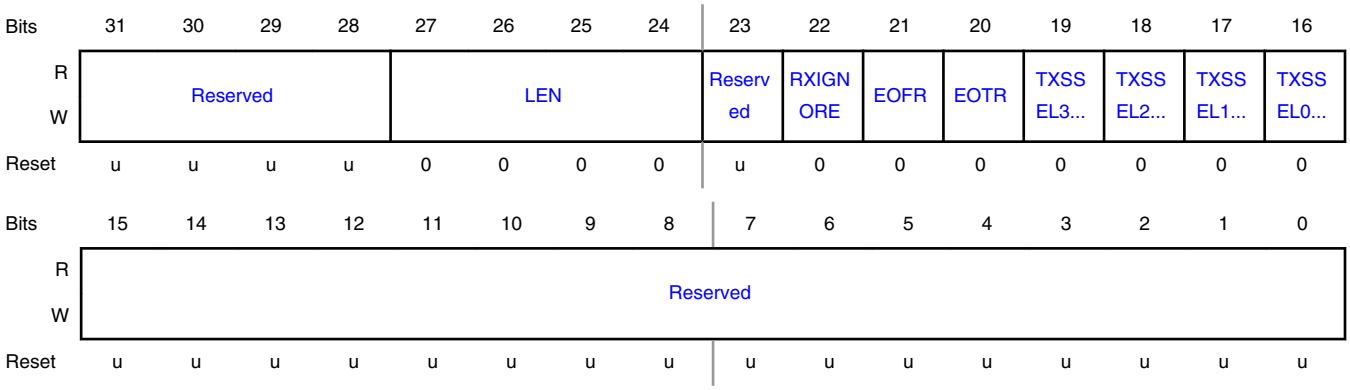
Offset

Register	Offset
TXCTL	420h

Function

If Transmit FIFO is enabled, in FIFOCFG, then values read in this register are affected values in FIFO.

Diagram



Fields

Field	Function
31-28 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
27-24 LEN	Data transfer Length
23 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
22 RXIGNORE	Receive Ignore
21 EOFR	End of Frame
20 EOTR	End of Transfer
19 TXSSEL3_N	[Reserved] Transmit Slave Select 3
18 TXSSEL2_N	Transmit Slave Select 2 Valid only for SPI-1
17 TXSSEL1_N	Transmit Slave Select 1 Valid only for SPI-1
16 TXSSEL0_N	Transmit Slave Select 0
15-0 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

14.1.8 SPI Clock Divider Register (DIV)

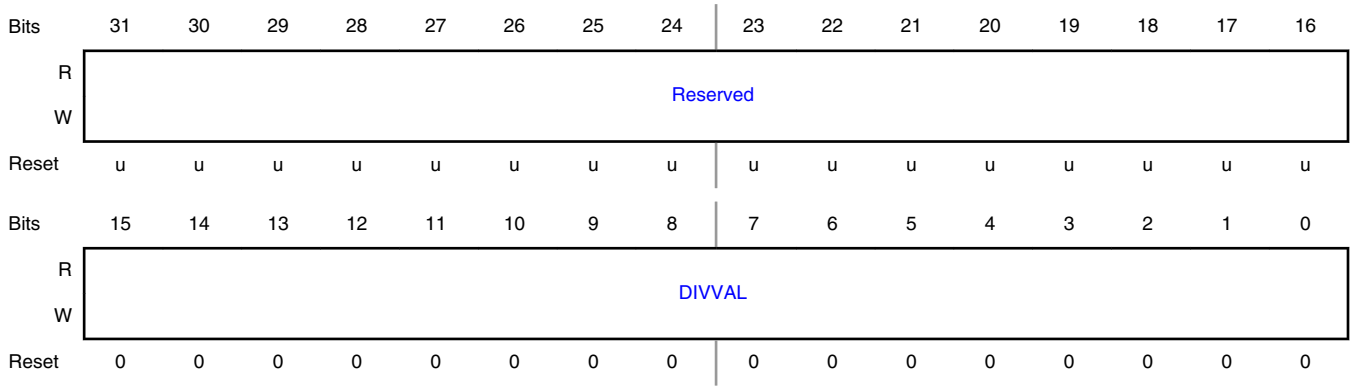
Offset

Register	Offset
DIV	424h

Function

The DIV register determines the clock used by the SPI in master mode.

Diagram



Fields

Field	Function
31-16 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
15-0 DIVVAL	Rate Divider Value Specifies how the SPI Module clock is divided to produce the SPI clock rate in master mode. DIVVAL is -1 encoded such that the value 0 results in SPICLK/1, the value 1 results in SPICLK/2, up to the maximum possible divide value of 0xFFFF, which results in SPICLK/65536.

14.1.9 SPI Interrupt Status Register (INTSTAT)

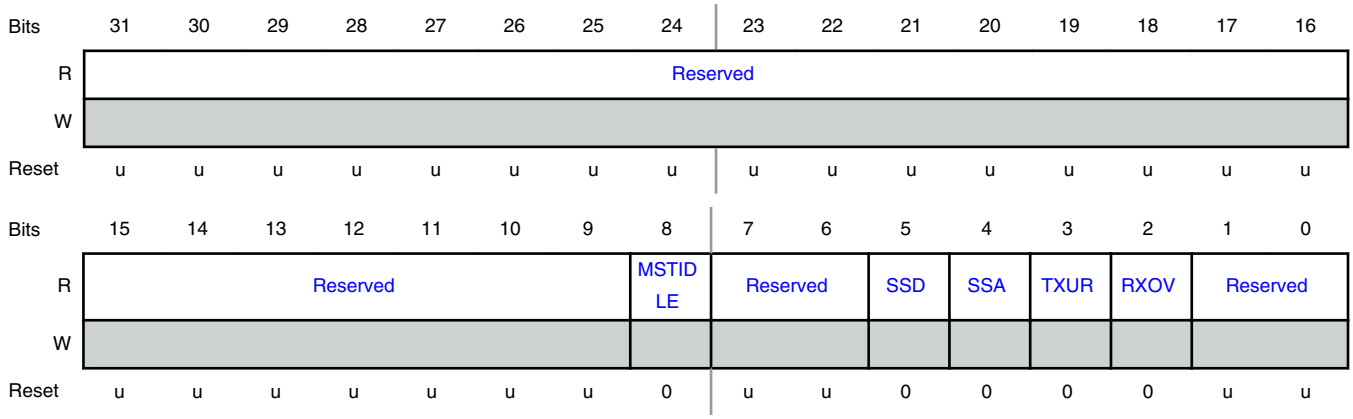
Offset

Register	Offset
INTSTAT	428h

Function

The read-only INTSTAT register provides a view of those interrupt flags that are currently enabled. This can simplify software handling of interrupts. See STAT for detailed descriptions of the interrupt flags.

Diagram



Fields

Field	Function
31-9 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
8 MSTIDLE	Master Idle Status Flag
7-6 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
5 SSD	Slave Select Deassert
4 SSA	Slave Select Assert
3 TXUR	Transmitter Underrun Interrupt Flag
2 RXOV	Receiver Overrun Interrupt Flag
1-0 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

14.1.10 FIFO Configuration and Enable Register (FIFOCFG)

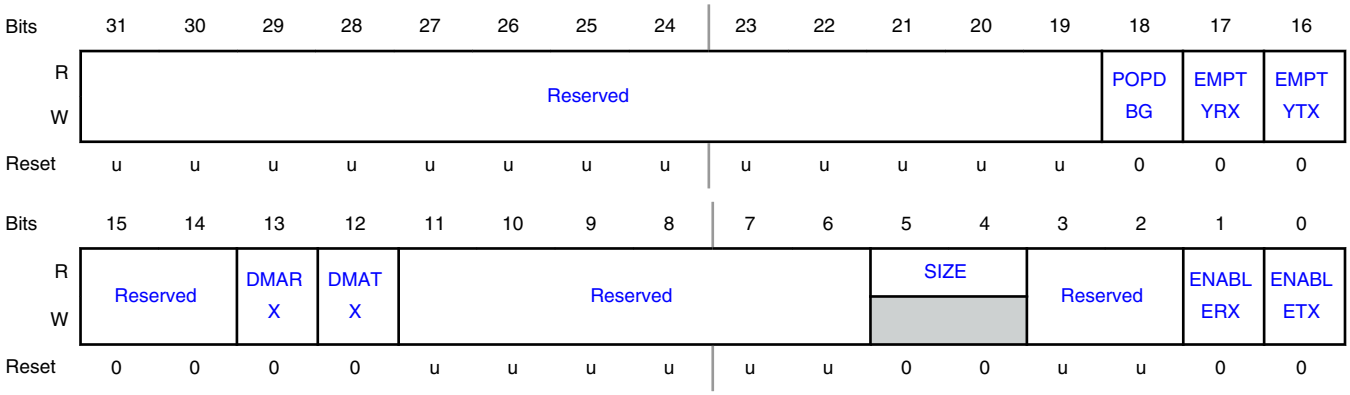
Offset

Register	Offset
FIFOCFG	E00h

Function

This register configures FIFO usage. The SPI function must be enabled in PSELID register before configuring the FIFO.

Diagram



Fields

Field	Function
31-19 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
18 POPDBG	Pop FIFO for Debug Reads
17 EMPTYRX	Empty Command for Receive FIFO When a 1 is written to this bit, the RX FIFO is emptied.
16 EMPTYTX	Empty Command for Transmit FIFO When a 1 is written to this bit, the TX FIFO is emptied.
15-14 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
13 DMARX	DMA Configuration for Receive 0b - DMA is not used for the receive function. 1b - Generate a DMA request for the receive function if the FIFO is not empty. Generally, data interrupts would be disabled if DMA is enabled.
12 DMATX	DMA Configuration for Transmit 0b - DMA is not used for the transmit function. 1b - Generate a DMA request for the transmit function if the FIFO is not full. Generally, data interrupts would be disabled if DMA is enabled.
11-6 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
5-4 SIZE	FIFO Size Configuration 00b - Reset value. 01b - FIFO is configured as 4 entries of 16bits. This value is read after PSELID.PERSEL=2 for SPI functionlaity. 10b - Not applicable. 11b - Not applicable.
3-2 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
1 ENABLERX	Enable Receive FIFO This is automatically enabled when PSELID.PERSEL is set to 2 to configure for SPI functionality 0b - The receive FIFO is not enabled. 1b - The receive FIFO is enabled.
0 ENABLETX	Enable Transmit FIFO This is automatically enabled when PSELID.PERSEL is set to 2 to configure for SPI functionality 0b - The transmit FIFO is not enabled. 1b - The transmit FIFO is enabled.

14.1.11 FIFO Status Register (FIFOSTAT)

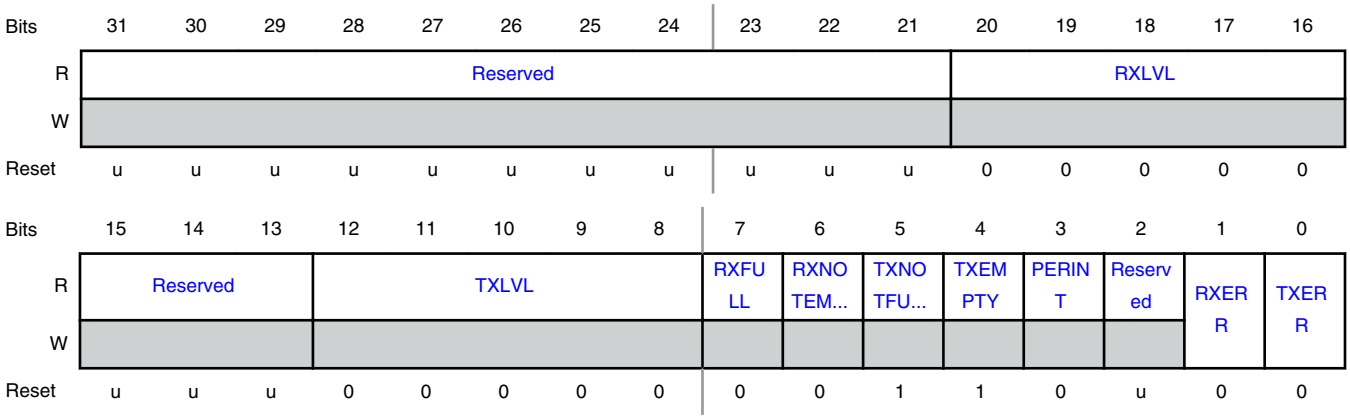
Offset

Register	Offset
FIFOSTAT	E04h

Function

This register provides status information for the FIFO and also indicates an interrupt from the peripheral function.

Diagram



Fields

Field	Function
31-21 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
20-16 RXLVL	Receive FIFO Current Level A 0 means the RX FIFO is currently empty, and the RXFULL and RXNOTEMPTY flags will be 0. Other values tell how much data is actually in the RX FIFO at the point where the read occurs. If the RX FIFO is full, the RXFULL and RXNOTEMPTY flags will be 1.
15-13 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
12-8 TXLVL	Transmit FIFO Current Level A 0 means the TX FIFO is currently empty, and the TXEMPTY and TXNOTFULL flags will be 1. Other values tell how much data is actually in the TX FIFO at the point where the read occurs. If the TX FIFO is full, the TXEMPTY and TXNOTFULL flags will be 0.
7 RXFULL	Receive FIFO Full When 1, the receive FIFO is full. Data needs to be read out to prevent the peripheral from causing an overflow.
6 RXNOTEMPTY	Receive FIFO not Empty Receive FIFO not empty. When 1, the receive FIFO is not empty, so data can be read. When 0, the receive FIFO is empty.

Table continues on the next page...

Table continued from the previous page...

Field	Function
5 TXNOTFULL	Transmit FIFO not Full 0b - The transmit FIFO is full and another write would cause it to overflow. 1b - The transmit FIFO is not full, so more data can be written.
4 TXEMPTY	Transmit FIFO Empty When 1, the transmit FIFO is empty. The peripheral may still be processing the last piece of data.
3 PERINT	Peripheral Interrupt When 1, this indicates that the peripheral function has asserted an interrupt. The details can be found by reading the peripheral s STAT register.
2 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
1 RXERR	RX FIFO Error Will be set if a receive FIFO overflow occurs, caused by software or DMA not emptying the FIFO fast enough. Cleared by writing a 1 to this bit.
0 TXERR	TX FIFO Error Will be set if a transmit FIFO error occurs. This could be an overflow caused by pushing data into a full FIFO, or by an underflow if the FIFO is empty when data is needed. Cleared by writing a 1 to this bit.

14.1.12 FIFO Trigger Settings for Interrupt and DMA Request Register (FIFOTRIG)

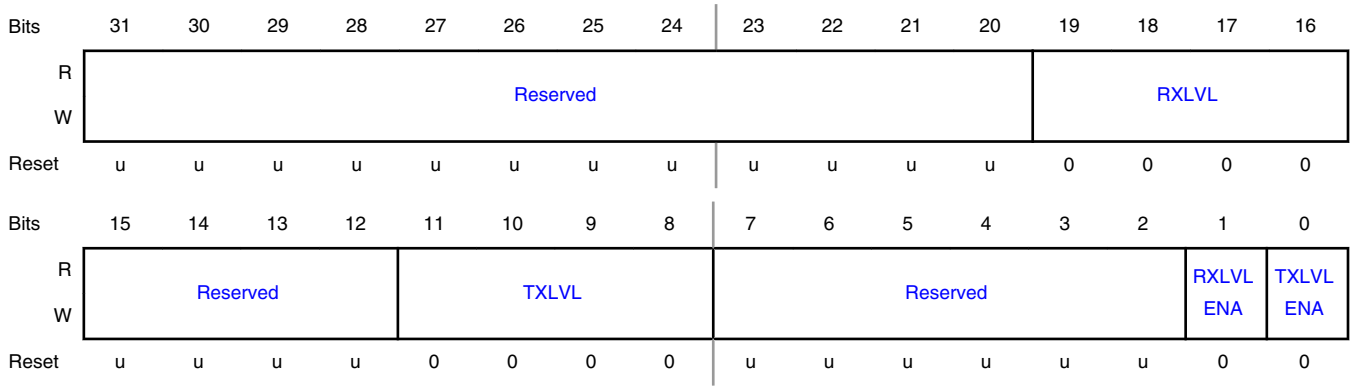
Offset

Register	Offset
FIFOTRIG	E08h

Function

This register allows selecting when FIFO-level related interrupts occur.

Diagram



Fields

Field	Function
31-20 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
19-16 RXLVL	Receive FIFO Level Trigger Point The RX FIFO level is checked when a new piece of data is received. This field is used only when RXLVLENA = 1. 0000b Trigger when the RX FIFO has received one entry (is no longer empty). 0001b Trigger when the RX FIFO has received two entries. ..... 0111b Trigger when the RX FIFO has received 8 entries (has become full).
15-12 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
11-8 TXLVL	Transmit FIFO Level Trigger Point This field is used only when TXLVLENA = 1. 0000b Trigger when the TX FIFO becomes empty. 0001b Trigger when the TX FIFO level decreases to one entry. ..... 0111b Trigger when the TX FIFO level decreases to 7 entries (is no longer full).
7-2 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
1 RXLVLENA	Receive FIFO Level Trigger Enable This trigger will become an interrupt if enabled in FIFOINTENSET, or a DMA trigger if DMARX in FIFOCFG is set. 0b - Receive FIFO level does not generate a FIFO level trigger. 1b - An interrupt will be generated if the receive FIFO level reaches the value specified by the RXLVL field in this register.
0 TXLVLENA	Transmit FIFO Level Trigger Enable This trigger will become an interrupt if enabled in FIFOINTENSET, or a DMA trigger if DMATX in FIFOCFG is set. 0b - Transmit FIFO level does not generate a FIFO level trigger. 1b - An interrupt will be generated if the receive FIFO level reaches the value specified by the RXLVL field in this register.

14.1.13 FIFO Interrupt Enable Set (enable) and Read Register (FIFOINTENSET)

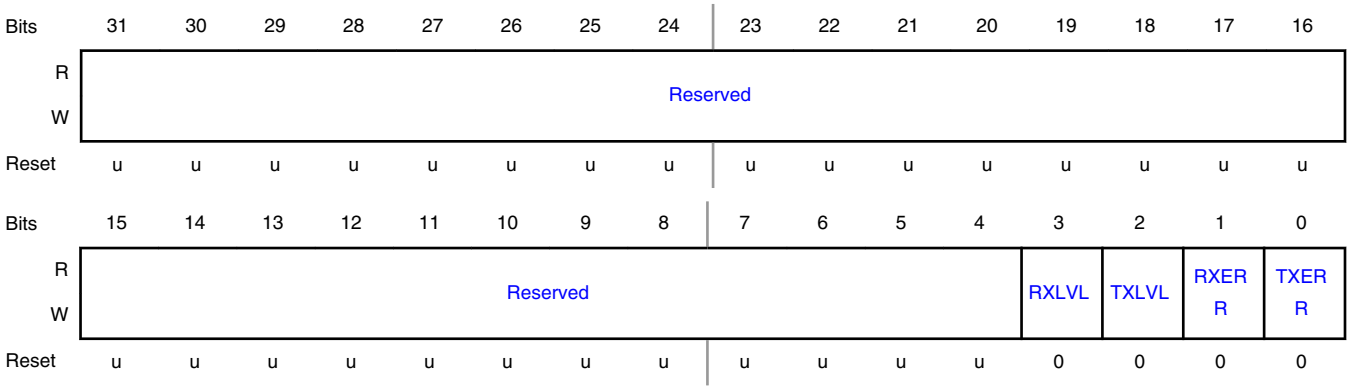
Offset

Register	Offset
FIFOINTENSET	E10h

Function

This register is used to enable various interrupt sources. The complete set of interrupt enables may be read from this register. Writing ones to implemented bits in this register causes those bits to be set. The FIFOINTENCLR register is used to clear bits in this register.

Diagram



Fields

Field	Function
31-4 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
3 RXLVL	Receive FIFO Reach Level Interrupt Determines whether an interrupt occurs when a the receive FIFO reaches the level specified by the TXLVL field in the FIFOTRIG register. 0b - No interrupt will be generated based on the RX FIFO level. 1b - If FIFOTRIG[RXLVLENA] = 1, an interrupt will be generated when the RX FIFO level increases to the level specified by the FIFOTRIG[RXLVL].
2 TXLVL	Transmit FIFO Reach Level Interrupt Determines whether an interrupt occurs when a the transmit FIFO reaches the level specified by the TXLVL field in the FIFOTRIG register. 0b - No interrupt will be generated based on the TX FIFO level. 1b - If FIFOTRIG[TXLVLENA] = 1, an interrupt will be generated when the TX FIFO level decreases to the level specified by the FIFOTRIG[TXLVL].
1 RXERR	Receive Error Interrupt Determines whether an interrupt occurs when a receive error occurs, based on the FIFOSTAT[RXERR] flag. 0b - No interrupt will be generated for a receive error. 1b - An interrupt will be generated when a receive error occurs.
0 TXERR	Transmit Error Interrupt Determines whether an interrupt occurs when a transmit error occurs, based on the FIFOSTAT[TXERR] flag. 0b - No interrupt will be generated for a transmit error. 1b - An interrupt will be generated when a transmit error occurs.

14.1.14 FIFO Interrupt Enable Clear (Disable) and Read Register (FIFOINTENCLR)

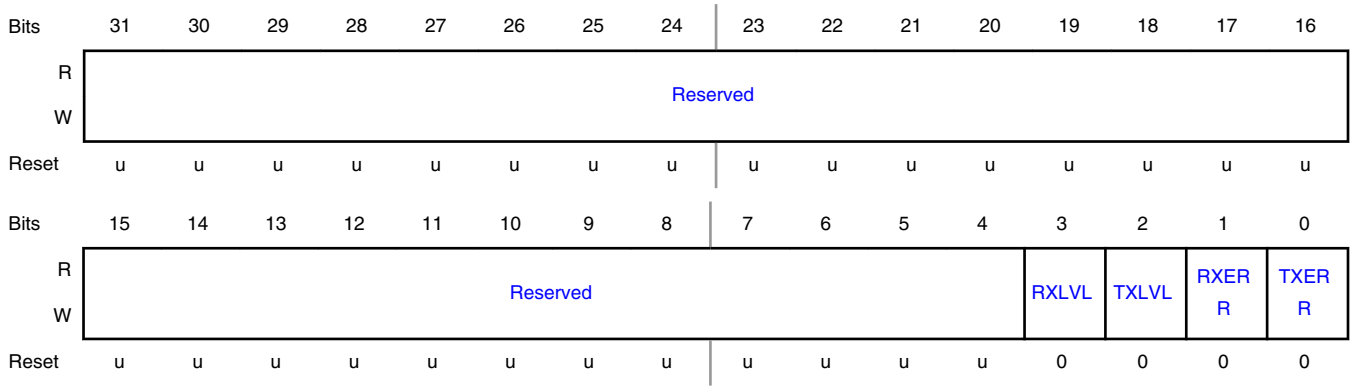
Offset

Register	Offset
FIFOINTENCLR	E14h

Function

The FIFOINTENCLR register is used to clear interrupt enable bits in FIFOINTENSET. The complete set of interrupt enables may also be read from this register as well as FIFOINTENSET.

Diagram



Fields

Field	Function
31-4 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
3 RXLVL	RXLVL Clear Writing 1 clears the RXLVL bit in the FIFOINTENSET register
2 TXLVL	TXLVL Clear Writing 1 clears the TXLVL bit in the FIFOINTENSET register
1 RXERR	RXERR Clear Writing 1 clears the RXERR bit in the FIFOINTENSET register
0 TXERR	TXERR Clear Writing 1 clears the TXERR bit in the FIFOINTENSET register

14.1.15 FIFO Interrupt Status Register (FIFOINTSTAT)

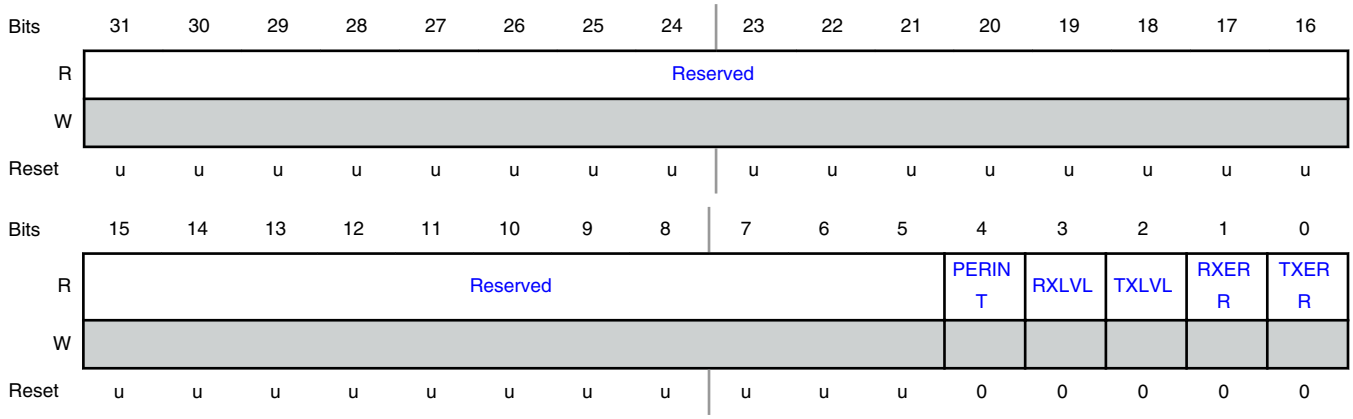
Offset

Register	Offset
FIFOINTSTAT	E18h

Function

The read-only FIFOINTSTAT register provides a view of those interrupt flags that are both pending and currently enabled. This can simplify software handling of interrupts. See FIFOSTAT and FIFOTRIG for details of the interrupts.

Diagram



Fields

Field	Function
31-5 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
4 PERINT	Peripheral Interrupt
3 RXLVL	Receive FIFO Level Interrupt
2 TXLVL	Transmit FIFO Level Interrupt
1 RXERR	RX FIFO Error
0 TXERR	TX FIFO Error

14.1.16 FIFO Write Data Register (FIFOWR)

Offset

Register	Offset
FIFOWR	E20h

Function

The FIFOWR register is used to write values to be transmitted to the FIFO.

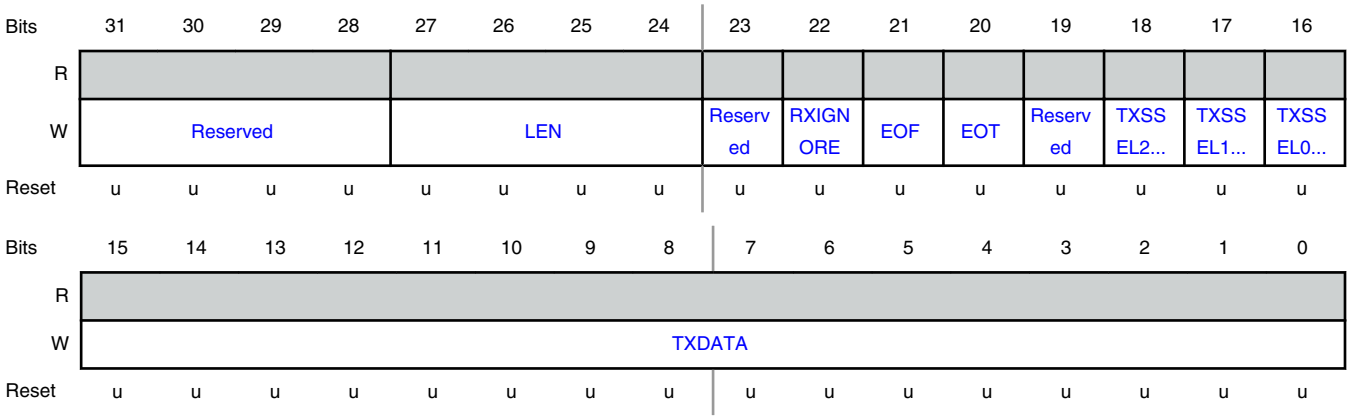
FIFOWR provides the possibility of altering some SPI controls at the same time as sending new data. For example, this can allow a series of SPI transactions involving multiple slaves to be stored in a DMA buffer and sent automatically. These added fields are described for bits 16 through 27 below.

Each FIFO entry holds data and associated control bits. Before data and control bits are pushed into the FIFO, the control bit settings can be modified. Halfword writes to just the control bits (offset 0xE22) does not push anything into the FIFO. A zero written to the upper halfword will not modify the control settings. Non-zero writes to it will modify all the control bits. Note that this is a write only register. Do not read-modify-write the register.

Byte, halfword or word writes to FIFOWR will push the data and control bits into the FIFO. Word writes with the upper halfword of zero, byte writes or halfword writes to FIFOWR will push the data and the current control bits, into the FIFO. Word writes with a non-zero upper halfword will modify the control bits before pushing them onto the stack.

FIFO data is not reset by block reset.

Diagram



Fields

Field	Function
31-28 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
27-24 LEN	Data Length Specifies the data length from 1 to 16 bits. Note that transfer lengths greater than 16 bits are supported by implementing multiple sequential transmits. 0000b-0010b - Reserved 0011b - Data transfer length is 4 bits 0100b - Data transfer length is 5 bits ..... 1111b - Data transfer length is 16 bits
23 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
22 RXIGNORE	Receive Ignore This allows data to be transmitted using the SPI without the need to read unneeded data from the receiver. Setting this bit simplifies the transmit process and can be used with the DMA. 0b - Read received data. Received data must be read first and then the RxData should be written to allow transmission to progress for non-DMA cases. SPI transmit will halt when the receive data FIFO is full. In slave mode, an overrun error will occur if received data is not read before new data is received. 1b - Ignore received data. Received data is ignored, allowing transmission without reading unneeded received data. No receiver flags are generated.
21 EOF	End of Frame Between frames, a delay may be inserted, as defined by the FRAME_DELAY value in the DLY register. The end of a frame may not be particularly meaningful if the FRAME_DELAY value = 0. This control can be used as part of the support for frame lengths greater than 16 bits. 0b - Data not EOF. This piece of data transmitted is not treated as the end of a frame. 1b - Data EOF. This piece of data is treated as the end of a frame, causing the FRAME_DELAY time to be inserted before subsequent data is transmitted.
20 EOT	End of Transfer The asserted SSEL will be deasserted at the end of a transfer, and remain so for at least the time specified by the Transfer_delay value in the DLY register. 0b - SSEL not deasserted. This piece of data is not treated as the end of a transfer. SSEL will not be deasserted at the end of this data. 1b - SSEL deasserted. This piece of data is treated as the end of a transfer. SSEL will be deasserted at the end of this piece of data.
19 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
18 TXSSEL2_N	Transmit Slave Select 2 This field asserts SSEL2 in master mode. The output on the pin is active LOW by default. Remark: The active state of the SSEL2 pin is configured by bits in the CFG register. 0b - SSEL2 asserted. 1b - SSEL2 not asserted.
17 TXSSEL1_N	Transmit Slave Select 1 This field asserts SSEL1 in master mode. The output on the pin is active LOW by default. Remark: The active state of the SSEL1 pin is configured by bits in the CFG register. 0b - SSEL1 asserted. 1b - SSEL1 not asserted.

Table continues on the next page...

Table continued from the previous page...

Field	Function
16 TXSSEL0_N	Transmit Slave Select 0 This field asserts SSEL0 in master mode. The output on the pin is active LOW by default. Remark: The active state of the SSEL0 pin is configured by bits in the CFG register. 0b - SSEL0 asserted. 1b - SSEL0 not asserted.
15-0 TXDATA	Transmit Data to FIFO

14.1.17 FIFO Read Data (FIFORD)

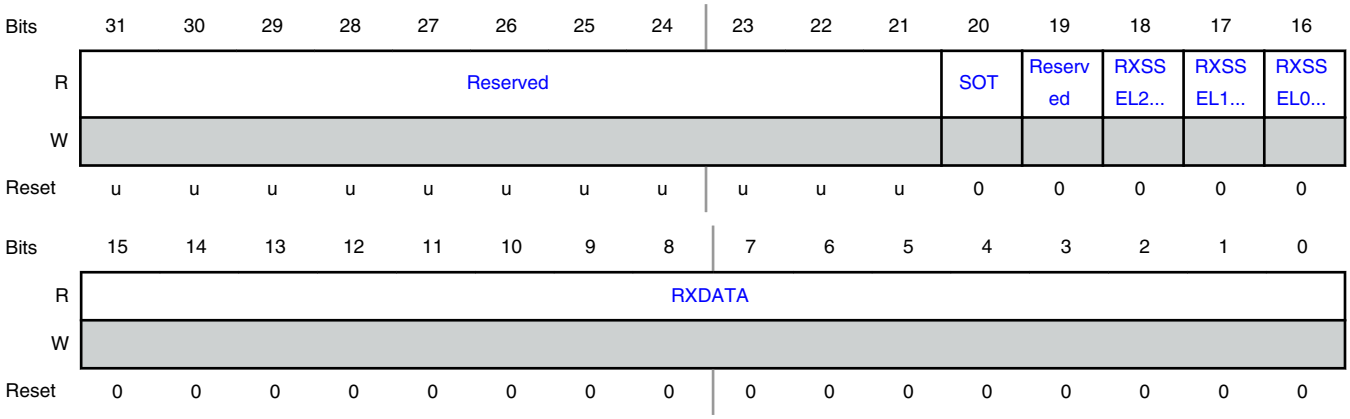
Offset

Register	Offset
FIFORD	E30h

Function

The FIFORD register is used to read values that have been received by the FIFO.

Diagram



Fields

Field	Function
31-21 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
20 SOT	Start of Transfer Flag This flag will be 1 if this is the first data after the SSELs went from deasserted to asserted (i.e., any previous transfer has ended). This information can be used to identify the first piece of data in cases where the transfer length is greater than 16 bit.
19 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
18 RXSSEL2_N	Slave Select for Receive 2 This field allows the state of the SSEL2 pin to be saved along with received data. The value will reflect the SSEL2 pin for both master and slave operation. A zero indicates that a slave select is active. The actual polarity of each slave select pin is configured by the related SPOL bit in CFG.
17 RXSSEL1_N	Slave Select for Receive 1 Slave Select for receive. This field allows the state of the SSEL1 pin to be saved along with received data. The value will reflect the SSEL1 pin for both master and slave operation. A zero indicates that a slave select is active. The actual polarity of each slave select pin is configured by the related SPOL bit in CFG.
16 RXSSEL0_N	Slave Select for Receive This field allows the state of the SSEL0 pin to be saved along with received data. The value will reflect the SSEL0 pin for both master and slave operation. A zero indicates that a slave select is active. The actual polarity of each slave select pin is configured by the related SPOL bit in CFG.
15-0 RXDATA	Received Data from FIFO

14.1.18 FIFO Data Read with no FIFO Pop (FIFORDNOPOP)

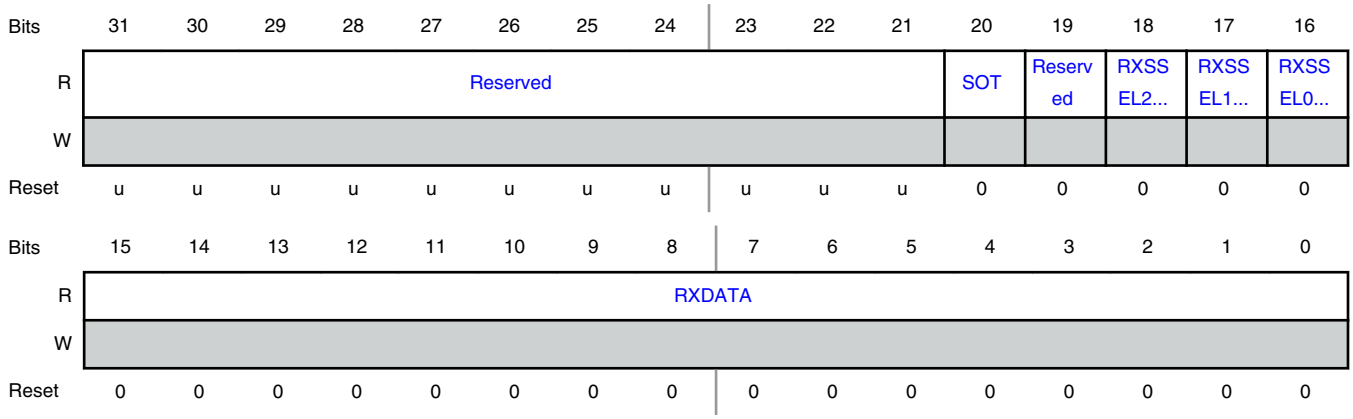
Offset

Register	Offset
FIFORDNOPOP	E40h

Function

This register acts in exactly the same way as FIFORD, except that it supplies data from the top of the FIFO without popping the FIFO (i.e. leaving the FIFO state unchanged). This could be used to allow system software to observe incoming data without interfering with the peripheral driver.

Diagram



Fields

Field	Function
31-21 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
20 SOT	Start of Transfer Flag This flag will be 1 if this is the first data after the SSELs went from deasserted to asserted (i.e., any previous transfer has ended). This information can be used to identify the first piece of data in cases where the transfer length is greater than 16 bit.
19 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
18 RXSSEL2_N	Slave Select for Receive 2 This field allows the state of the SSEL2 pin to be saved along with received data. The value will reflect the SSEL2 pin for both master and slave operation. A zero indicates that a slave select is active. The actual polarity of each slave select pin is configured by the related SPOL bit in CFG.
17 RXSSEL1_N	Slave Select for Receive 1 This field allows the state of the SSEL1 pin to be saved along with received data. The value will reflect the SSEL1 pin for both master and slave operation. A zero indicates that a slave select is active. The actual polarity of each slave select pin is configured by the related SPOL bit in CFG.
16 RXSSEL0_N	Slave Select for Receive 0 This field allows the state of the SSEL0 pin to be saved along with received data. The value will reflect the SSEL0 pin for both master and slave operation. A zero indicates that a slave select is active. The actual polarity of each slave select pin is configured by the related SPOL bit in CFG.
15-0 RXDATA	Received Data from FIFO

14.1.19 Peripheral Function Select and ID Register (PSELID)

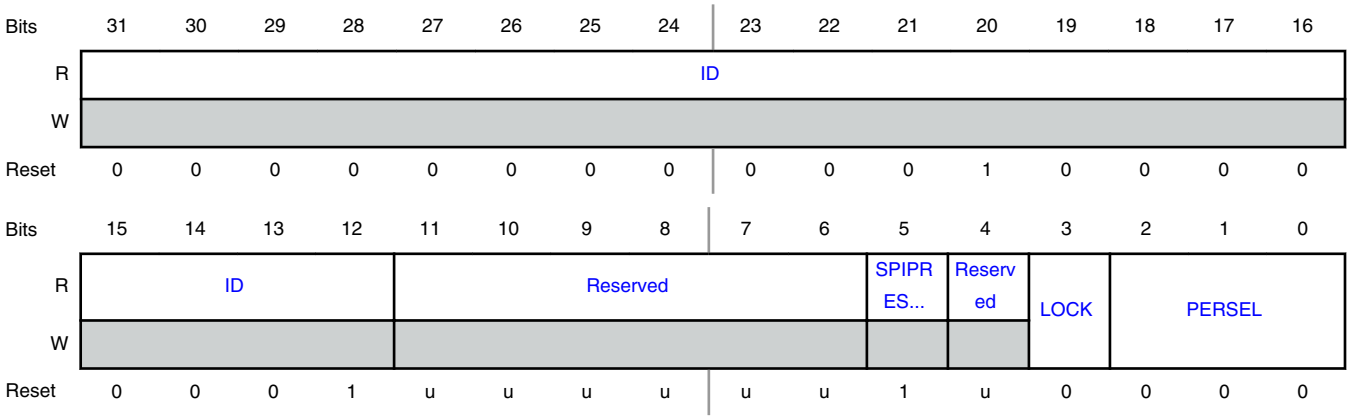
Offset

Register	Offset
PSELID	FF8h

Function

This register is used to enable the SPI FIFO operation.

Diagram



Fields

Field	Function
31-12 ID	Peripheral Select ID
11-6 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
5 SPIPRESENT	SPI Present Indicator This field is Read-only and has value 0x1 to indicate SPI function is present.
4 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
3 LOCK	Lock the Peripheral Select This field is writable by software. 0 Peripheral select can be changed by software. 1 Peripheral select is locked and cannot be changed until this peripheral or the entire device is reset.

Table continues on the next page...

Table continued from the previous page...

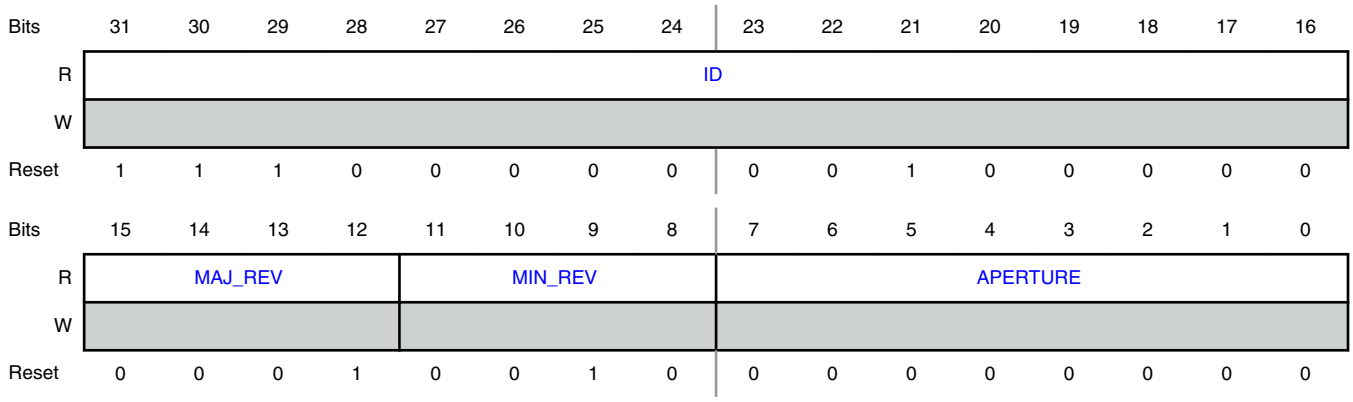
Field	Function
2-0 PERSEL	Peripheral Select This field is writable by software. Reset value is 0x0 showing that no peripheral is selected. Write 0x2 to select the SPI function. All other values are not valid.

14.1.20 SPI Module Identifier (ID)

Offset

Register	Offset
ID	FFCh

Diagram



Fields

Field	Function
31-16 ID	Identifier This is the unique identifier of the module
15-12 MAJ_REV	Major Revision Major revision implies software modifications
11-8 MIN_REV	Minor Revision Minor revision with no software consequences
7-0 APERTURE	Aperture Aperture number minus 1 of consecutive packets 4 Kbytes reserved for this IP

Chapter 15
Inter-Integrated Circuit (I2C)

15.1 I2C register descriptions

15.1.1 I2C memory map

I2C0 base address: 4000_3000h

I2C1 base address: 4000_4000h

I2C2 base address: 4000_5000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Configuration Register (CFG)	32	RW	See description
4h	Status Register (STAT)	32	RW	See description
8h	Interrupt Enable Set and Read Register (INTENSET)	32	RW	See description
Ch	Interrupt Enable Clear Register (INTENCLR)	32	WO	See description
10h	Time-out Value Register (TIMEOUT)	32	RW	See description
14h	Clock Pre-divider Register (CLKDIV)	32	RW	See description
18h	Interrupt Status Register (INTSTAT)	32	RO	See description
20h	Master Control Register (MSTCTL)	32	RW	See description
24h	Master Timing Configuration Register (MSTTIME)	32	RW	See description
28h	Combined Master Receiver and Transmitter Data Register (MSTDAT)	32	RW	See description
40h	Slave Control Register (SLVCTL)	32	RW	See description
44h	Combined Slave Receiver and Transmitter Data Register (SLVDAT)	32	RW	See description
48h	Slave Address 0 (SLVADR0)	32	RW	See description

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
4Ch	Slave Address 1 (SLVADR1)	32	RW	See description
50h	Slave Address 2 (SLVADR2)	32	RW	See description
54h	Slave Address 3 (SLVADR3)	32	RW	See description
58h	Slave Qualification for Address 0 (SLVQUAL0)	32	RW	See description
80h	Monitor Receiver Data Register (MONRXDAT)	32	RO	See description
FFCh	I2C Module Identifier (ID)	32	RO	E030_1300h

15.1.2 Configuration Register (CFG)

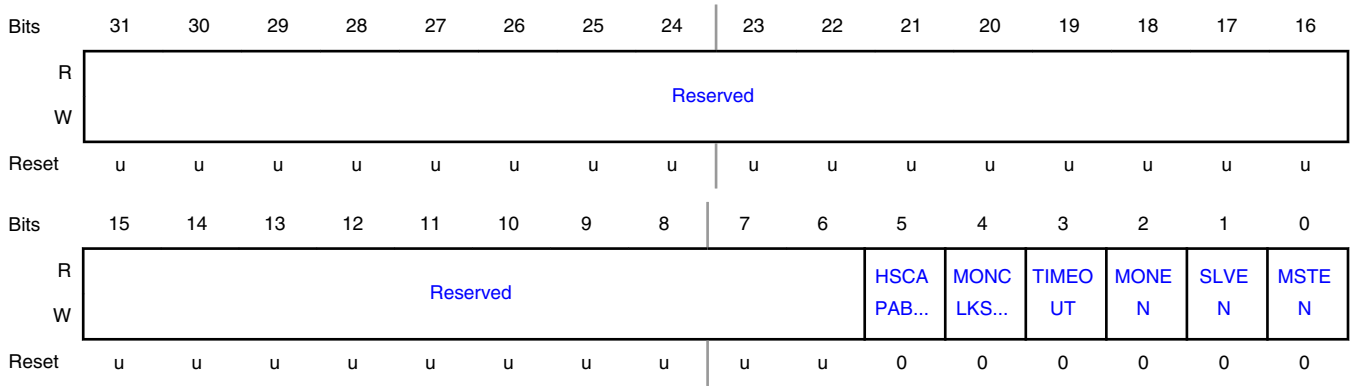
Offset

Register	Offset
CFG	0h

Function

The CFG register contains mode settings that apply to Master, Slave, and Monitor functions.

Diagram



Fields

Field	Function
31-6 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
5 HSCAPABLE	High-speed Mode Capable Enable Since High Speed mode alters the way I2C pins drive and filter, as well as the timing for certain I2C signalling, enabling High-speed mode applies to all functions: master, slave, and monitor. 0b - Standard or Fast mode. The I2C interface supports Standard-mode, Fast-mode, and Fast-mode Plus, to the extent that the pin electronics support these modes. Any changes that need to be made to the pin controls, such as changing the drive strength or filtering, must be made by software via the IOCON register associated with each I2C pin. 1b - High-speed mode. In addition to Standard-mode, Fast-mode, and Fast-mode Plus, the I2C interface will support High-speed mode (slave mode only) to the extent that the pin electronics support these modes.
4 MONCLKSTR	Monitor Function Clock Stretching 0b - Disabled. The Monitor function will not perform clock stretching. Software or DMA may not always be able to read data provided by the Monitor function before it is overwritten. This mode may be used when non-invasive monitoring is critical. 1b - Enabled. The Monitor function will perform clock stretching in order to ensure that software or DMA can read all incoming data supplied by the Monitor function.
3 TIMEOUT	I2C bus Time-out Enable When disabled, the time-out function is internally reset. 0b - Timeout function is disabled. 1b - Time-out function is enabled. Both types of time-out flags will be generated and will cause interrupts if they are enabled. Typically, only one time-out will be used in a system.
2 MONEN	Monitor Enable When disabled, configurations settings for the Monitor function are not changed, but the Monitor function is internally reset. 0b - Monitor function is disabled. 1b - Monitor function is enabled.
1 SLVEN	Slave Enable When disabled, configurations settings for the Slave function are not changed, but the Slave function is internally reset. 0b - Slave function is disabled. 1b - Slave function is enabled.

Table continues on the next page...

Table continued from the previous page...

Field	Function
0 MSTEN	Master Enable When disabled, configurations settings for the Master function are not changed, but the Master function is internally reset. 0b - Master function is disabled. 1b - Master function is enabled.

15.1.3 Status Register (STAT)

Offset

Register	Offset
STAT	4h

Function

The STAT register provides status flags and state information about all of the functions of the I2C interface.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved						SCLTIME...	EVENTTI...	Reserved				MONIDLE	MONACTI...	MONOV	MONRDY
W							W1C	W1C					W1C		W1C	
Reset	u	u	u	u	u	u	0	0	u	u	u	u	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SLVDESEL	SLVSEL	SLVIDX		SLVNOTS...	SLVSTATE		SLVPEND...	Reserved	MSTSTST...	Reserved	MSTARBL...	MSTSTATE			MSTPEND...
W	W1C									W1C		W1C				
Reset	0	0	0	0	1	0	0	0	u	0	u	0	0	0	0	1

Fields

Field	Function
31-26 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
25 SCLTIMEOUT	SCL Time-out Interrupt Flag Indicates when SCL has remained low longer than the time specific by the TIMEOUT register. The flag is cleared by writing a 1 to this bit. 0b - No time-out. SCL low time has not caused a time-out. 1b - Time-out. SCL low time has not caused a time-out.
24 EVENTTIMEOUT	Event Time-out Interrupt Flag Indicates when the time between events has been longer than the time specified by the TIMEOUT register. Events include Start, Stop, and clock edges. The flag is cleared by writing a 1 to this bit. No time-out is created when the I2C-bus is idle. 0b - No time-out. I2C bus events have not casued a time-out. 1b - Event time-out. The time between I2C bus events has been longer than the time specified by the TIMEOUT register.
23-20 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
19 MONIDLE	Monitor Idle Flag This flag is set when the Monitor function sees the I2C bus change from active to inactive. This can be used by software to decide when to process data accumulated by the Monitor function. This flag will cause an interrupt when set if enabled via the INTENSET register. The flag can be cleared by writing a 1 to this bit. 0b - Not idle. The I2C bus is not idle, or this flag has been cleared by software. 1b - Idle. The I2C bus has gone idle at least once since the last time this flag was cleared by software.
18 MONACTIVE	Monitor Active Flag Indicates when the Monitor function considers the I2C bus to be active. Active is defined here as when some Master is on the bus: a bus Start has occurred more recently than a bus Stop. 0b - Inactive. The Monitor function considers the I2C bus to be inactive. 1b - Active. The Monitor function considers the I2C bus to be active.
17 MONOV	Monitor Overflow Flag 0b - No overrun. Monitor data has not overrun. 1b - Overrun. A monitor data overrun has occurred. This can only happen when Monitor clock stretching is not enabled via the MOCCLKSTR bit in the CFG register. Writing 1 to this bit clears the flag.

Table continues on the next page...

Table continued from the previous page...

Field	Function
16 MONRDY	Monitor Ready This flag is cleared when the MONRXDAT register is read. 0b - No data. The Monitor function does not currently have data available. 1b - Data waiting. The Monitor function has data waiting to be read.
15 SLVDESEL	Slave Deselected Flag This flag will cause an interrupt when set if enabled via INTENSET. This flag can be cleared by writing a 1 to this bit. 0b - Not deselected. The slave function has not become deslected. This does not mean that it is currently selected. That information can be found in the SLVSEL flag. 1b - Deselected. The slave function has become deselected. This is specifically caused by the SLVSEL flag changing from 1 to 0. See the description of SLVSEL for details on when that event occurs.
14 SLVSEL	Slave Selected Flag SLVSEL is set after an address match when software tells the Slave function to acknowledge the address, or when the address has been automatically acknowledged. It is cleared when another address cycle presents an address that does not match an enabled address on the Slave function, when slave software decides to NACK a matched address, when there is a Stop detected on the bus, when the master NACKs slave data, and in some combinations of Automatic Operation. SLVSEL is not cleared if software NACKs data. 0b - Not selected. The slave function is not currently selected. 1b - Selected. The slave function is currently selected.
13-12 SLVIDX	Slave Address Match Index This field is valid when the I2C slave function has been selected by receiving an address that matches one of the slave addresses defined by any enabled slave address registers, and provides an identification of the address that was matched. It is possible that more than one address could be matched, but only one match can be reported here. 00b - Address 0. Slave address 0 was matched. 01b - Address 1. Slave address 1 was matched. 10b - Address 2. Slave address 2 was matched. 11b - Address 3. Slave address 3 was matched.

Table continues on the next page...

Table continued from the previous page...

Field	Function
11 SLVNOTSTR	Slave Not Stretching Indicates when the slave function is stretching the I2C clock. This is needed in order to gracefully invoke Deep Sleep or Power-down modes during slave operation. This read-only flag reflects the slave function status in real time. 0b - Stretching. The slave function is currently stretching the I2C bus clock. Deep-sleep mode can not be entered at this time. 1b - Not stretching. The slave function is not currently stretching the I2C bus clock. Deep-sleep mode could be entered at this time.
10-9 SLVSTATE	Slave State Code Each value of this field indicates a specific required service for the Slave function. All other values are reserved. See Table 393 for state values and actions. Remark: note that the occurrence of some states and how they are handled are affected by DMA mode and Automatic Operation modes. 00b - Slave address. Address plus R/W received. At least one of the four slave addresses has been matched by hardware. 01b - Slave receive. Received data is available (Slave Receiver mode). 10b - Slave transmit. Data can be transmitted (Slave transmitter mode). 11b - Reserved.
8 SLVPENDING	Slave Pending Indicates that the Slave function is waiting to continue communication on the I2C-bus and needs software service. This flag will cause an interrupt when set if enabled via INTENSET. The SLVPENDING flag is not set when the DMA is handling an event (if the SLVDMA bit in the SLVCTL register is set). The SLVPENDING flag is read-only and is automatically cleared when a 1 is written to the SLVCONTINUE bit in the SLVCTL register. The point in time when SlvPending is set depends on whether the I2C interface is in HSCAPABLE mode. When the I2C interface is configured to be HSCAPABLE, HS master codes are detected automatically. Due to the requirements of the HS I2C specification, slave addresses must also be detected automatically, since the address must be acknowledged before the clock can be stretched. 0b - In progress. The slave function does not currently need service. 1b - Pending. The Slave function needs service. Information on what is needed can be found in the adjacent SLVSTATE field.
7 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
6 MSTSTSTPERR	Master Start/Stop Error Flag This flag can be cleared by software writing a 1 to this bit. It is also cleared automatically when a 1 is written to MSTCONTINUE. 0b - No start/stop Error has occurred. 1b - The master function has experienced a Start/Stop Error. A Start or Stop was detected at a time when it is not allowed by the I2C specification. The Master interface has stopped driving the bus and gone to an idle state, no action is required. A request for a Start could be made, or software could attempt to insure that thet bus has not stalled.
5 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
4 MSTARBLOSS	Master Arbitration Loss Flag This flag can be cleared by software writing a 1 to this bit. It is also cleared automatically when a 1 is written to MSTCONTINUE. 0b - No Arbitration Loss has occurred. 1b - Arbitration Loss. The Master function has experienced an Arbitration Loss. At this point, the Master function has already stopped driving the bus and gone to an idle state. Software can respond by doing nothing, or by sending a Start in order to attempt to gain control of the bus when it next becomes idle.
3-1 MSTSTATE	Master State Code The master state code reflects the master state when the MSTPENDING bit is set, that is, the master is pending or in the idle state. Each value of this field indicates a specific required service for the Master function. 000b - Idle. The Master function is available to be used for a new transaction. 001b - Receive ready. Received data available (Master Receiver mode). Address plus Read was previously sent and Acknowledged by slave. 010b - Transmit ready. Data can be transmitted (Master Transmitter mode). Address plus Write was previously sent and Acknowledged by slave. 011b - NACK address. Slave NACKed address. 100b - NACK data. Slave NACKed transmitted data. 101b - Reserved. 110b - Reserved. 111b - Reserved.

Table continues on the next page...

Table continued from the previous page...

Field	Function
0 MSTPENDING	Master Pending Indicates that the Master is waiting to continue communication on the I2C-bus (pending) or is idle. When the master is pending, the MSTSTATE bits indicate what type of software service if any the master expects. This flag will cause an interrupt when set, if enabled via the INTENSET register. The MSTPENDING flag is not set when the DMA is handling an event (if the MSTCTL[MSTDMA] bit is set). If the master is in the idle state, and no communication is needed, mask this interrupt. 0b - In progress. Communication is in progress and the Master function is busy and cannot currently accept a command. 1b - Pending. The Master function needs software service or is in the idle state. If the master is not in the idle state, it is waiting to receive or transmit data or the NACK bit.

15.1.4 Interrupt Enable Set and Read Register (INTENSET)

Offset

Register	Offset
INTENSET	8h

Function

The INTENSET register controls which I2C status flags generate interrupts. Writing a 1 to a bit position in this register enables an interrupt in the corresponding position in the STAT register, if an interrupt is supported there. Reading INTENSET indicates which interrupts are currently enabled.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved						SCLTIME...	EVENTTI...	Reserved				MONIDLE...	Reserved	MONOVEN	MONRDYEN
Reset	u	u	u	u	u	u	0	0	u	u	u	u	0	u	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SLVDESE...	Reserved			SLVNOTS...	Reserved		SLVPEND...	Reserved	MSTSTST...	Reserved	MSTARBL...	Reserved			MSTPEND...
Reset	0	u	u	u	0	u	u	0	u	0	u	0	u	u	u	0

Fields

Field	Function
31-26 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
25 SCLTIMEOUTEN	SCL Time-out Interrupt Enable 0b - Interrupt is disabled. 1b - Interrupt is enabled.
24 EVENTTIMEOUTEN	Event Time-out Interrupt Enable 0b - Interrupt is disabled. 1b - Interrupt is enabled.
23-20 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
19 MONIDLEEN	Monitor Idle Interrupt Enable. 0b - Interrupt is disabled. 1b - Interrupt is enabled.
18 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
17 MONOVEN	Monitor Overrun Interrupt Enable 0b - Interrupt is disabled. 1b - Interrupt is enabled.
16 MONRDYEN	Monitor Data Ready Interrupt Enable 0b - Interrupt is disabled. 1b - Interrupt is enabled.
15 SLVDESELEN	Slave Deselect Interrupt Enable 0b - Interrupt is disabled. 1b - Interrupt is enabled.
14-12 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
11 SLVNOTSTREN	Slave Not Stretching Interrupt Enable 0b - Interrupt is disabled. 1b - Interrupt is enabled.
10-9 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
8 SLVPENDINGEN	Slave Pending Interrupt Enable 0b - Interrupt is disabled. 1b - Interrupt is enabled.
7 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
6 MSTSTSTPERREN	Master Start/Stop Error Interrupt Enable 0b - Interrupt is disabled. 1b - Interrupt is enabled.
5 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
4 MSTARBLOSSEN	Master Arbitration Loss Interrupt Enable 0b - Interrupt is disabled. 1b - Interrupt is enabled.
3-1 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
0 MSTPENDINGEN	Master Pending Interrupt Enable 0b - Interrupt is disabled. 1b - Interrupt is enabled.

15.1.5 Interrupt Enable Clear Register (INTENCLR)

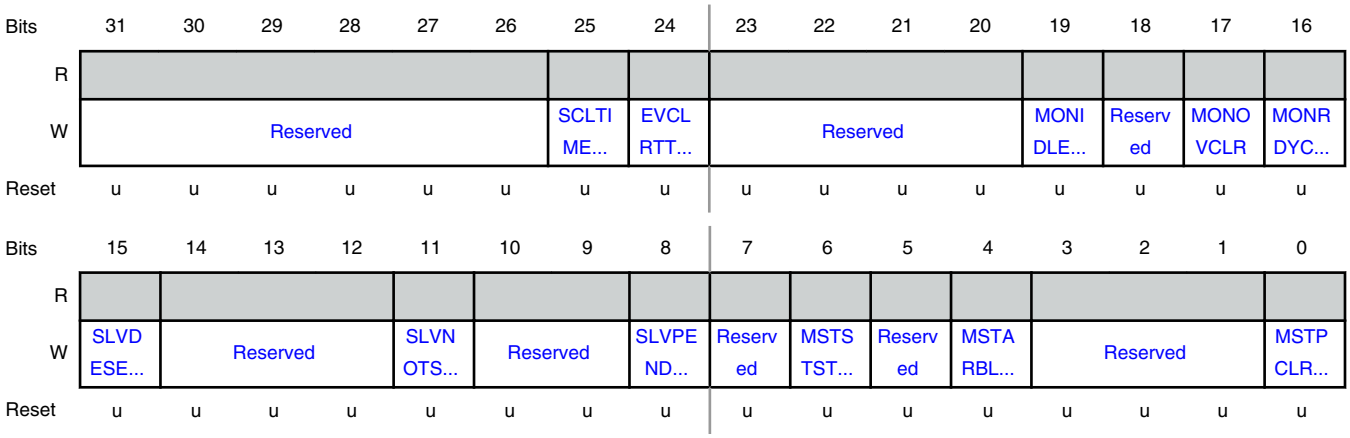
Offset

Register	Offset
INTENCLR	Ch

Function

Writing a 1 to this bit clears the corresponding bit in the INTENSET register, disabling that interrupt. This is a Write-only register. Bits that do not correspond to defined bits in INTENSET are reserved and only zeros should be written to them.

Diagram



Fields

Field	Function
31-26 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
25 SCLTIMEOUTCLR	SCL Time-out Interrupt Clear
24 EVCLRTTIMEOUTCLR	Event Time-out Interrupt Clear
23-20 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
19 MONIDLECLR	Monitor Idle Interrupt Clear
18 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
17 MONOVCLR	Monitor Overrun Interrupt Clear

Table continues on the next page...

Table continued from the previous page...

Field	Function
16 MONRDYCLR	Monitor Data Ready Interrupt Clear
15 SLVDESELCLR	Slave Deselect Interrupt Clear
14-12 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
11 SLVNOTSTRCLR	Slave Not Stretching Interrupt Clear
10-9 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
8 SLVPENDINGCLR	Slave Pending Interrupt Clear
7 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
6 MSTSTSTPERRCLR	Master Start/Stop Error Interrupt Clear
5 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
4 MSTARBLOSSCLR	Master Arbitration Loss Interrupt Clear
3-1 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
0 MSTPCLRDINGCLR	Master Pending Interrupt Clear

15.1.6 Time-out Value Register (TIMEOUT)

Offset

Register	Offset
TIMEOUT	10h

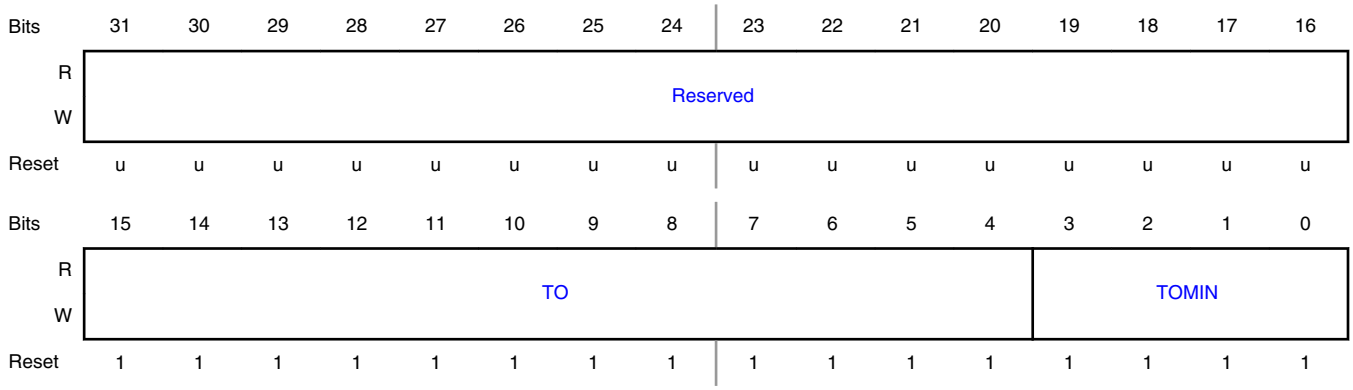
Function

The TIMEOUT register allows setting an upper limit to certain I2C-bus times, informing by status flag and/or interrupt when those times are exceeded.

Two time-outs are generated, and software can select to use either of them.

- EVENTTIMEOUT checks the time between bus events while the bus is not idle: Start, SCL rising, SCL falling, and Stop. The EVENTTIMEOUT status flag in the STAT register is set if the time between any two events becomes longer than the time configured in the TIMEOUT register. The EVENTTIMEOUT status flag can cause an interrupt if enabled to do so by the INTENSET[EVENTTIMEOUTEN] bit.
- SCLTIMEOUT checks only the time that the SCL signal remains low while the bus is not idle. The SCLTIMEOUT status flag in the STAT register is set if SCL remains low longer than the time configured in the TIMEOUT register. The SCLTIMEOUT status flag can cause an interrupt if enabled to do so by the INTENSET[SCLTIMEOUTEN] bit. The SCLTIMEOUT can be used with the SMBus.

Diagram



Fields

Field	Function
31-16 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
15-4 TO	Time-out Time Value Specifies the time-out interval value in increments of 16 I2C function clocks, as defined by the CLKDIV register. To change this value while I2C is in operation, disable all time-outs, write a new value to TIMEOUT, then re-enable time-outs. 0x000: A time-out will occur after 16 counts of the I2C function clock. 0x001: A time-out will occur after 32 counts of the I2C function clock. ..... 0xFFF: A time-out will occur after 65,536 counts of the I2C function clock.
3-0 TOMIN	Time-out Time Value Time-out time value, bottom four bits. These are hard-wired to 0xF. This gives a minimum time-out of 16 I2C function clocks and also a time-out resolution of 16 I2C function clocks.

15.1.7 Clock Pre-divider Register (CLKDIV)

Offset

Register	Offset
CLKDIV	14h

Function

The CLKDIV register divides down the I2C clock (I2CCLK) to produce the I2C function clock that is used to time various aspects of the I2C interface. The I2C function clock is used for some internal operations in the I2C interface and to generate the timing required by the I2C-bus specification, some of which are user configured in the MSTTIME register for Master operation. Slave operation uses CLKDIV for some timing functions.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DIVVAL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-16 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
15-0 DIVVAL	Divider Value This field controls how the I2C clock (I2CCLK) is used by the I2C functions that need an internal clock in order to operate. I2C block should be configured for 8MHz clock, this will limit SCL master clock range from 444kHz to 2MHz. 0000000000000000b - I2CCLK is used directly by the I2C. 0000000000000001b - I2CCLK is divided by 2 before use. 0000000000000010b - I2CCLK is divided by 3 before use. ..... 1111111111111111b - I2CCLK is divided by 65,536 before use.

15.1.8 Interrupt Status Register (INTSTAT)

Offset

Register	Offset
INTSTAT	18h

Function

The INTSTAT register provides a view of those interrupt flags that are currently enabled. This can simplify software handling of interrupts. See STAT for detailed descriptions of the interrupt flags.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved						SCLTIME...	EVTTIME...	Reserved				MONIDLE	Reserved	MONOV	MONRDY
W																
Reset	u	u	u	u	u	u	0	0	u	u	u	u	0	u	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SLVDESEL	Reserved			SLVNOTS...	Reserved		SLVPEND...	Reserved	MSTSTST...	Reserved	MSTARBL...	Reserved			MSTPEND...
W																
Reset	0	u	u	u	0	u	u	0	u	0	u	0	u	u	u	0

Fields

Field	Function
31-26 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
25 SCLTIMEOUT	SCL Time-out Interrupt
24 EVTTIMEOUT	Event Time-out Interrupt
23-20 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
19 MONIDLE	Monitor Idle Interrupt
18 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
17 MONOV	Monitor Overrun Interrupt
16 MONRDY	Monitor data Ready Interrupt
15 SLVDESEL	Slave Deselect Interrupt
14-12 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
11 SLVNOTSTR	Slave Not Stretching Interrupt
10-9 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
8 SLVPENDING	Slave Pending Interrupt

Table continues on the next page...

Table continued from the previous page...

Field	Function
7 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
6 MSTSTSTPERR	Master Start/Stop Error Interrupt
5 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
4 MSTARBLOSS	Master Arbitration Loss Interrupt
3-1 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
0 MSTPENDING	Master Pending Interrupt

15.1.9 Master Control Register (MSTCTL)

Offset

Register	Offset
MSTCTL	20h

Function

The MSTCTL register contains bits that control various functions of the I2C Master interface. This register is write only when the master is pending (STAT[MSTPENDING] = 1).

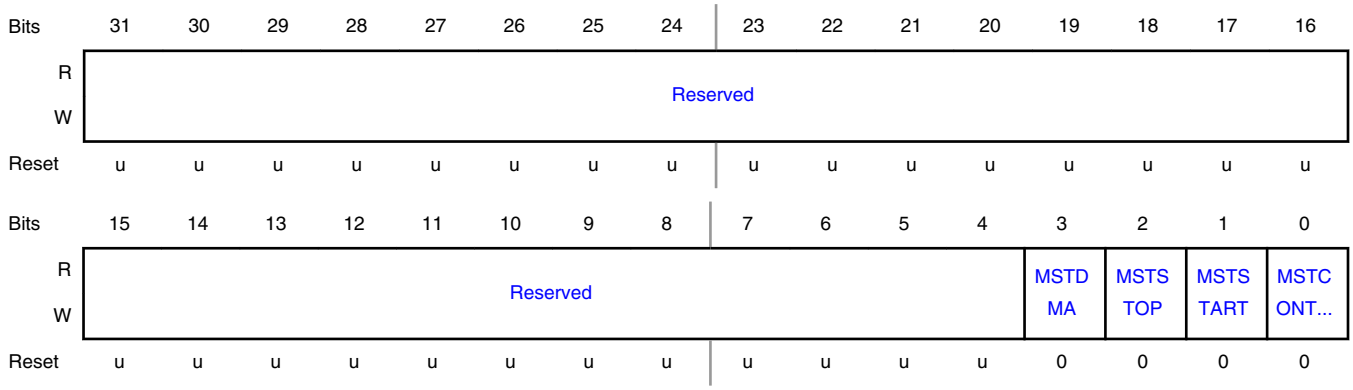
Software should always write a complete value to MSTCTL, and not OR new control bits into the register as is possible in other registers such as CFG. This is due to the fact that MSTSTART and MSTSTOP are not self clearing flags. ORing in new data following a Start or Stop may cause undesirable side effects.

After an initial I2C Start, MSTCTL should generally only be written when the STAT[MSTPENDING] flag is set, after the last bus operation has completed. An exception is when DMA is being used and a transfer completes. In this case there is no MSTPENDING flag, and the MSTDMA control bit would be cleared by software potentially at the same time as setting either the MSTSTOP or MSTSTART control bit.

NOTE

When in the idle or slave NACKed states, set the MSTDMA bit either with or after the MSTCONTINUE bit. MSTDMA can be cleared at any time

Diagram



Fields

Field	Function
31-4 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
3 MSTDMA	Master DMA Enable Master DMA enable. Data operations of the I2C can be performed with DMA. Protocol type operations such as Start, address, Stop, and address match must always be done with software, typically via an interrupt. Address acknowledgement must also be done by software except when the I2C is configured to be HSCAPABLE (and address acknowledgement is handled entirely by hardware) or when Automatic Operation is enabled. When a DMA data transfer is complete, MSTDMA must be cleared prior to beginning the next operation, typically a Start or Stop.This bit is read/write. 0b - Disable. No DMA requests are generated for master operation. 1b - Enable. A DMA request is generated for I2C master data operations. When this I2C master is generating Acknowledge bits in Master Receiver mode, the acknowledge is generated automatically.
2 MSTSTOP	Master Stop Control 0b - No effect. 1b - Stop. A stop will be generated on the I2C bus at the next allowed time, preceded by a NACK to the slave if the master is receiving data from the salve (Master Receiver mode).
1 MSTSTART	Master Start Control 0b - No effect. 1b - Start. A start will be generated on the I2C bus at the next allowed time.
0 MSTCONTINUE	Master Continue 0b - No effect. 1b - Continue. Informs the Master function to continue to the next operation. This must be done after writing transmit data, reading received data, or other housekeeping related to the next bus operation.

15.1.10 Master Timing Configuration Register (MSTTIME)

Offset

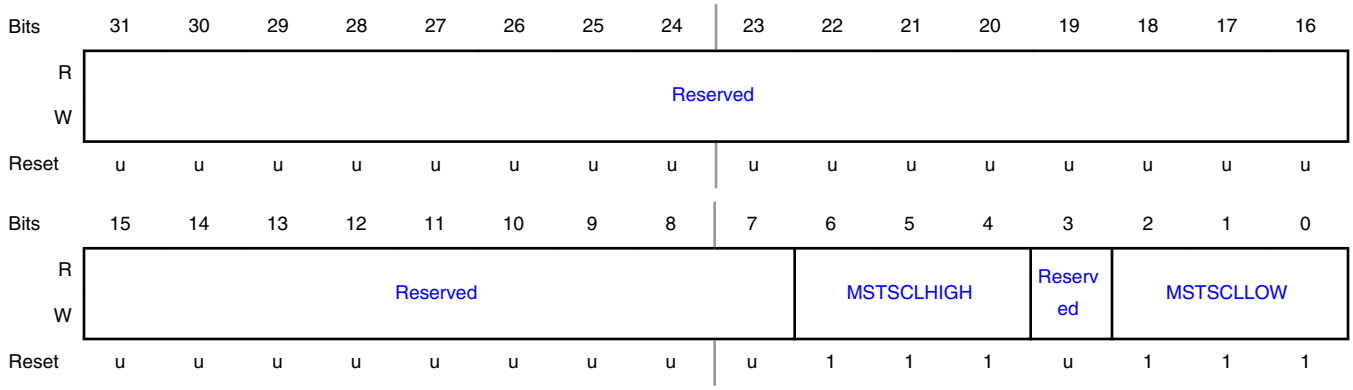
Register	Offset
MSTTIME	24h

Function

The MSTTIME register allows programming of certain times that may be controlled by the Master function. These include the clock (SCL) high and low times, repeated Start setup time, and transmitted data setup time.

The I2C clock pre-divider is described in CLKDIV.

Diagram



Fields

Field	Function
31-7 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
6-4 MSTSCLHIGH	Master SCL High Time Specifies the minimum high time that will be asserted by this master on SCL. SCL High time multiplier = (MSTSCLHIGH +2). Other masters in a multi-master system could shorten this time. This corresponds to the parameter t_{HIGH} in the I2C bus specification. I2C bus specification parameters $t_{SU;STO}$ and $t_{HD;STA}$ have the same values and are also controlled by MSTSCLHIGH.
3 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
2-0 MSTSCLLOW	Master SCL Low Time Specifies the minimum low time that will be asserted by this master on SCL. SCL Low time multiplier = (MSTSCLLOW +2). Other devices on the bus (masters or slaves) could lengthen this time. This corresponds to the parameter t_{LOW} in the I2C bus specification. I2C bus specification parameters t_{BUF} and $t_{SU;STA}$ have the same values and are also controlled by MSTSCLLOW. The minimum SCL low time is (MSTSCLLOW + 2) clocks of the I2C clock pre-divider.

15.1.11 Combined Master Receiver and Transmitter Data Register (MSTDAT)

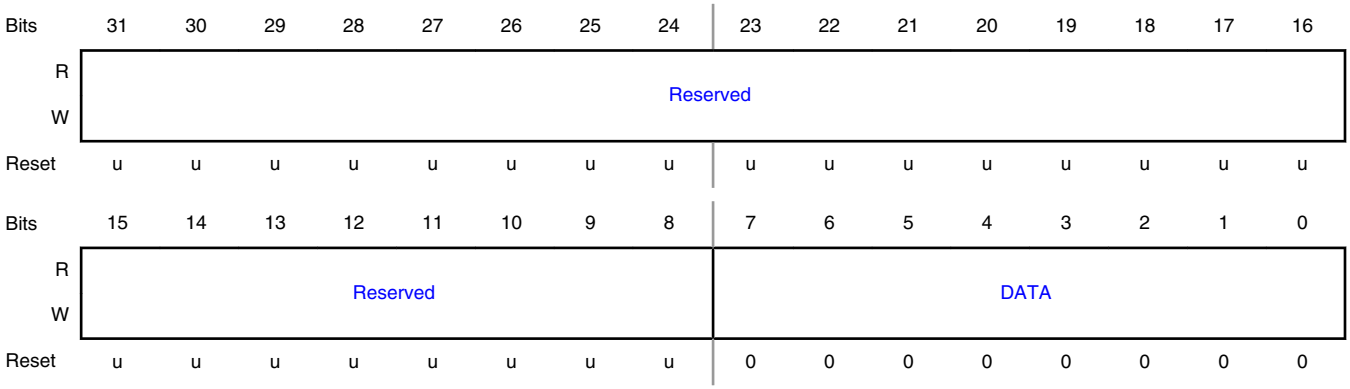
Offset

Register	Offset
MSTDAT	28h

Function

The MSTDAT register provides the means to read the most recently received data for the Master function, and to transmit data using the Master function.

Diagram



Fields

Field	Function
31-8 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
7-0 DATA	Master Function Data Read: read the most recently received data for the Master function. Write: write the transmit data for the Master Function. If the protocol requires address data and the read/write control bit then [7:1]=address, [0] = read (not write)

15.1.12 Slave Control Register (SLVCTL)

Offset

Register	Offset
SLVCTL	40h

Function

The SLVCTL register contains bits that control various functions of the I2C Slave interface.

Only write to this register when the slave is pending (STAT[SLVPENDING] = 1).

NOTE

When in the slave address state (STAT[SLVSTATE] =00), set the SLVDMA bit either with or after the SLVCONTINUE bit. SLVDMA can be cleared at any time.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved						AUTOMAT...	AUTOACK	Reserved				SLVDMA	Reserved	SLVNACK	SLVCONT...
Reset	u	u	u	u	u	u	0	0	u	u	u	u	0	u	0	0

Fields

Field	Function
31-10 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
9 AUTOMATCHREAD	Auto Match Read Write When AUTOACK is set, this bit controls whether it matches a read or write request on the next header with an address matching SLVADR0. Since DMA needs to be configured to match the transfer direction, the direction needs to be specified. This bit allows a direction to be chosen for the next operation. 0b - The expected next operation in Automatic Mode is an I2C write. 1b - The expected next operation in Automatic Mode is an I2C read.
8 AUTOACK	Automatic Acknowledge When this bit is set, it will cause an I2C header which matches SLVADR0 and the direction set by AUTOMATCHREAD to be ACKed immediately; this is used with DMA to allow processing of the data without intervention. If this bit is clear and a header matches SLVADR0, the behavior is controlled by AUTONACK in the SLVADR0 register: allowing NACK or interrupt. 0b - Normal, non-automatic operation. If AUTONACK = 0, an SlvPending interrupt is generated when a matching address is received. If AUTONACK-1, receiver addresses are NACKed (ignored). 1b - A header with matching SLVADR0 and matching direction as set by AUTOMATCHREAD will be ACKed immediately, allowing the master to move on to the data bytes. If the address matches SLVADR0, but the direction does not match AUTOMATCHREAD, the behaviour will depend on the AUTONACK bit in the SLVADR0 register: if AUTONACK is set, then it will be Nacked; else if AUTONACK is cl;ear, then a SlvPending interrupt is geernated.
7-4 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
3 SLVDMA	Slave DMA Enable 0b - Disabled. No DMA requests are issued for Slave mode operation. 1b - Enabled. DMA requests are issued for I2C slave data transmission and reception.
2 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
1 SLVNACK	Slave NACK. Slave NACK. 0: No effect. 1: NACK. Causes the Slave function to NACK the master when the slave is receiving data from the master (Slave Receiver mode).
0 SLVCONTINUE	Slave Continue 0b - No effect. 1b - Continue. Informs the Slave function to continue to the next operation, by clearing the SLVPENDING flag in the STAT register. This must be done after writing transmit data, reading recevied data, or any other housekeeping related to the next bus operation. Automatic Operation has different requirements. SLVCONTINUE should not be set unless SLVPENDING=1.

15.1.13 Combined Slave Receiver and Transmitter Data Register (SLVDAT)

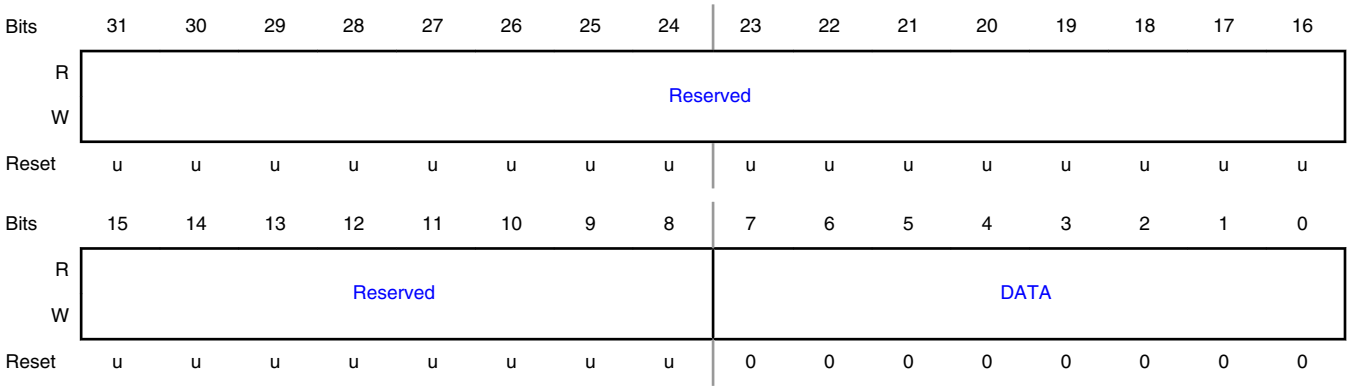
Offset

Register	Offset
SLVDAT	44h

Function

The SLVDAT register provides the means to read the most recently received data for the Slave function and to transmit data using the Slave function.

Diagram



Fields

Field	Function
31-8 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
7-0 DATA	Slave Function Data Read: read the most recently received data for the Slave function. Write: transmit data using the Slave function.

15.1.14 Slave Address 0 (SLVADR0)

Offset

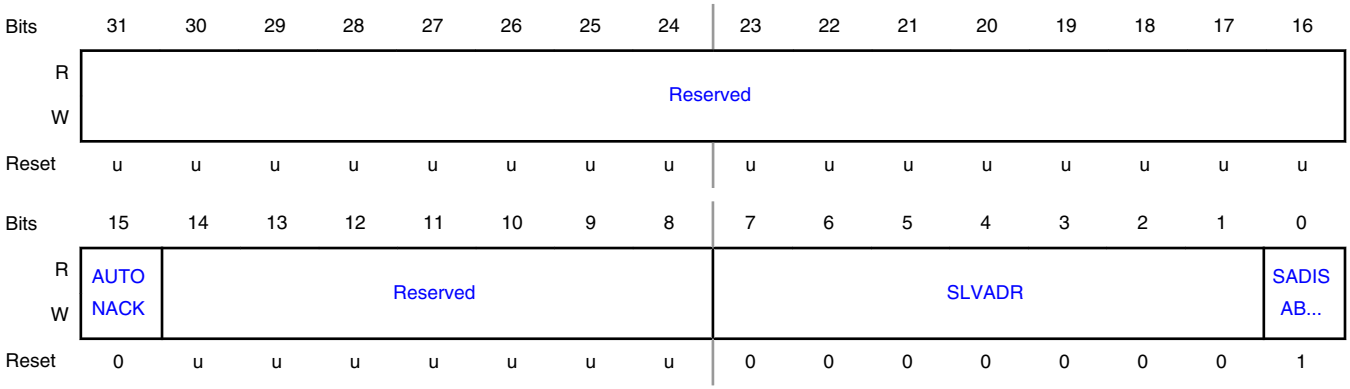
Register	Offset
SLVADR0	48h

Function

The SLVADR0 register allows enabling and defining one of the addresses that can be automatically recognized by the I2C slave hardware.

The I2C slave function has a total of 4 address comparators. The value in SLVADR0 can be qualified by the setting of the SLVQUAL0 register. The additional 3 address comparators do not include the address qualifier feature. For handling of the general call address, one of the 4 address registers can be programmed to respond to address 0.

Diagram



Fields

Field	Function
31-16 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
15 AUTONACK	Automatic NACK Operation Used in conjunction with AUTOACK and AUTOMATCHREAD, allows software to ignore I2C traffic while handling previous I2C data or other operations. 0b - Normal operation, matching I2C addresses are not ignored. 1b - Automatic-only mode. If AUTOACK is not set, all incoming I2C addresses are ignored.
14-8 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
7-1 SLVADR	Slave Address Seven bit slave address that is compared to received addresses if enabled. The compare can be affected by the setting of the SLVQUAL0 register.
0 SADISABLE	Slave Address 0 Disable 0b - Slave Address 0 is enabled. 1b - Slave Address 0 is ignored.

15.1.15 Slave Address 1 (SLVADR1)

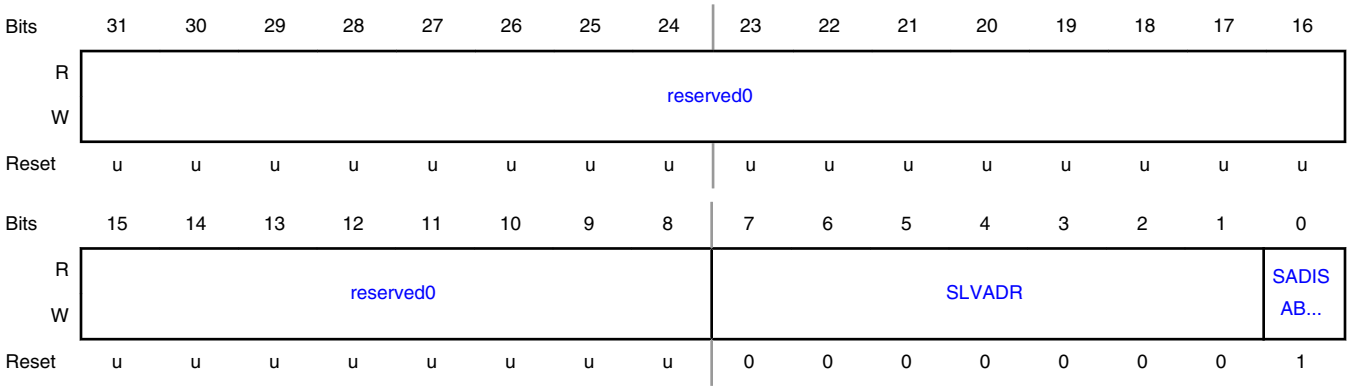
Offset

Register	Offset
SLVADR1	4Ch

Function

Slave address register provides for an additional addresses that can be automatically recognized by the I2C slave hardware.

Diagram



Fields

Field	Function
31-8 reserved0	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
7-1 SLVADR	Slave Address 1 Seven bit slave address that is compared to received addresses if enabled.
0 SADISABLE	Slave Address 1 Disable 0b - Slave Address 1 is enabled. 1b - Slave Address 1 is ignored.

15.1.16 Slave Address 2 (SLVADR2)

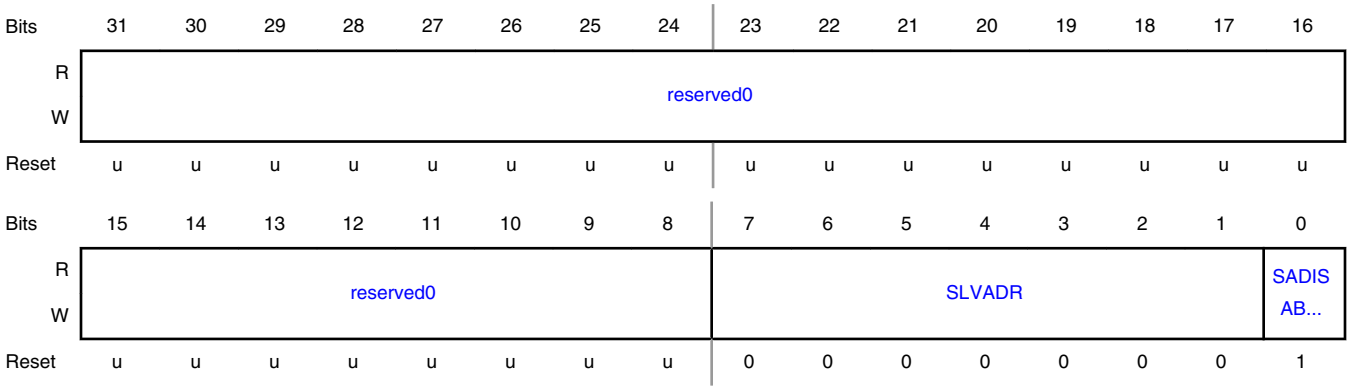
Offset

Register	Offset
SLVADR2	50h

Function

Slave address register provides for an additional addresses that can be automatically recognized by the I2C slave hardware.

Diagram



Fields

Field	Function
31-8 reserved0	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
7-1 SLVADR	Slave Address 2 Seven bit slave address that is compared to received addresses if enabled.
0 SADISABLE	Slave Address 2 Disable 0b - Slave Address 2 is enabled. 1b - Slave Address 2 is ignored.

15.1.17 Slave Address 3 (SLVADR3)

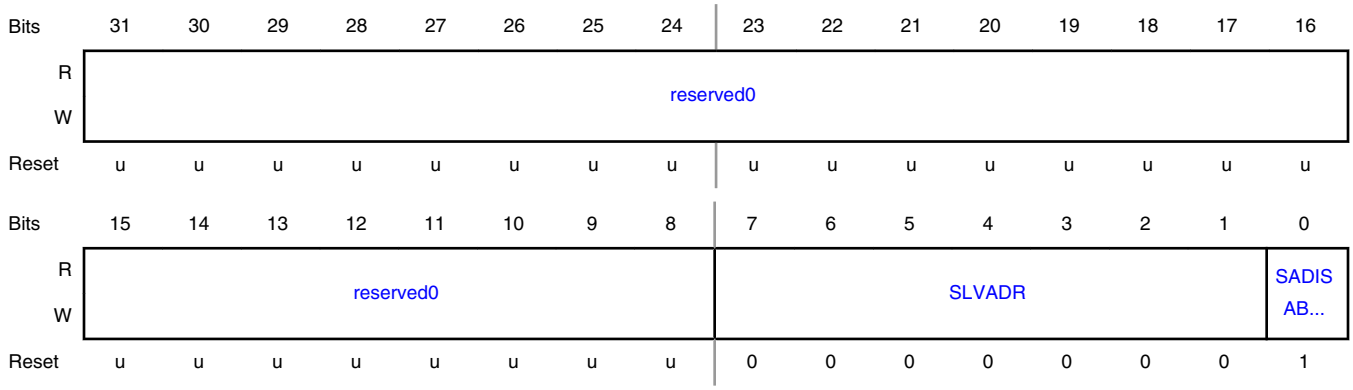
Offset

Register	Offset
SLVADR3	54h

Function

Slave address register provides for an additional addresses that can be automatically recognized by the I2C slave hardware.

Diagram



Fields

Field	Function
31-8 reserved0	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
7-1 SLVADR	Slave Address 3 Seven bit slave address that is compared to received addresses if enabled.
0 SADISABLE	Slave Address 3 Disable 0b - Slave Address 3 is enabled. 1b - Slave Address 3 is ignored.

15.1.18 Slave Qualification for Address 0 (SLVQUAL0)

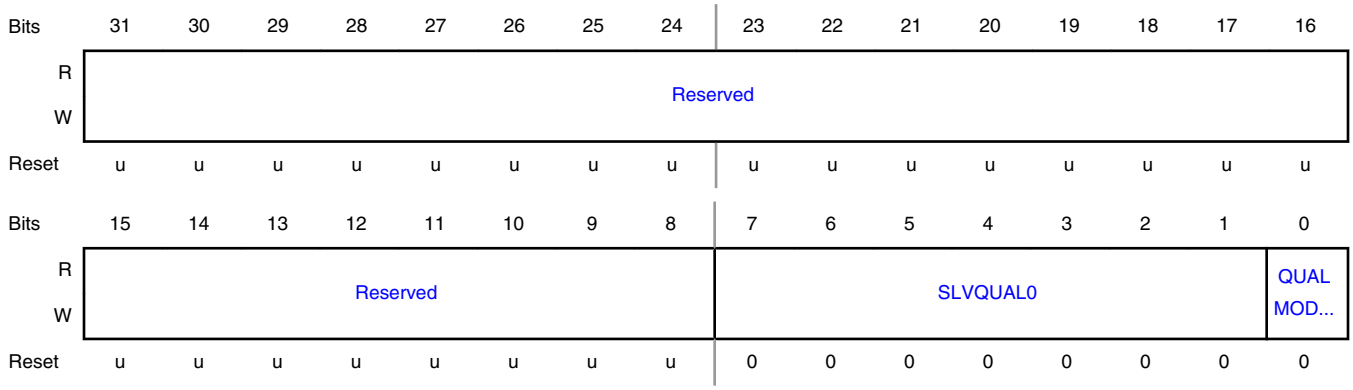
Offset

Register	Offset
SLVQUAL0	58h

Function

The SLVQUAL0 register can alter how Slave Address 0 (specified by the SLVADR0 register) is interpreted.

Diagram



Fields

Field	Function
31-8 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
7-1 SLVQUAL0	Slave Address Qualifier for Address 0 A value of 0 causes the address in SLVADR0 to be used as-is, assuming that it is enabled. If QUALMODE0 = 0, any bit in this field which is set to 1 will cause an automatic match of the corresponding bit of the received address when it is compared to the SLVADR0 register. If QUALMODE0 = 1, an address range is matched for address 0. This range extends from the value defined by SLVADR0 to the address defined by SLVQUAL0 (address matches when SLVADR0[7:1] received address SLVQUAL0[7:1]).
0 QUALMODE0	Qualify Mode for Slave Address 0 0b - Mask. The SLVQUAL0 field is used as a logical mask for matching address 0. 1b - Extend. The SLVQAL0 field is used to extend address 0 matching in a range of addresses.

15.1.19 Monitor Receiver Data Register (MONRXDAT)

Offset

Register	Offset
MONRXDAT	80h

Function

The read-only MONRXDAT register provides information about events on the I2C-bus, primarily to facilitate debugging of the I2C during application development. All data addresses and data passing on the bus and whether these were acknowledged, as well as Start and Stop events, are reported.

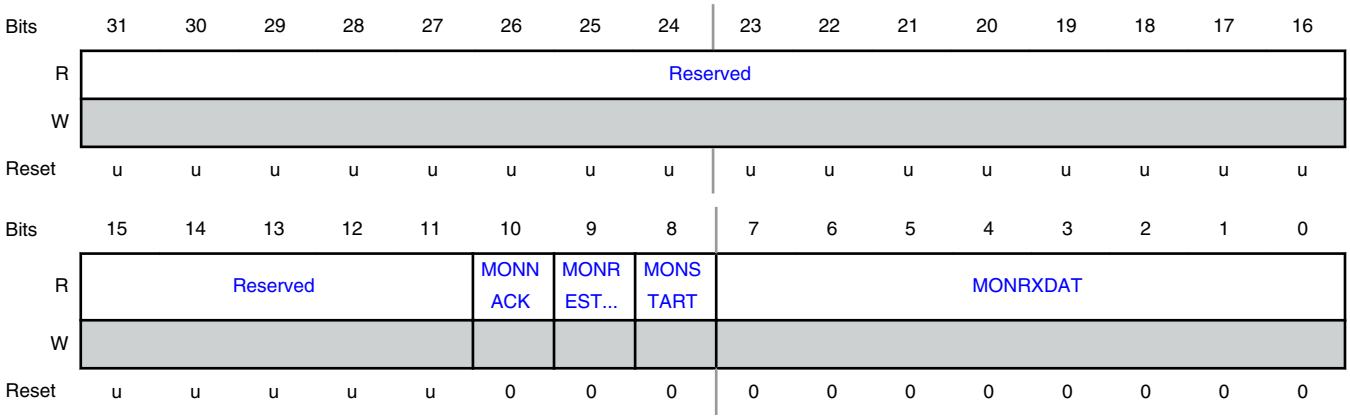
The Monitor function must be enabled by the CFG[MONEN] bit. Monitor mode can be configured to stretch the I2C clock if data is not read from the MONRXDAT register in time to prevent it, via the CFG[MONCLKSTR] bit. This can help ensure that nothing is missed but can cause the Monitor function to be somewhat intrusive (by potentially adding clock delays, depending on software

or DMA response time). In order to improve the chance of collecting all Monitor information if clock stretching is not enabled, Monitor data is buffered such that it is available until the end of the next piece of information from the I2C-bus.

NOTE

The details of clock stretching are different in HS mode compared to the slower modes.

Diagram



Fields

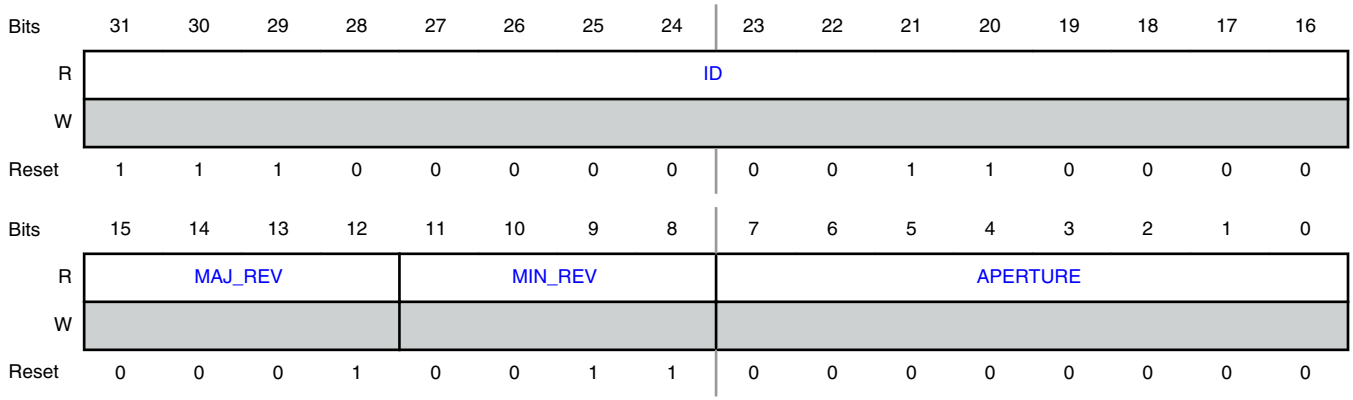
Field	Function
31-11 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
10 MONNACK	Monitor Received NACK 0b - Acknowledged. The data currently being provided by the Monitor function was acknowledged by at least one master or slave recevier. 1b - Not Acknowledged. The data currently being provided by the Monitor function was not acknowledged by any receiver.
9 MONRESTART	Monitor Received Repeated Start 0b - No repeated start detected. The Monitor function has not detected a Repeated Start event on the I2C bus. 1b - Repeated start detected. The Monitor function has detected a Repeated Start event on the I2C bus.
8 MONSTART	Monitor Received Start 0b - No start detected. The monitor function has not detected a Start event on the I2C bus. 1b - Start detected. The Monitor function has detected a Start event on the I2C bus.
7-0 MONRXDAT	Monitor Function Receiver Data This reflects every data byte that passes on the I2C pins.

15.1.20 I2C Module Identifier (ID)

Offset

Register	Offset
ID	FFCh

Diagram



Fields

Field	Function
31-16 ID	Identifier This is the unique identifier of the module
15-12 MAJ_REV	Major Revision There may be software incompatability between major revisions.
11-8 MIN_REV	Minor Revision Minor revision with no software consequences
7-0 APERTURE	Aperture Aperture number minus 1 of consecutive packets 4 Kbytes reserved for this IP

Chapter 16
Digital Microphone Interface (DMIC)

16.1 DMIC register descriptions

16.1.1 DMIC memory map

DMIC base address: 4008_A000h

Offset	Register	Width (In bits)	Access	Reset value
20h	Oversample Rate Register 0 (OSR0)	32	RW	See description
24h	DMIC Clock Register 0 (DIVHFCLK0)	32	RW	See description
28h	Pre-Emphasis Filter Coefficient for 2 FS Register 0 (PREAC2FS COEF0)	32	RW	See description
2Ch	Pre-Emphasis Filter Coefficient for 4 FS Register 0 (PREAC4FS COEF0)	32	RW	See description
30h	Decimator Gain Shift Register 0 (GAINSHIFT0)	32	RW	See description
A0h	FIFO Control Register 0 (FIFO_CTRL0)	32	RW	See description
A4h	FIFO Status Register 0 (FIFO_STATUS0)	32	RW	See description
A8h	FIFO Data Register 0 (FIFO_DATA0)	32	RW	See description
ACh	PDM Source Configuration Register 0 (PHY_CTRL0)	32	RW	See description
B0h	DC Control Register 0 (DC_CTRL0)	32	RW	See description
120h	Oversample Rate Register 1 (OSR1)	32	RW	See description
124h	DMIC Clock Register 1 (DIVHFCLK1)	32	RW	See description
128h	Pre-Emphasis Filter Coefficient for 2 FS Register 1 (PREAC2FS COEF1)	32	RW	See description
12Ch	Pre-Emphasis Filter Coefficient for 4 FS Register 1 (PREAC4FS COEF1)	32	RW	See description

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
130h	Decimator Gain Shift Register 1 (GAINSHIFT1)	32	RW	See description
1A0h	FIFO Control Register 1 (FIFO_CTRL1)	32	RW	See description
1A4h	FIFO Status Register 1 (FIFO_STATUS1)	32	RW	See description
1A8h	FIFO Data Register 1 (FIFO_DATA1)	32	RW	See description
1ACh	PDM Source Configuration Register 1 (PHY_CTRL1)	32	RW	See description
1B0h	DC Control Register 1 (DC_CTRL1)	32	RW	See description
F00h	Channel Enable Register (CHANEN)	32	RW	See description
F0Ch	I/O Configuration Register (IOCFG)	32	RW	See description
F10h	Use 2FS Register (USE2FS)	32	RW	See description
F80h	HWVAD Input Gain (HWVADGAIN)	32	RW	See description
F84h	HWVAD Filter Control Register (HWVADHPFS)	32	RW	See description
F88h	HWVAD Control Register (HWVADST10)	32	RW	See description
F8Ch	HWVAD Filter Reset Register (HWVADRSTT)	32	RW	See description
F90h	HWVAD Noise Estimator Gain Register (HWVADTHGN)	32	RW	See description
F94h	HWVAD Signal Estimator Gain Register (HWVADTHGS)	32	RW	See description
F98h	HWVAD Noise Envelope Estimator Register (HWVADLOWZ)	32	RO	See description
FFCh	Module Identification Register (ID)	32	RO	0000_0002h

16.1.2 Oversample Rate Register a (OSR0 - OSR1)

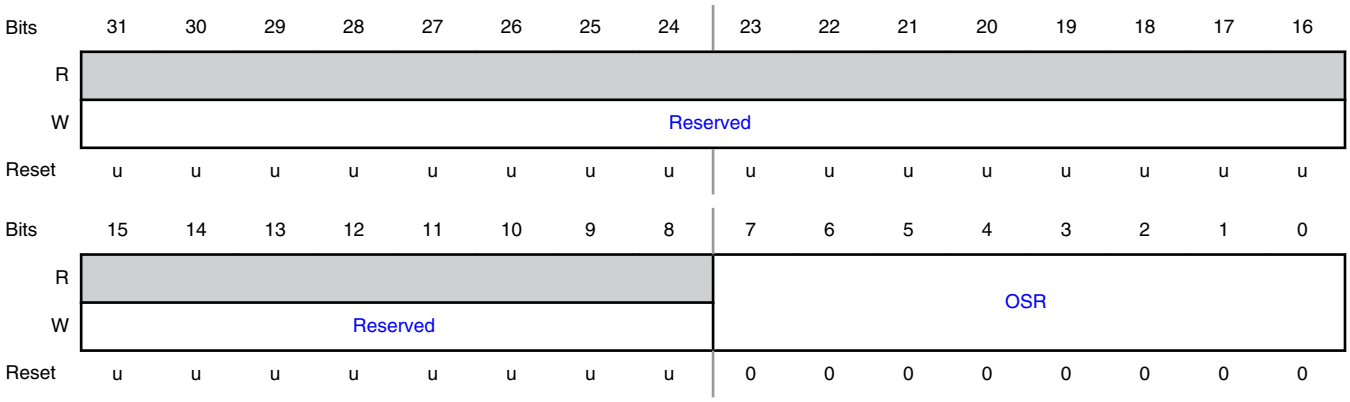
Offset

Register	Offset
OSR0	20h
OSR1	120h

Function

This register selects the oversample rate (CIC decimation rate) for the input channel.

Diagram



Fields

Field	Function
31-8 —	Reserved Reserved. Read value is undefined, only zero should be written.
7-0 OSR	Oversample Rate Selection Selects the oversample rate for the related input channel.

16.1.3 DMIC Clock Register a (DIVHFCLK0 - DIVHFCLK1)

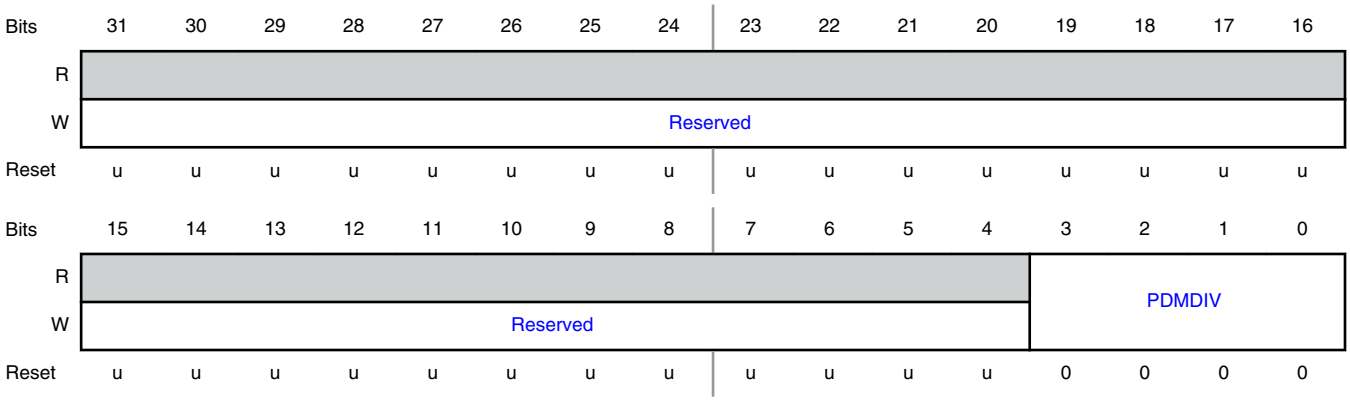
Offset

Register	Offset
DIVHFCLK0	24h
DIVHFCLK1	124h

Function

This register controls the clock pre-divider for the input channel.

Diagram



Fields

Field	Function
31-4 —	Reserved Reserved. Read value is undefined, only zero should be written.
3-0 PDMDIV	PDM Clock Divider Value 11: divide by 64; 12: divide by 96; 13: divide by 128; others = reserved. 0000b Divide by 1 0001b Divide by 2 0010b Divide by 3 0011b Divide by 4 0100b Divide by 6 0101b Divide by 8 0110b Divide by 12 0111b Divide by 16 1000b Divide by 24 1001b Divide by 32 1010b Divide by 48 1011b Divide by 64 1100b Divide by 96 1101b Divide by 128 Others Reserved

16.1.4 Pre-Emphasis Filter Coefficient for 2 FS Register a (PREAC2FSCOEF0 - PREAC2FSCOEF1)

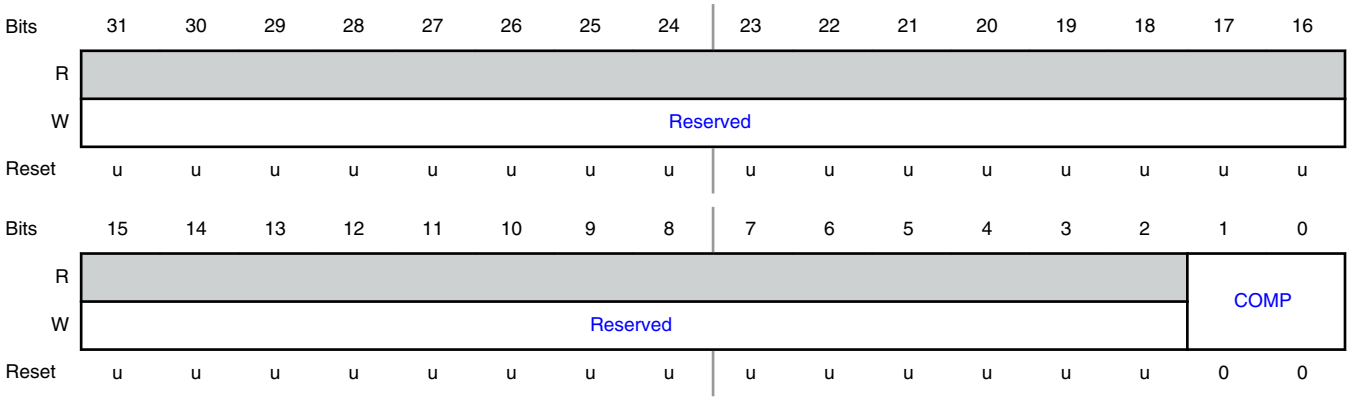
Offset

Register	Offset
PREAC2FSCOEF0	28h
PREAC2FSCOEF1	128h

Function

This register seclects the pre-emphasis filter coeffcient for the input channel when 2 FS mode is used.

Diagram



Fields

Field	Function
31-2 —	Reserved Reserved. Read value is undefined, only zero should be written.
1-0 COMP	Pre-emphasis Filer Coefficient for 2 FS Mode 00b - Compensation = 0 01b - Compensation = -0.16 10b - Compensation = -0.15 11b - Compensation = -0.13

16.1.5 Pre-Emphasis Filter Coefficient for 4 FS Register a (PREAC4FSCOEF0 - PREAC4FSCOEF1)

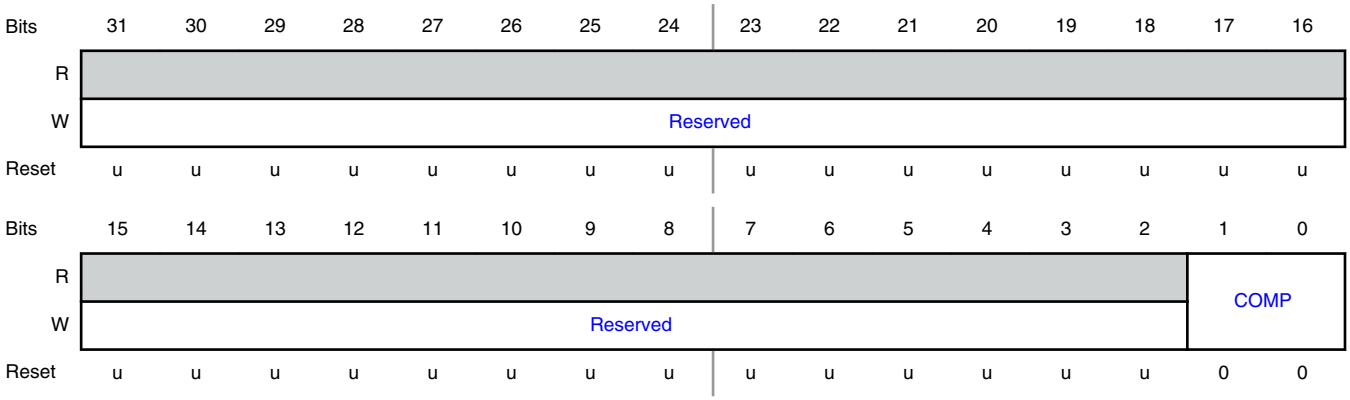
Offset

Register	Offset
PREAC4FSCOEF0	2Ch
PREAC4FSCOEF1	12Ch

Function

This register seclects the pre-emphasis filter coeffcient for the input channel when 4FS mode is used.

Diagram



Fields

Field	Function
31-2 —	Reserved Reserved. Read value is undefined, only zero should be written.
1-0 COMP	Pre-emphasis Filer Coefficient for 4 FS Mode 00b - Compensation = 0 01b - Compensation = -0.16 10b - Compensation = -0.15 11b - Compensation = -0.13

16.1.6 Decimator Gain Shift Register a (GAINSHIFT0 - GAINSHIF T1)

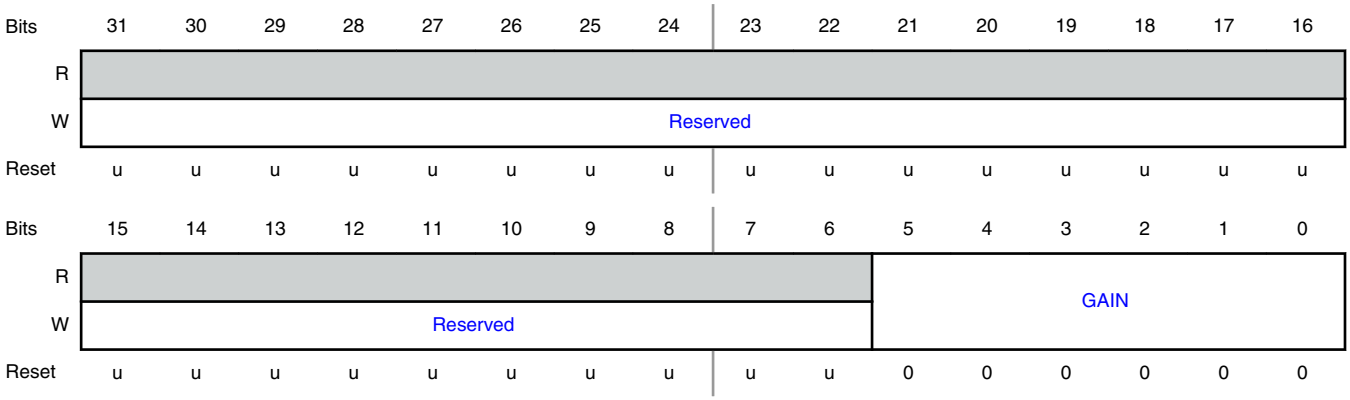
Offset

Register	Offset
GAINSHIFT0	30h
GAINSHIFT1	130h

Function

This register adjusts the gain of the 4FS PCM data from the input filter.

Diagram



Fields

Field	Function
31-6 —	Reserved Reserved. Read value is undefined, only zero should be written.
5-0 GAIN	Gain Control Gain control, as a positive or negative (two's complement) number of bits to shift.

16.1.7 FIFO Control Register a (FIFO_CTRL0 - FIFO_CTRL1)

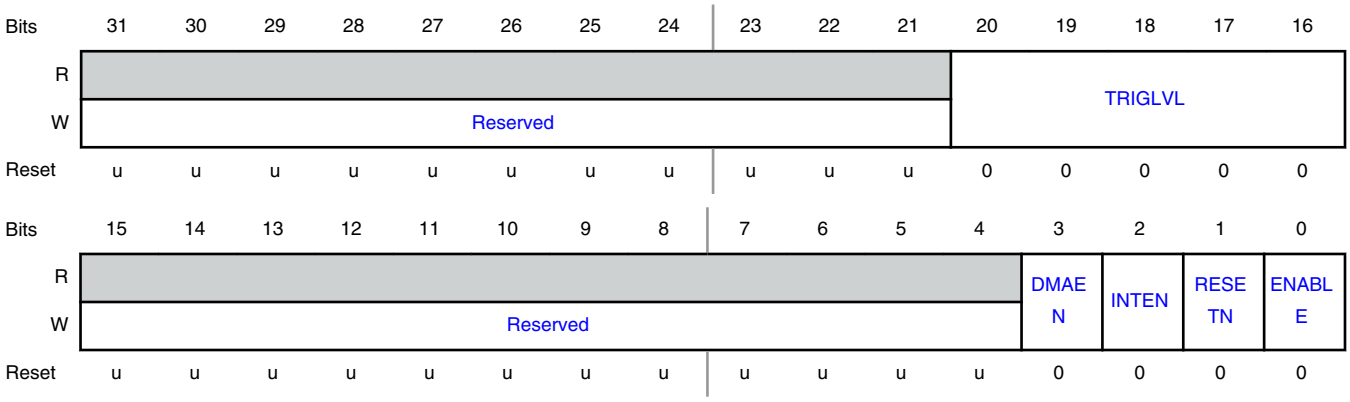
Offset

Register	Offset
FIFO_CTRL0	A0h
FIFO_CTRL1	1A0h

Function

This register configures FIFO usage.

Diagram



Fields

Field	Function
31-21 —	Reserved Reserved. Read value is undefined, only zero should be written.
20-16 TRIGLVL	FIFO Trigger Level Selects the data trigger level for interrupt or DMA operation. If enabled to do so, the FIFO level can wake up the device. 00000b Trigger when the FIFO has received one entry (is no longer empty). 00001b Trigger when the FIFO has received two entries. ..... 01111b trigger when the FIFO has received 16 entries (has become full).
15-4 —	Reserved Reserved. Read value is undefined, only zero should be written.
3 DMAEN	DMA Enable 0b - DMA requests are not enabled. 1b - DMA requests based on FIFO level are enabled.
2 INTEN	Interrupt Enable 0b - FIFO level interrupts are not enabled. 1b - FIFO level interrupts are enabled.

Table continues on the next page...

Table continued from the previous page...

Field	Function
1 RESETN	FIFO Reset This bit must be cleared before resuming operation. 0b - Reset the FIFO. 1b - Normal operation.
0 ENABLE	FIFO Enable 0b - FIFO is not enabled. Enabling a DMIC channel with the FIFO disabled could be useful in order to avoid a filter settling. delay when a channel is re-enabled after a period when the data was not needed. 1b - FIFO is enabled. The FIFO must be enabled in order for the CPU or DMA to read data from the DMIC via the FIFO_DATA register.

16.1.8 FIFO Status Register a (FIFO_STATUS0 - FIFO_STATUS1)

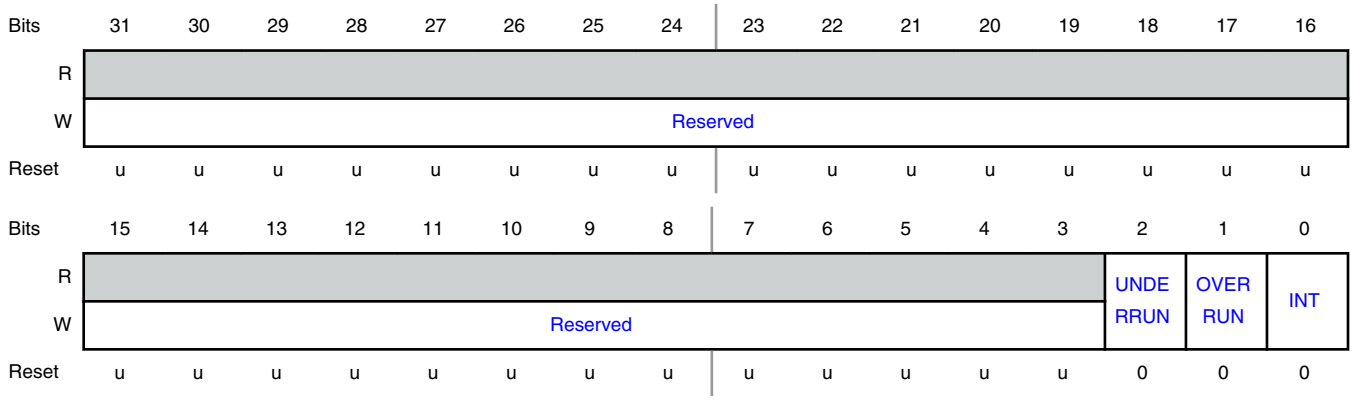
Offset

Register	Offset
FIFO_STATUS0	A4h
FIFO_STATUS1	1A4h

Function

This register provides status information for the FIFO and also indicates an interrupt from the peripheral funcion.

Diagram



Fields

Field	Function
31-3 —	Reserved Reserved. Read value is undefined, only zero should be written.
2 UNDERRUN	Underrun Flag Indicates that a FIFO underflow has occurred at some point. Writing a one to this bit clears the flag.
1 OVERRUN	Overrun Flag Indicates that a FIFO overflow has occurred at some point. Writing a one to this bit clears the flag. This flag does not cause an interrupt.
0 INT	Interrupt Flag Asserted when FIFO data reaches the level specified in the FIFO_CTRL register. Writing a one to this bit clears the flag. Remark: note that the bus clock to the DMIC subsystem must be running in order for an interrupt to occur.

16.1.9 FIFO Data Register a (FIFO_DATA0 - FIFO_DATA1)

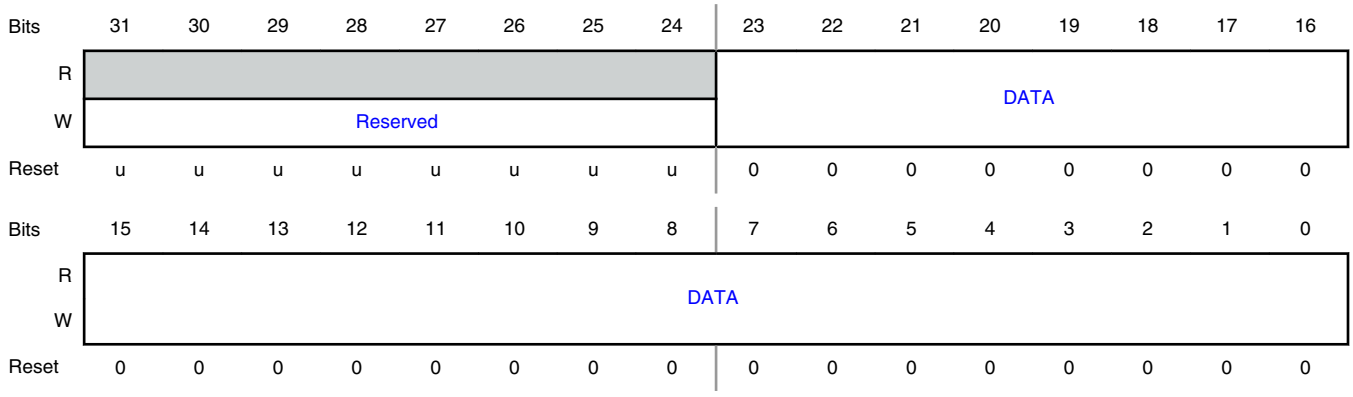
Offset

Register	Offset
FIFO_DATA0	A8h
FIFO_DATA1	1A8h

Function

This register is used to read values that have been received via the PDM stream.

Diagram



Fields

Field	Function
31-24 —	Reserved Reserved. Read value is undefined, only zero should be written.
23-0 DATA	Data from the Top of the Input Filter FIFO

16.1.10 PDM Source Configuration Register a (PHY_CTRL0 - PHY_CTRL1)

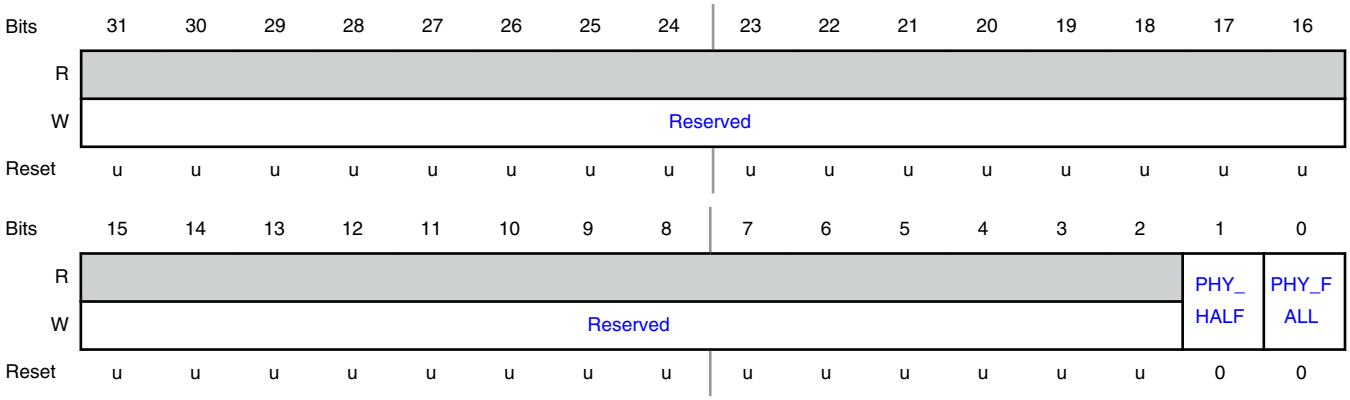
Offset

Register	Offset
PHY_CTRL0	ACh
PHY_CTRL1	1ACh

Function

This register configures how the PDM source signals are interpreted.

Diagram



Fields

Field	Function
31-2 —	Reserved Reserved. Read value is undefined, only zero should be written.

Table continues on the next page...

Table continued from the previous page...

Field	Function
1 PHY_HALF	Half Rate Sampling 0b - Standard half rate sampling. The clock to the DMIC is sent at the same rate as the decimator is providing. 1b - Use half rate sampling. The PDM clock to DMIC is divided by 2. Each PDM data is sampled twice into the decimator. The purpose of this mode is to allow slower sampling rate in quiet periods of listening for a trigger. Allowing the decimator to maintain the same decimation rate between the higher quality, higher PDM clock rate and the lower quality lower PDM clock rate means that the user can quickly switch to higher quality without re-configuring the decimator, and thus avoiding long filter settling times, when switching to higher quality (higher freq PDM clock) for recognition.
0 PHY_FALL	Capture PDM_DATA 0b - Capture PDM_DATA on the rising edge of PDM_CLK. 1b - Capture PDM_DATA on the falling edge of PDM_CLK.

16.1.11 DC Control Register a (DC_CTRL0 - DC_CTRL1)

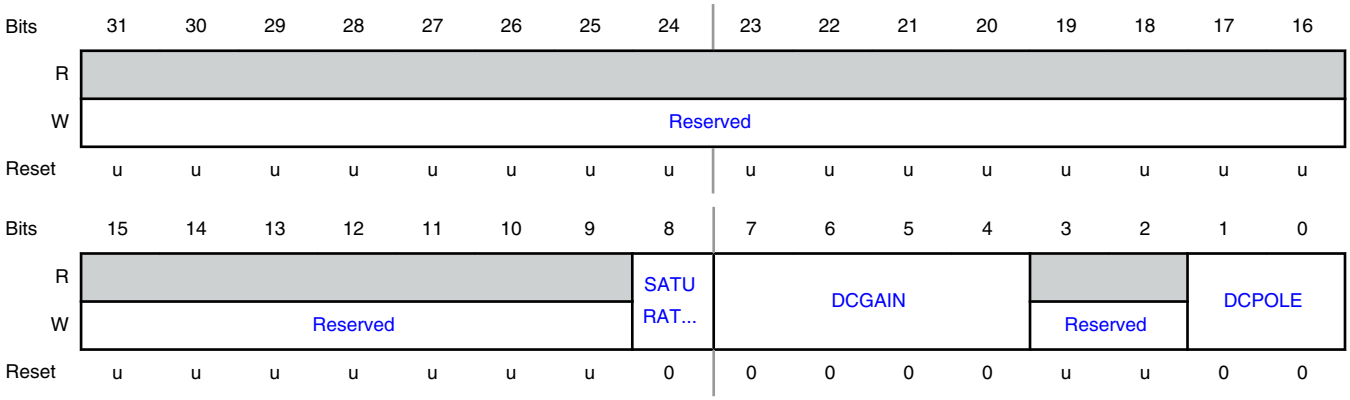
Offset

Register	Offset
DC_CTRL0	B0h
DC_CTRL1	1B0h

Function

This register controls the DC filter.

Diagram



Fields

Field	Function
31-9 —	Reserved Reserved. Read value is undefined, only zero should be written.
8 SATURATEAT16BIT	Select 16-bit Saturation 0b - Results roll over if out of range and do not saturate. 1b - If the result overflows, it saturates at 0xFFFF for positive overflow and 0x8000 for negative overflow.
7-4 DCGAIN	Fine Gain Adjustment in Form of Bits to Downshift
3-2 —	Reserved Reserved. Read value is undefined, only zero should be written.
1-0 DCPOLE	DC Block Filter These frequencies assume a PCM output frequency of 16 MHz. If the actual PCM output frequency is 8 MHz, for example, the noted frequencies would be divided by 2. 00b - Flat response, no filter. 01b - 155 Hz. 10b - 78 Hz. 11b - 39 Hz.

16.1.12 Channel Enable Register (CHANEN)

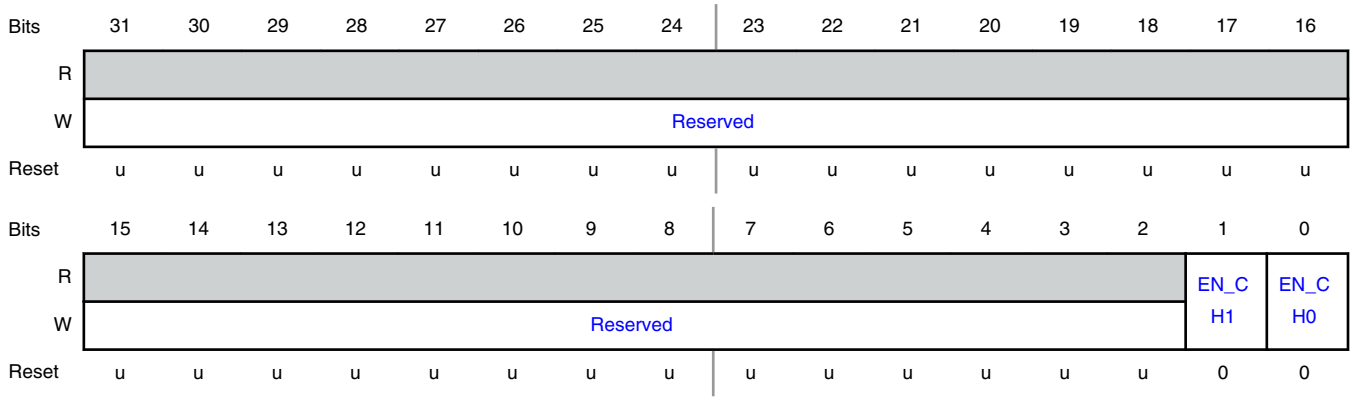
Offset

Register	Offset
CHANEN	F00h

Function

This register allows enabling either or both PDM channels.

Diagram



Fields

Field	Function
31-2 —	Reserved Reserved. Read value is undefined, only zero should be written.
1 EN_CH1	Enable Channel 1 When 1, PDM channel 1 is enabled.
0 EN_CH0	Enable Channel 0 When 1, PDM channel 0 is enabled.

16.1.13 I/O Configuration Register (IOCFG)

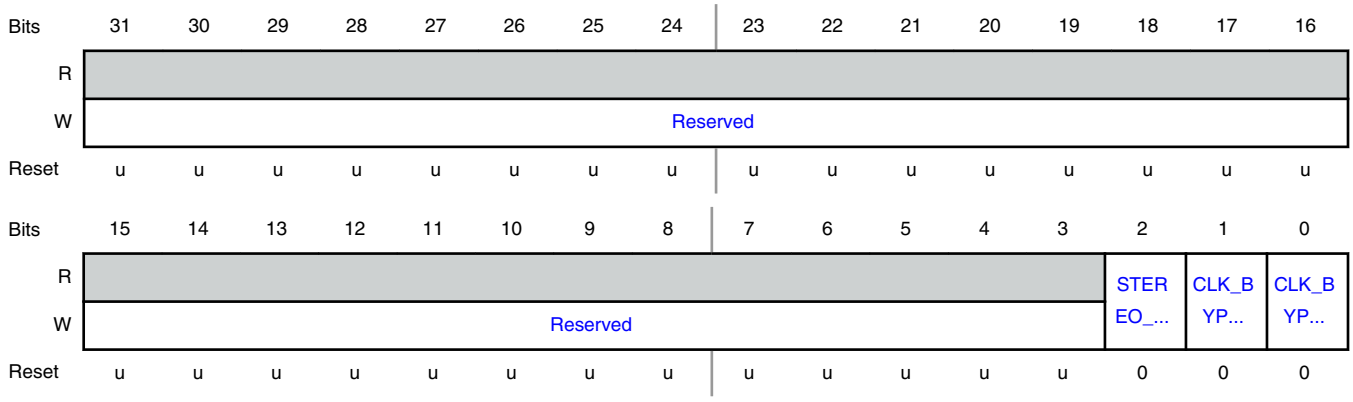
Offset

Register	Offset
IOCFG	F0Ch

Function

This register configures the use of the PDM pins.

Diagram



Fields

Field	Function
31-3 —	Reserved Reserved. Read value is undefined, only zero should be written.
2 STEREO_DATA0	Stereo PDM Select When 1, PDM_DATA0 is routed to both PDM channels in a configuration that supports a single stereo digital microphone.
1 CLK_BYPASS1	Bypass CLK1 When 1, PDM_DATA1 becomes the clock for PDM channel 1. This provides for the possibility of an external codec taking over the PDM bus.
0 CLK_BYPASS0	Bypass CLK0 When 1, PDM_DATA1 becomes the clock for PDM channel 0. This provides for the possibility of an external codec taking over the PDM bus.

16.1.14 Use 2FS Register (USE2FS)

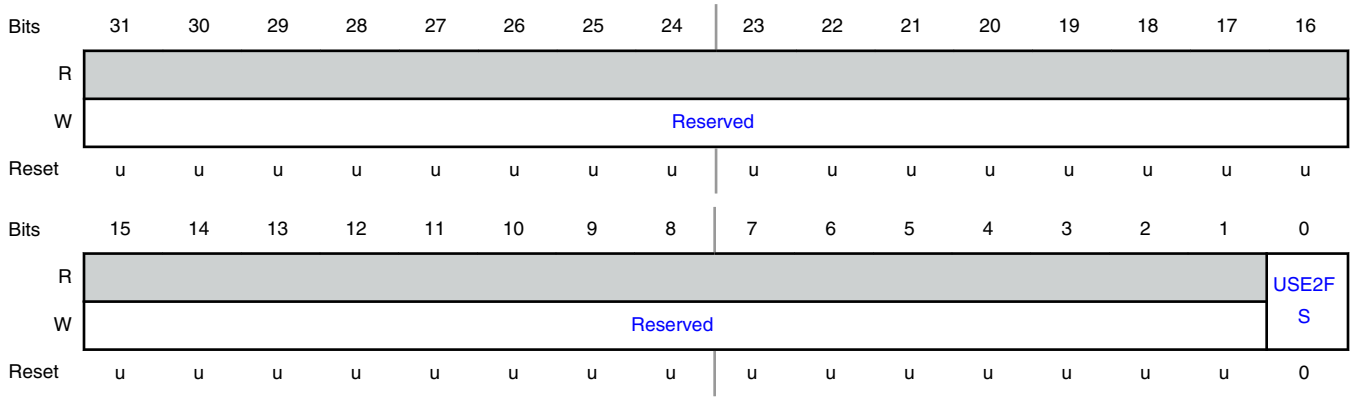
Offset

Register	Offset
USE2FS	F10h

Function

This register allow selecting 2FS output rather than 1FS output.

Diagram



Fields

Field	Function
31-1 —	Reserved Reserved. Read value is undefined, only zero should be written.
0 USE2FS	PCM Data 0b - Use 1FS output for PCM data. 1b - Use 2FS output for PCM data.

16.1.15 HWVAD Input Gain (HWVADGAIN)

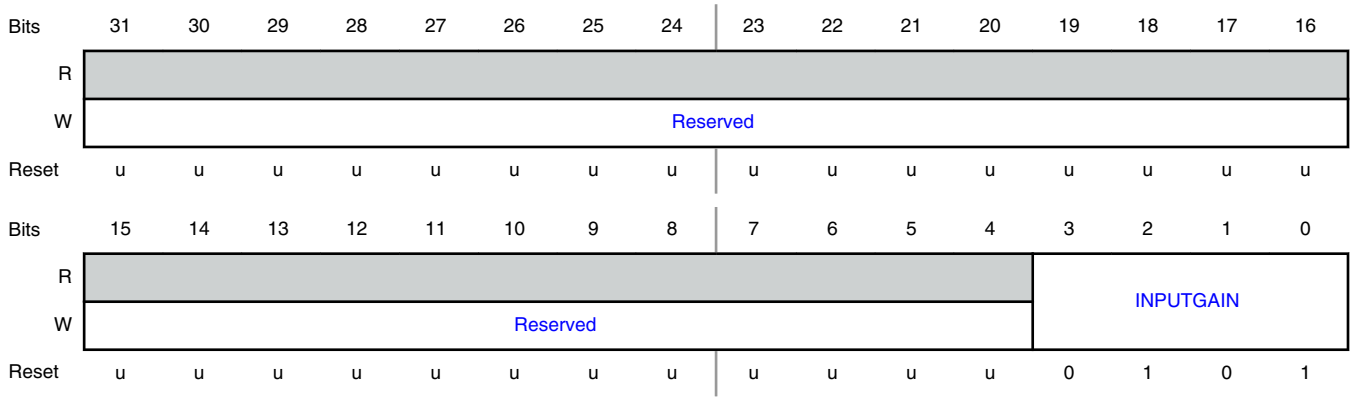
Offset

Register	Offset
HWVADGAIN	F80h

Function

This register controls the input gain of the HWVAD.

Diagram



Fields

Field	Function
31-4 —	Reserved Reserved. Read value is undefined, only zero should be written.
3-0 INPUTGAIN	Shift Value for Input Bit 0000b - -10 bits 0001b - -8 bits 0010b - -6 bits 0011b - -4 bits 0100b - -2 bits 0101b - 0 bits (default) 0110b - 2 bits 0111b - 4 bits 1000b - 6 bits 1001b - 8 bits 1010b - 10 bits 1011b - 12 bits 1100b - 14 bits 1101b - Reserved 1110b - Reserved 1111b - Reserved

16.1.16 HWVAD Filter Control Register (HWVADHPFS)

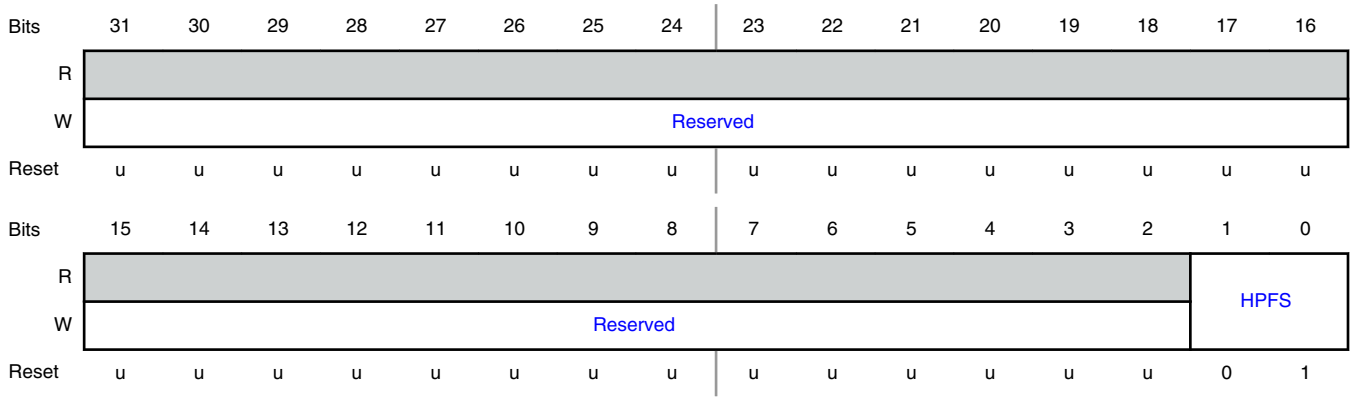
Offset

Register	Offset
HWVADHPFS	F84h

Function

This register controls the HWVAD filter setting.

Diagram



Fields

Field	Function
31-2 —	Reserved Reserved. Read value is undefined, only zero should be written.
1-0 HPFS	High Pass Filter This filter setting parameter can be used to optimize performance for different background noise situations. In order to find the best setting, software needs to perform a rough spectral analysis of the audio signal. Rule of thumb: If the amount of low-frequency content in the background noise is small, then set HPFS=0x2, otherwise use 0x1. 00b - First filter by-pass. 01b - High pass filter with -3 dB cut-off at 1750 Hz. 10b - High pass filter with -3 dB cut-off at 215 Hz. 11b - Reserved.

16.1.17 HWVAD Control Register (HWVADST10)

Offset

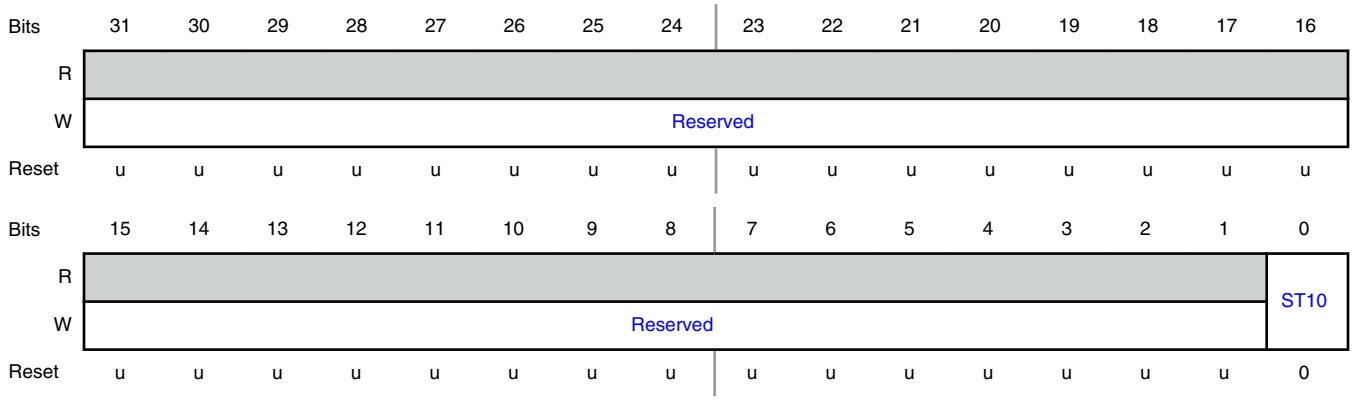
Register	Offset
HWVADST10	F88h

Function

This register controls the operation of the filter block and resets the internal interrut flag. Once the HWVAD triggered an interrupt, a short ‘1’ pulse on bit ST10 clears the interrupt.

Keeping the bit on ‘1’ level for some time also has a special function for filter convergence.

Diagram



Fields

Field	Function
31-1 —	Reserved Reserved. Read value is undefined, only zero should be written
0 ST10	ST10 Once the HWVAD has triggered an interrupt, a short '1' pulse on bit ST10 clears the interrupt. Alternatively, keeping the bit on '1' level for some time has a special function for filter convergence. 0b - Normal operation, waiting for HWVAD trigger event (stage 0). 1b - Reset internal interrupt flag by writing a 1 pulse.

16.1.18 HWVAD Filter Reset Register (HWVADRSTT)

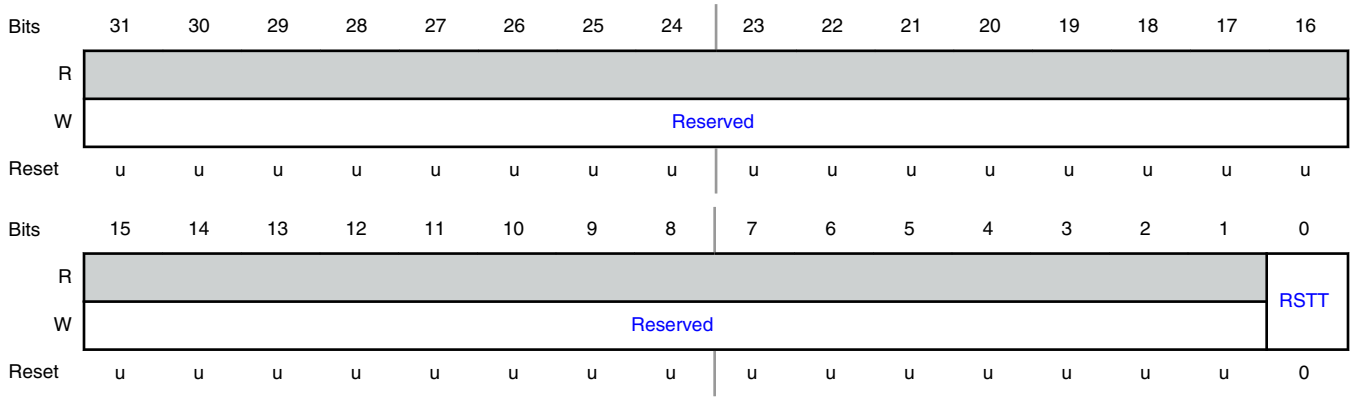
Offset

Register	Offset
HWVADRSTT	F8Ch

Function

Setting bit RSTT to ‘1’ causes a synchronous reset of all filters inside the HVWAD. The RSTT bit must be cleared in order to allow HWVAD operation.

Diagram



Fields

Field	Function
31-1 —	Reserved Reserved. Read value is undefined, only zero should be written
0 RSTT	HWVAD Filter Reset Writing a 1, then writing a '0' resets all filter values. 0b - Filters anr not held in reset 1b - Holds the filters in reset.

16.1.19 HWVAD Noise Estimator Gain Register (HWVADTHGN)

Offset

Register	Offset
HWVADTHGN	F90h

Function

Gain value for the noise estimator value. This parameter is used in the following calculation (implemented in hardware):

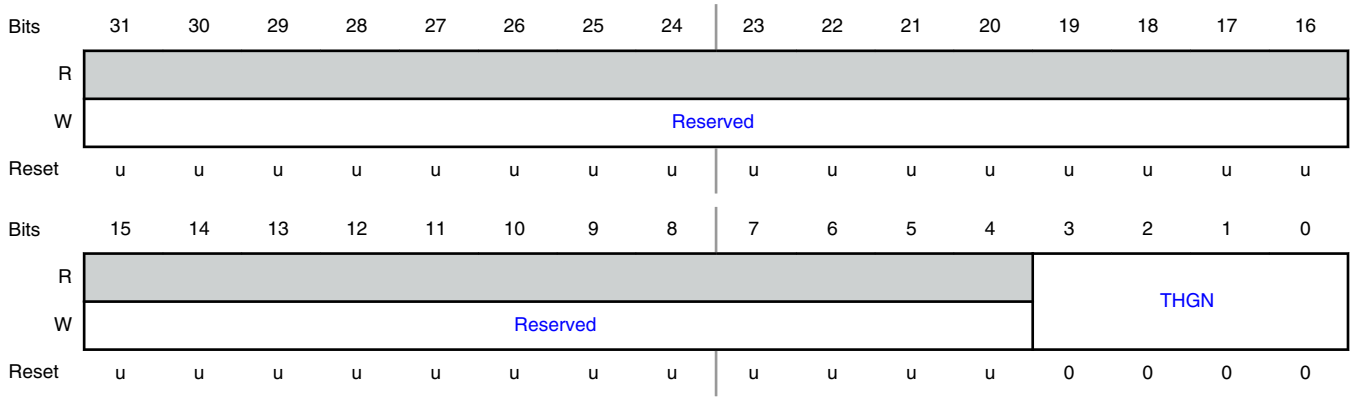
if z8 * (THGS+1) > z7 (THGN+1)

HWVAD_RESULT = 1;

else

HWVAD_RESULT = 0

Diagram



Fields

Field	Function
31-4 —	Reserved Reserved. Read value is undefined, only zero should be written.
3-0 THGN	Gain Value for Noise Estimator Values 0 to 14. 0 corresponds to a gain of 1. THGN and THGS are used within the hardware to determine when to assert the HWVAD result.

16.1.20 HWVAD Signal Estimator Gain Register (HWVADTHGS)

Offset

Register	Offset
HWVADTHGS	F94h

Function

Gain value for the signal estimator value. This parameter is used in the following calculation (implemented in hardware):

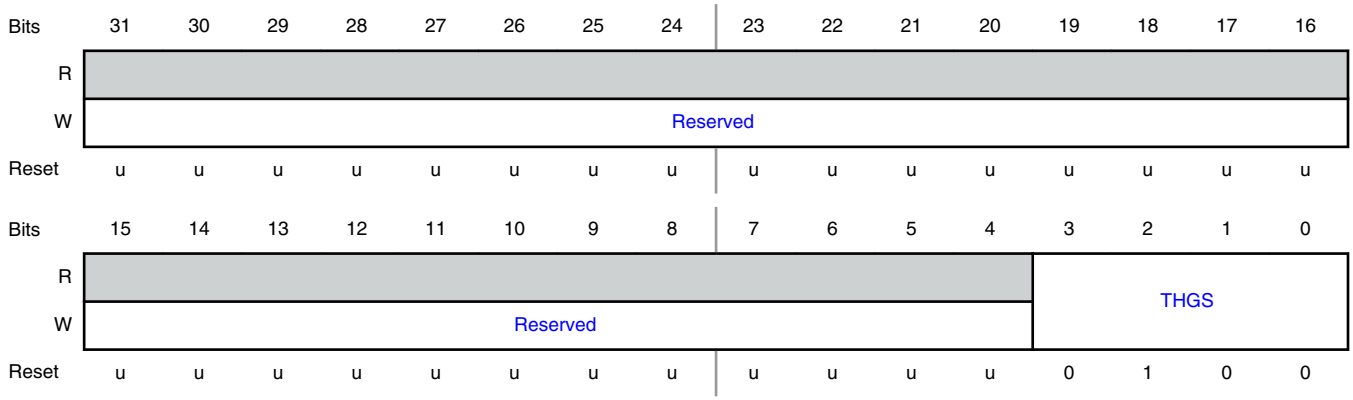
if z8 * (THGS+1) > z7 * (THGN+1)

HWVAD_RESULT = 1;

else

HWVAD_RESULT = 0

Diagram



Fields

Field	Function
31-4 —	Reserved Reserved. Read value is undefined, only zero should be written.
3-0 THGS	Gain Value for Signal Estimator Values 0 to 14. 0 corresponds to a gain of 1. THGN and THGS are used within the hardware to determine when to assert the HWVAD result.

16.1.21 HWVAD Noise Envelope Estimator Register (HWVADLOWZ)

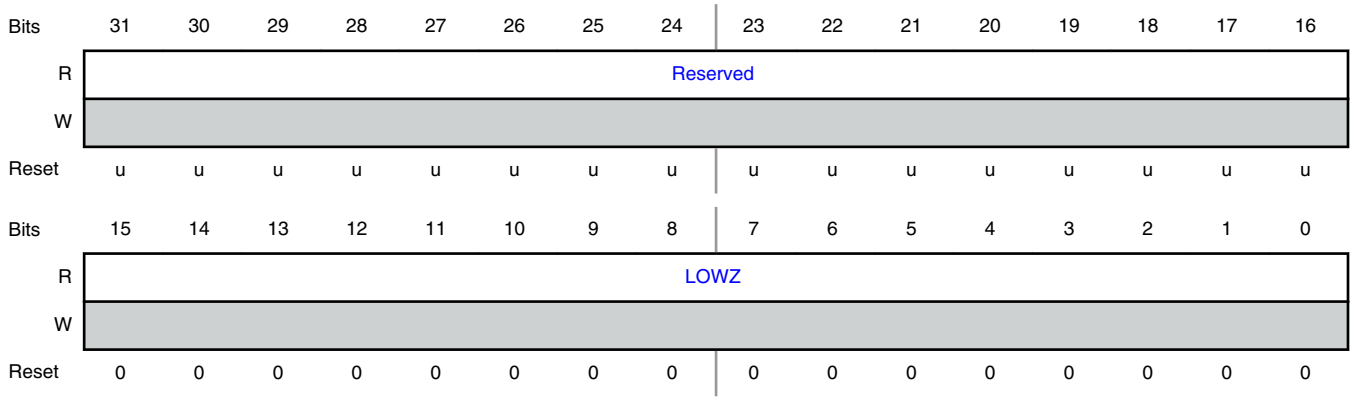
Offset

Register	Offset
HWVADLOWZ	F98h

Function

This register contains of the noise envelop estimator.

Diagram



Fields

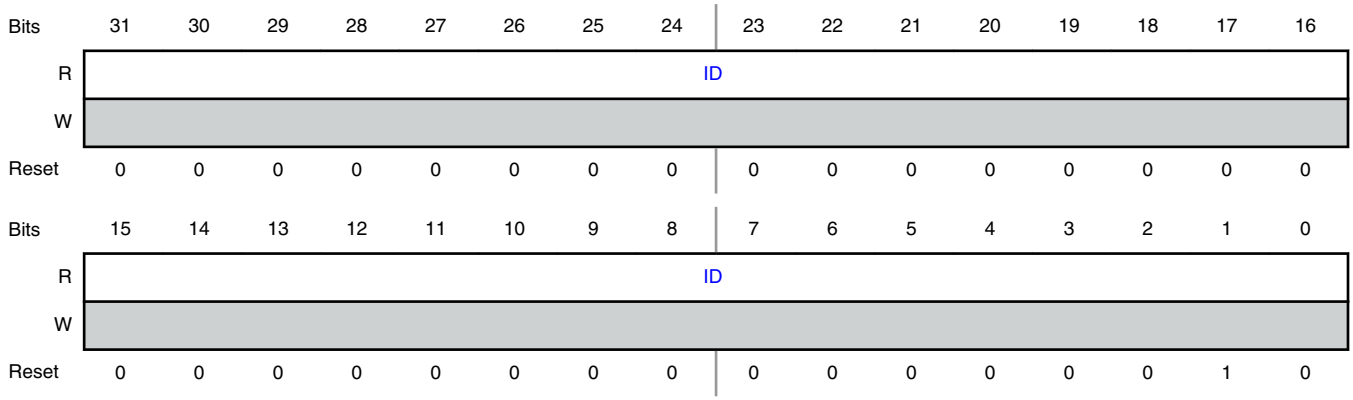
Field	Function
31-16 —	Reserved Reserved. Read value is undefined, only zero should be written.
15-0 LOWZ	Noise Envelope Estimator Value This field contains 2 bytes of the output of filter stage z7. It can be used as an indication for the noise floor and must be evaluated by software. **NOTE** For power saving reasons, this field is not synchronized to the AHB bus clock domain. To ensure correct data is read, the register should be read twice. If the data is the same, then the data is correct, if not, the field should be read one more time. The noise floor is a slowly moving calculation, so several reads in a row can guarantee that register value being read can be assured not to be in the middle of a transition.

16.1.22 Module Identification Register (ID)

Offset

Register	Offset
ID	FFCh

Diagram



Fields

Field	Function
31-0 ID	Identity Indicate module ID and the number of channels in this DMIC interface.

Chapter 17
12-bit ADC controller (ADC)

17.1 ADC register descriptions

17.1.1 ADC memory map

ADC base address: 4008_9000h

Offset	Register	Width (In bits)	Access	Reset value
0h	ADC Control Register (CTRL)	32	RW	See description
8h	ADC Conversion Sequence Control Register (SEQ_CTRL)	32	RW	See description
10h	ADC Sequence Global Data Register (SEQ_GDAT)	32	RO	See description
20h - 3Ch	ADC Channel a Data Register (DAT0 - DAT7)	32	RO	See description
50h	ADC Low Compare Threshold Register 0 (THR0_LOW)	32	RW	See description
54h	ADC Low Compare Threshold Register 1 (THR1_LOW)	32	RW	See description
58h	ADC High Compare Threshold Register 0 (THR0_HIGH)	32	RW	See description
5Ch	ADC High Compare Threshold Register 1 (THR1_HIGH)	32	RW	See description
60h	ADC Channel-Threshold Select Register (CHAN_THRSEL)	32	RW	See description
64h	ADC Interrupt Enable Register (INTEN)	32	RW	See description
68h	ADC Flags Register (FLAGS)	32	RW	See description
6Ch	ADC Startup Register (STARTUP)	32	RW	See description
70h	Second ADC Control Register (GPADC_CTRL0)	32	RW	See description
74h	Third ADC Control Register (GPADC_CTRL1)	32	RW	See description

17.1.2 ADC Control Register (CTRL)

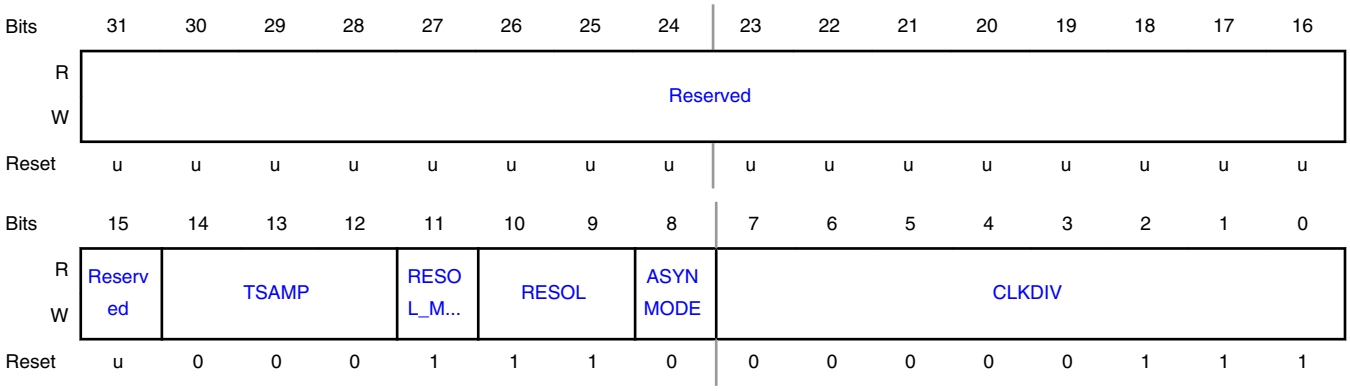
Offset

Register	Offset
CTRL	0h

Function

This register contains the clock divide value, resolution selection, sampling time selection, and mode controls.

Diagram



Fields

Field	Function
31-15 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
14-12 TSAMP	Sample Time The default sampling period (TSAMP = 000) at the start of each conversion is 2.5 ADC clock periods. Depending on a variety of factors, including operating conditions and the output impedance of the analog source, longer sampling times may be required. The TSAMP field should stay to default during application The TSAMP field specifies the number of additional ADC clock cycles, from zero to seven, by which the sample period will be extended. The total conversion time will increase by the same number of clocks. 000b The sample period will be the default 2.5 ADC clocks. A complete conversion with 12-bits of accuracy will require 15 clocks. 001b The sample period will be extended by one ADC clock to a total of 3.5 clock periods. A complete 12-bit conversion will require 16 clocks. 010b The sample period will be extended by two clocks to 4.5 ADC clock cycles. A complete 12-bit conversion will require 17 ADC clocks. ..... 111b The sample period will be extended by seven clocks to 9.5 ADC clock cycles. A complete 12-bit conversion will require 22 ADC clocks.
11 RESOL_MASK_DIS	According RESOL bit LSB bits are automatickly masked if RESOL_MASK_DIS = 0. According RESOL bit LSB bits are automatickly masked if RESOL_MASK_DIS = 0. If RESOL_MASK_DIS = 1, the 12bits comming from ADC are directly connect to register RESULT
10-9 RESOL	ADC Resolution Bits Number Note, whatever the resolution setting, the ADC data will always be shifted to use the MSBs of any ADC data words. Accuracy can be reduced to achieve higher conversion rates. A single conversion (including one conversion in a burst or sequence) requires the selected number of bits of resolution plus 3 ADC clocks. Remark: This field must only be altered when the ADC is fully idle. Changing it during any kind of ADC operation may have unpredictable results. Remark: ADC clock frequencies for various resolutions must not exceed: • 5x the system clock rate for 12-bit resolution. • 4.3x the system clock rate for 10-bit resolution. • 3.6x the system clock for 8-bit resolution. • 3x the bus clock rate for 6-bit resolution. 00b - 12-bit resolution. An ADC conversion requires 15 ADC clocks, plus any clocks specified by the TSAMP field. 01b - 10-bit resolution. An ADC conversion requires 13 ADC clocks, plus any clocks specified by the TSAMP field. 10b - 8-bit resolution. An ADC conversion requires 11 ADC clocks, plus any clocks specified by the TSAMP field. 11b - 6-bit resolution. An ADC conversion requires 9 ADC clocks, plus any clocks specified by the TSAMP field.

Table continues on the next page...

Table continued from the previous page...

Field	Function
8 ASYNMODE	Select Clock Mode 0b - Synchronous mode. Not Supported. 1b - Asynchronous mode. The ADC clock is based on the output of the ADC clock divider ADCCLKSEL in the SYSCON block.
7-0 CLKDIV	Reserved Reserved. No changes to this field are necessary.

17.1.3 ADC Conversion Sequence Control Register (SEQ_CTRL)

Offset

Register	Offset
SEQ_CTRL	8h

Function

This register controls triggering and channel selection for conversion sequence. Also specifies interrupt mode for sequence. All ADC conversions are controlled through this sequencer which can be used for a single conversion or sets of conversions on one or more channels.

Diagram

Fields

Field	Function
31 SEQ_ENA	Sequence Enable In order to avoid spuriously triggering the sequence, care should be taken to only set the SEQ_ENA bit when the selected trigger input is in its INACTIVE state (as defined by the TRIGPOL bit). If this condition is not met, the sequence will be triggered immediately upon being enabled. 0b - Disabled. Sequence A is disabled. Sequence A triggers are ignored. If this bit is cleared while sequence A is in progress, the sequence will be halted at the end of the current conversion. After the sequence is re-enabled, a new trigger will be required to restart the sequence beginning with the next enabled channel. 1b - Enabled. Sequence A is enabled.
30 MODE	Mode Indicates whether the primary method for retrieving conversion results for this sequence will be accomplished via reading the global data register (SEQ_GDAT) at the end of each conversion, or the individual channel result registers at the end of the entire sequence. Impacts when conversion-complete interrupt/DMA trigger for sequence-A will be generated and which overrun conditions contribute to an overrun interrupt as described below. 0b - End of conversion. The sequence A interrupt/DMA trigger will be set at the end of each individual ADC conversion performed under sequence A. This flag will mirror the SEQ_GDAT[DATAVALID] bit. The SEQ_GDAT[OVERRUN] bit will contribute to generation of an overrun interrupt/DMA trigger if enabled. 1b - End of sequence. The sequence A interrupt/DMA trigger will be set when the entire set of sequence-A conversions completes. This flag will need to be explicitly cleared by software or by the DMA-clear signal in this mode. The SEQ_GDAT[OVERRUN] bit will NOT contribute to generation of an overrun interrupt/DMA trigger since it is assumed this register may not be utilized in this mode.
29 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
28 SINGLESTEP	Single Step When this bit is set, a hardware trigger or a write to the START bit will launch a single conversion on the next channel in the sequence instead of the default response of launching an entire sequence of conversions. Once all of the channels comprising a sequence have been converted, a subsequent trigger will repeat the sequence beginning with the first enabled channel. Interrupt generation will still occur either after each individual conversion or at the end of the entire sequence, depending on the state of the MODE bit.
27 BURST	Burst Writing a 1 to this bit will cause this conversion sequence to be continuously cycled through. Other sequence A triggers will be ignored while this bit is set. Repeated conversions can be halted by clearing this bit. The sequence currently in progress will be completed before conversions are terminated. Note that a new sequence could begin just before BURST is cleared.

Table continues on the next page...

Table continued from the previous page...

Field	Function
26 START	Start Writing a 1 to this field will launch one pass through this conversion sequence. The behavior will be identical to a sequence triggered by a hardware trigger. Do not write 1 to this bit if the BURST bit is set. Remark: This bit is only set to a 1 momentarily when written to launch a conversion sequence. It will consequently always read back as a zero.
25 START_BEHAVIOUR	Start Behavior 0b - ADC is in repeat start mode, so a start is issued before every conversion. This is necessary if the sequencer is being used with more than one input. In this mode the ADC word rate is divided by two. 1b - ADC is in continuous start mode which gives the quickest operation. This setting can only be used if a single ADC input is selected.
24-20 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
19 SYNCBYPASS	Synchronization Bypass Setting this bit allows the hardware trigger input to bypass synchronization flip-flop stages and therefore shorten the time between the trigger input signal and the start of a conversion. There are slightly different criteria for whether or not this bit can be set depending on the clock operating mode: Synchronous mode (the ASYNMODE in the CTRL register = 0): Synchronization may be bypassed (this bit may be set) if the selected trigger source is already synchronous with the main system clock (eg. coming from an on-chip, system-clock-based timer). Whether this bit is set or not, a trigger pulse must be maintained for at least one system clock period. Asynchronous mode (the ASYNMODE in the CTRL register = 1): Synchronization may be bypassed (this bit may be set) if it is certain that the duration of a trigger input pulse will be at least one cycle of the ADC clock (regardless of whether the trigger comes from and on-chip or off-chip source). If this bit is NOT set, the trigger pulse must at least be maintained for one system clock period. 0: Enable trigger synchronization. The hardware trigger bypass is not enabled. 1: Bypass trigger synchronization. The hardware trigger bypass is enabled. 0b - 1b -
18 TRIGPOL	Trigger Polarity Select the polarity of the selected input trigger for this conversion sequence. Remark: In order to avoid generating a spurious trigger, it is recommended writing to this field only when SEQ_ENA (bit 31) is low. It is safe to change this field and set bit 31 in the same write. 0: A negative edge launches the conversion sequence on the selected trigger input. 1: A positive edge launches the conversion sequence on the selected trigger input. 0b - 1b -

Table continues on the next page...

Table continued from the previous page...

Field	Function
17-12 TRIGGER	Trigger Select which of the available hardware trigger sources will cause this conversion sequence to be initiated. Program the trigger input number in this field. Setting: 0 : PINT0; 1 : PWM8; 2 : PWM9; 3 : ARM TX EV. Remark: In order to avoid generating a spurious trigger, it is recommended writing to this field only when SEQ_ENA (bit 31) is low. It is safe to change this field and set SEQ_ENA in the same write.
11-8 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
7-0 CHANNELS	ADC Channels Select which one or more of the ADC channels will be sampled and converted when this sequence is launched. A 1 in any bit of this field will cause the corresponding channel to be included in the conversion sequence, where bit 0 corresponds to channel 0, bit 1 to channel 1 and so forth. Bit 6 is channel 6; the supply monitor. Bit 7 is channel 7; the temperature sensor. When this conversion sequence is triggered, either by a hardware trigger or via software command, ADC conversions will be performed on each enabled channel, in sequence, beginning with the lowest-ordered channel. Remark: This field can ONLY be changed while SEQ_ENA (bit 31) is LOW. It is allowed to change this field and set bit 31 in the same write.

17.1.4 ADC Sequence Global Data Register (SEQ_GDAT)

Offset

Register	Offset
SEQ_GDAT	10h

Function

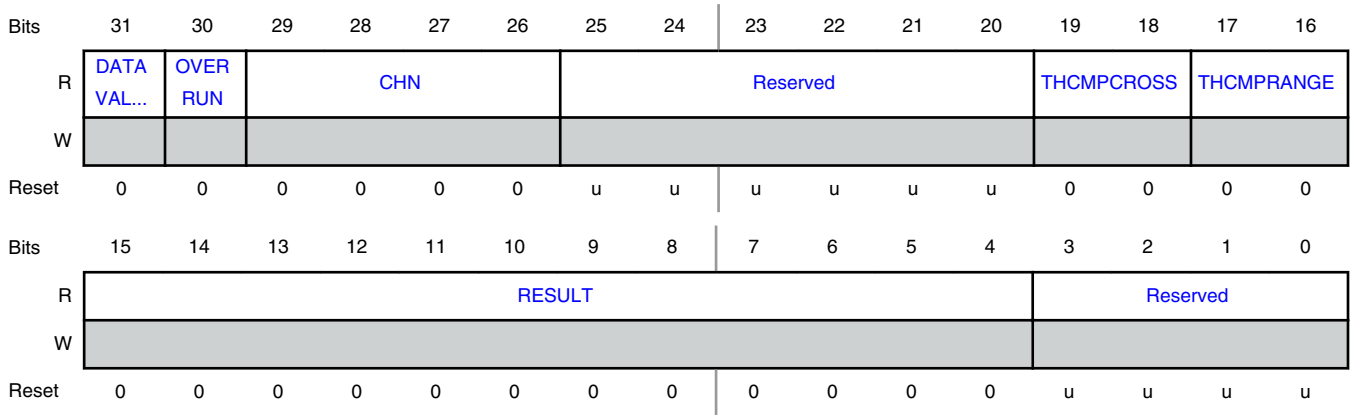
This register contains the result of the most recent ADC conversion completed under each conversion sequence. Results of ADC conversions can be read in one of two ways. One is to use this register to read data from the ADC at the end of each ADC conversion. The other is to read the individual ADC Channel Data (DATn) registers, typically after the entire sequence has completed. It is recommended to use one method consistently for a given conversion sequence.

This register is useful in conjunction with DMA operation - particularly when the channels selected for conversion are not sequential (hence the addresses of the individual result registers will not be sequential, making it difficult for the DMA engine to address them). For interrupt-driven code, it will more likely be advantageous to wait for an entire sequence to complete and then retrieve the results from the individual channel registers.

NOTE

The method to be employed for each sequence should be reflected in the SEQ_CTRL[MODE] bit since this will impact interrupt and overrun flag generation.

Diagram



Fields

Field	Function
31 DATAVALID	Data Valid This bit is set to 1 at the end of each conversion when a new result is loaded into the RESULT field. It is cleared whenever this register is read. This bit will cause a conversion-complete interrupt for the corresponding sequence if the MODE bit (in SEQ_CTRL) for that sequence is set to 0 (and if the interrupt is enabled).
30 OVERRUN	Overrun This bit is set if a new conversion result is loaded into the RESULT field before a previous result has been read - i.e. while the DATAVALID bit is set. This bit is cleared, along with the DATAVALID bit, whenever this register is read. This bit will contribute to an overrun interrupt/DMA trigger if the MODE bit (in SEQ_CTRL) for the corresponding sequence is set to 0 (and if the overrun interrupt is enabled).
29-26 CHN	Channel These bits contain the channel from which the RESULT bits were converted (e.g. 0000 identifies channel 0, 0001 channel 1, etc.).
25-20 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
19-18 THCMPCROSS	Threshold Crossing Comparison Result 00b - No threshold Crossing detected: The most recent completed conversion on this channel had the same relationship (above or below) to the threshold value established by the designated LOW threshold register (THRn_LOW) as did the previous conversion on this channel. 01b - Reserved. 10b - Downward Threshold Crossing Detected. Indicates that a threshold crossing in the downward direction has occurred - i.e. the previous sample on this channel was above the threshold value established by the designated LOW threshold register (THRn_LOW) and the current sample is below that threshold. 11b - Upward Threshold Crossing Detected. Indicates that a threshold crossing in the upward direction has occurred - i.e. the previous sample on this channel was below the threshold value established by the designated LOW threshold register (THRn_LOW) and the current sample is above that threshold.
17-16 THCMPRANGE	Threshold Range Comparison Result 00b - In Range: The last completed conversion was greater than or equal to the value programmed into the designated LOW threshold register (THRn_LOW) but less than or equal to the value programmed into the designated HIGH threshold register (THRn_HIGH). 01b - Below Range: The last completed conversion on was less than the value programmed into the designated LOW threshold register (THRn_LOW). 10b - Above Range: The last completed conversion was greater than the value programmed into the designated HIGH threshold register (THRn_HIGH). 11b - Reserved.
15-4 RESULT	ADC Conversion Result This field contains the 12-bit ADC conversion result from the most recent conversion performed under conversion sequence associated with this register. DATAVALID = 1 indicates that this result has not yet been read. If less than 12-bit resolultion is used the ADC result occupies the upper MSBs and unused LSBs should be ignored.
3-0 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

17.1.5 ADC Channel a Data Register (DAT0 - DAT7)

Offset

For a = 0 to 7:

Register	Offset
DATa	20h + (a × 4h)

Function

These registers hold the result of the last conversion completed for each ADC channel. They also include status bits to indicate when a conversion has been completed, when a data overrun has occurred, and where the most recent conversion fits relative to the range dictated by the high and low threshold registers.

Results of ADC conversion can be read in one of two ways. One is to use the SEQ_GDAT register for each of the sequences to read data from the ADC at the end of each ADC conversion. The other is to use these individual ADC Channel Data registers, typically after the entire sequence has completed. It is recommended to use one method consistently for a given conversion sequence.

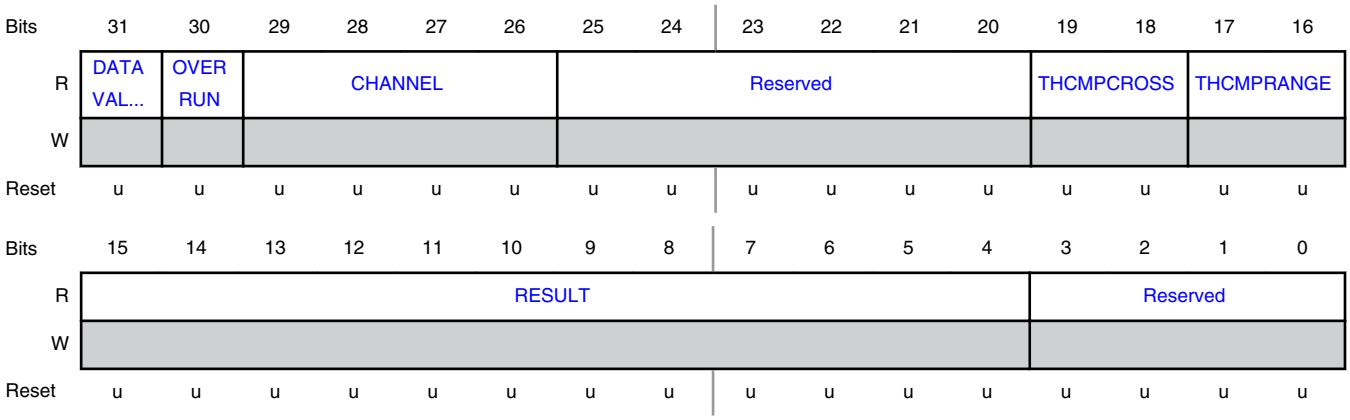
NOTE

The method to be employed for each sequence should be reflected in the MODE bit in the SEQ_CTRL register since this will impact interrupt and overrun flag generation.

The information presented in the DAT registers always pertains to the most recent conversion completed on that channel regardless of what sequence requested the conversion or which trigger caused it.

The OVERRUN fields for each channel are also replicated in the FLAGS register.

Diagram



Fields

Field	Function
31 DATAVALID	Data Valid This bit is set to 1 at the end of each conversion when a new result is loaded into the RESULT field. It is cleared whenever this register is read. This bit will cause a conversion-complete interrupt for the corresponding sequence if the SEQ_CTRL[MODE] bit for that sequence is set to 0 (and if the interrupt is enabled).
30 OVERRUN	Overrun This bit is set if a new conversion result is loaded into the RESULT field before a previous result has been read - i.e. while the DATAVALID bit is set. This bit is cleared, along with the DATAVALID bit, whenever this register is read. This bit will contribute to an overrun interrupt/DMA trigger if the SEQ_CTRL[MODE] bit for the corresponding sequence is set to 0 (and if the overrun interrupt is enabled).

Table continues on the next page...

Table continued from the previous page...

Field	Function
29-26 CHANNEL	Channel This field identifies the ADC channel associated with the data in this register.
25-20 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
19-18 THCMPCROSS	Threshold Crossing Comparison Result 00b - No threshold Crossing detected: The most recent completed conversion on this channel had the same relationship (above or below) to the threshold value established by the designated LOW threshold register (THRn_LOW) as did the previous conversion on this channel. 01b - Reserved. 10b - Downward Threshold Crossing Detected. Indicates that a threshold crossing in the downward direction has occurred - i.e. the previous sample on this channel was above the threshold value established by the designated LOW threshold register (THRn_LOW) and the current sample is below that threshold. 11b - Upward Threshold Crossing Detected. Indicates that a threshold crossing in the upward direction has occurred - i.e. the previous sample on this channel was below the threshold value established by the designated LOW threshold register (THRn_LOW) and the current sample is above that threshold.
17-16 THCMPRANGE	Threshold Range Comparison Result 00b - In Range: The last completed conversion was greater than or equal to the value programmed into the designated LOW threshold register (THRn_LOW) but less than or equal to the value programmed into the designated HIGH threshold register (THRn_HIGH). 01b - Below Range: The last completed conversion on was less than the value programmed into the designated LOW threshold register (THRn_LOW). 10b - Above Range: The last completed conversion was greater than the value programmed into the designated HIGH threshold register (THRn_HIGH). 11b - Reserved.
15-4 RESULT	ADC Conversion Result This field contains the 12-bit ADC conversion result from the most recent conversion performed under conversion sequence associated with this register. DATAVALID = 1 indicates that this result has not yet been read. If less than 12-bit resolultion is used, the ADC result occupies the upper MSBs and unused LSBs should be ignored.
3-0 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

17.1.6 ADC Low Compare Threshold Register 0 (THR0_LOW)

Offset

Register	Offset
THR0_LOW	50h

Function

This register sets the LOW threshold levels against which ADC conversions on all channels will be compared.

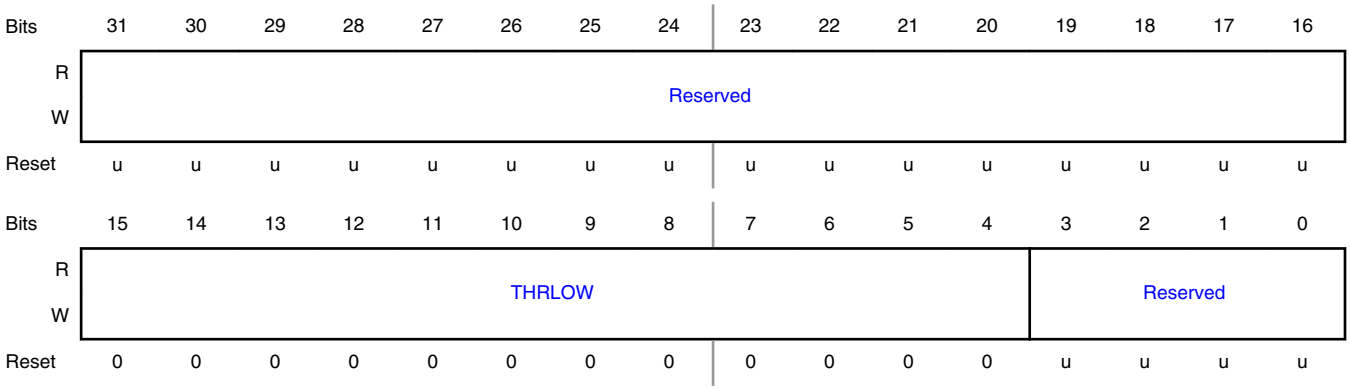
Each channel will either be compared to the THR0_LOW/HIGH registers or to the THR1_LOW/HIGH registers depending on what is specified for that channel in the CHAN_THRSEL register.

A conversion result less than this value on any channel will cause the SEQ_GDAT[THCMPRANGE] status bits for that channel to be set to 0b01. This result will also generate an interrupt/DMA trigger if enabled to do so via the INTEN[ADCMPINTENn] bits associated with each channel.

If, for two successive conversions on a given channel, the latest result is below this threshold and the previous value is equal-to or above this threshold, then a threshold crossing has occurred. In this case the SEQ_GDAT[THCMPCROSS] status bits will indicate that a downward threshold crossing has occurred and the direction of the crossing. A threshold crossing event will also generate an interrupt/DMA trigger if enabled to do so via the INTEN[ADCMPINTENn] bits associated with each channel.

THCMPRANGE and THCMPCROSS are channel specific status bits available in the relevant DATn register.

Diagram



Fields

Field	Function
31-16 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
15-4 THRLOW	Threshold Low Low threshold value against which ADC results will be compared

Table continues on the next page...

Table continued from the previous page...

Field	Function
3-0 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

17.1.7 ADC Low Compare Threshold Register 1 (THR1_LOW)

Offset

Register	Offset
THR1_LOW	54h

Function

This register sets the LOW threshold levels against which ADC conversions on all channels will be compared.

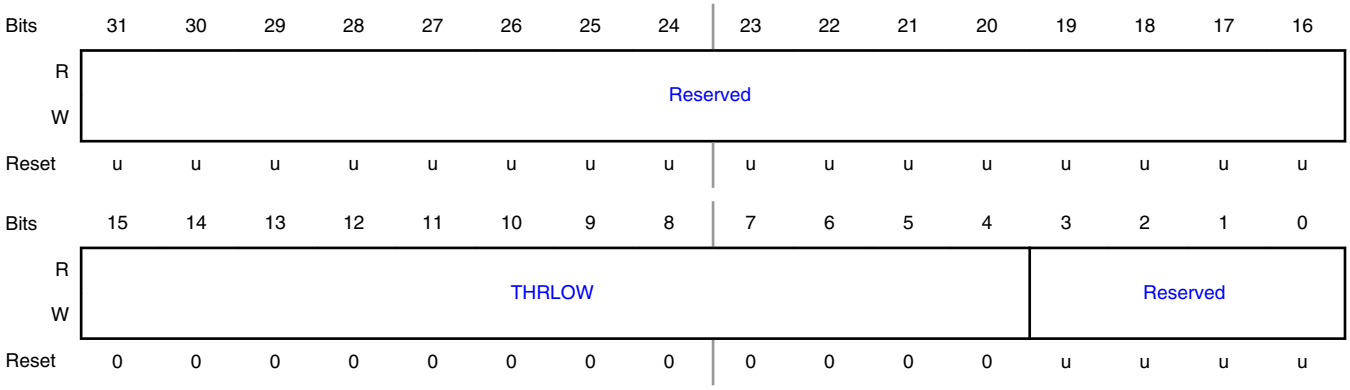
Each channel will either be compared to the THR0_LOW/HIGH registers or to the THR1_LOW/HIGH registers depending on what is specified for that channel in the CHAN_THRSEL register.

A conversion result less than this value on any channel will cause the SEQ_GDAT[THCMPRANGE] status bits for that channel to be set to 0b01. This result will also generate an interrupt/DMA trigger if enabled to do so via the INTEN[ADCMPINTENn] bits associated with each channel.

If, for two successive conversions on a given channel, the latest result is below this threshold and the previous value is equal-to or above this threshold, then a threshold crossing has occurred. In this case the SEQ_GDAT[THCMPCROSS] status bits will indicate that a downward threshold crossing has occurred and the direction of the crossing. A threshold crossing event will also generate an interrupt/DMA trigger if enabled to do so via the INTEN[ADCMPINTENn] bits associated with each channel.

THCMPRANGE and THCMPCROSS are channel specific status bits available in the relevant DATn register.

Diagram



Fields

Field	Function
31-16 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
15-4 THRLOW	Threshold Low Low threshold value against which ADC results will be compared
3-0 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

17.1.8 ADC High Compare Threshold Register 0 (THR0_HIGH)

Offset

Register	Offset
THR0_HIGH	58h

Function

These registers set the HIGH threshold level against which ADC conversions on all channels will be compared.

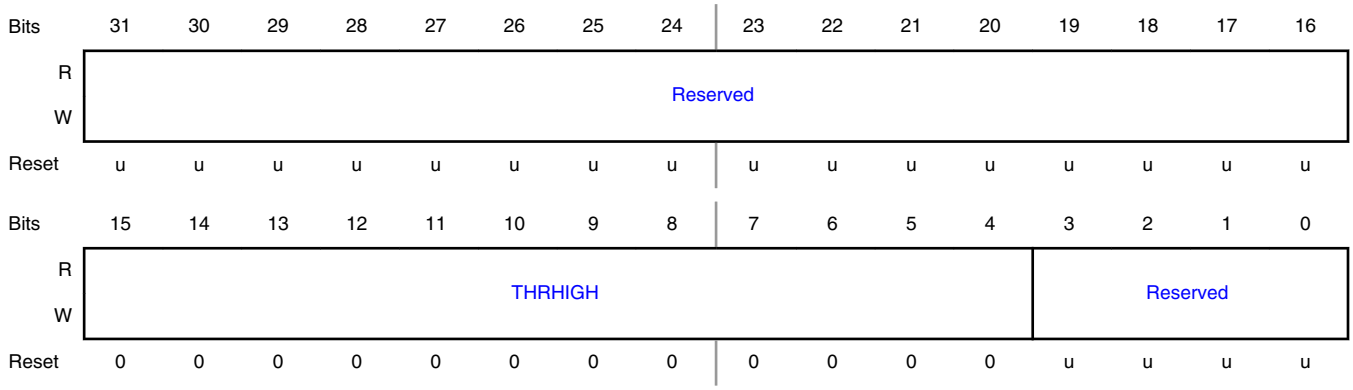
Each channel will either be compared to the THR0_LOW/HIGH registers or to the THR1_LOW/HIGH registers depending on what is specified for that channel in the CHAN_THRSEL register.

A conversion result greater than this value on any channel will cause the DAT[THCMPRANGE] status bits for that channel to be set to 0b10. This result will also generate an interrupt/DMA trigger if enabled to do so via the INTEN[ADCMPINTEN] bits associated with each channel.

If, for two successive conversions on a given channel, the current result is above this threshold and the previous result is equal-to or below this threshold, than a threshold crossing has occurred. In this case the DATn[THCMPCROSS] status bits will indicate that a upward threshold crossing has occurred. A threshold crossing event will also generate an interrupt/DMA trigger if enabled to do so via the INTEN[ADCMPINTEN] bits associated with each channel.

THCMPRANGE and THCMPCROSS are channel specific status bits available in the relevant DATn register.

Diagram



Fields

Field	Function
31-16 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
15-4 THRHIGH	Threshold High High threshold value against which ADC results will be compared
3-0 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

17.1.9 ADC High Compare Threshold Register 1 (THR1_HIGH)

Offset

Register	Offset
THR1_HIGH	5Ch

Function

These registers set the HIGH threshold level against which ADC conversions on all channels will be compared.

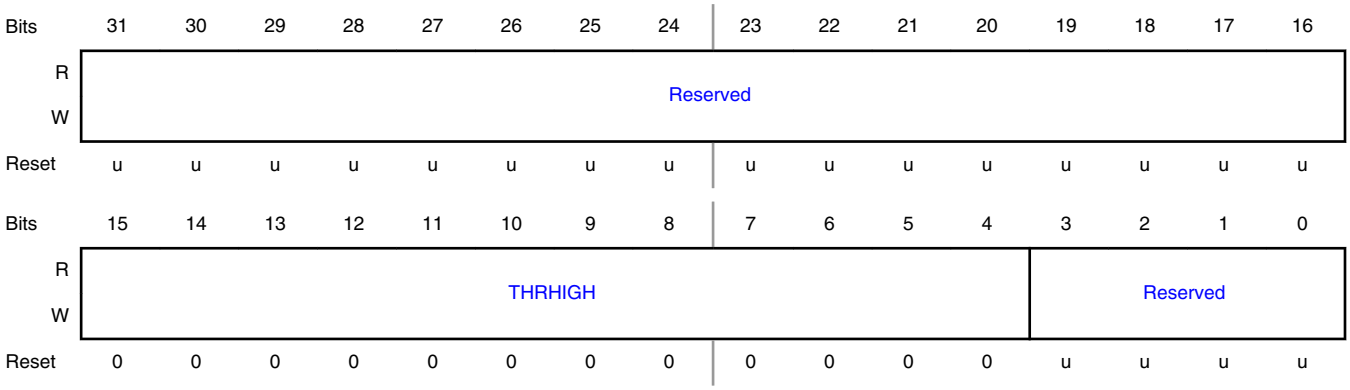
Each channel will either be compared to the THR0_LOW/HIGH registers or to the THR1_LOW/HIGH registers depending on what is specified for that channel in the CHAN_THRSEL register.

A conversion result greater than this value on any channel will cause the DAT[THCMPRANGE] status bits for that channel to be set to 0b10. This result will also generate an interrupt/DMA trigger if enabled to do so via the INTEN[ADCMPINTEN] bits associated with each channel.

If, for two successive conversions on a given channel, the current result is above this threshold and the previous result is equal-to or below this threshold, than a threshold crossing has occurred. In this case the DATn[THCMPCROSS] status bits will indicate that a upward threshold crossing has occurred. A threshold crossing event will also generate an interrupt/DMA trigger if enabled to do so via the INTEN[ADCMPINTEN] bits associated with each channel.

THCMPRANGE and THCMPCROSS are channel specific status bits available in the relevant DATn register.

Diagram



Fields

Field	Function
31-16 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
15-4 THRHIGH	Threshold High High value against which ADC results will be compared
3-0 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

17.1.10 ADC Channel-Threshold Select Register (CHAN_THRSEL)

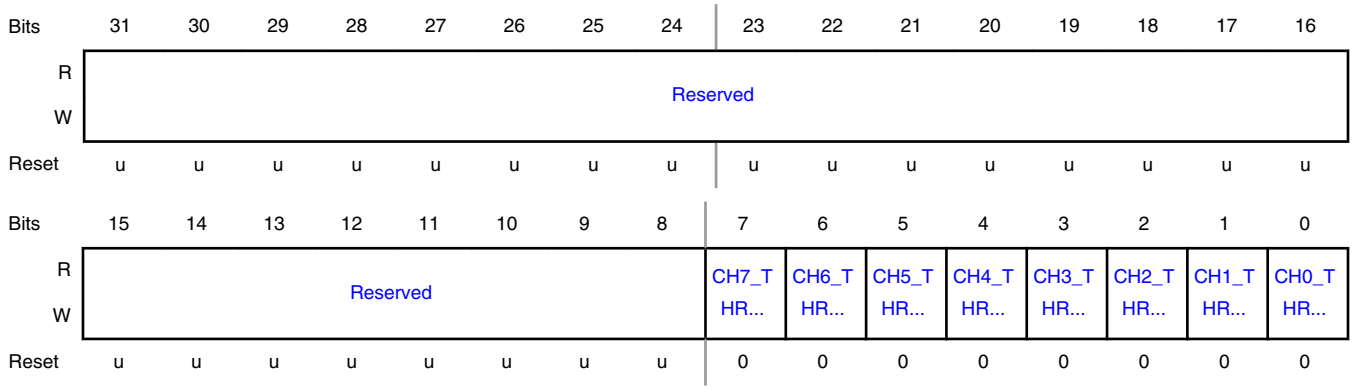
Offset

Register	Offset
CHAN_THRSEL	60h

Function

Specifies which set of threshold compare registers are to be used for each channel

Diagram



Fields

Field	Function
31-8 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
7-0 CHn_THRSEL	Channeln Threshold Select Threshold select for channel n. 0b - Threshold 0. Results for this channel will be compared against the threshold levels indicated in the THR0_LOW and THR0_HIGH registers. 1b - Threshold 1. Results for this channel will be compared against the threshold levels indicated in the THR1_LOW and THR1_HIGH registers.

17.1.11 ADC Interrupt Enable Register (INTEN)

Offset

Register	Offset
INTEN	64h

Function

There are four separate interrupt requests generated by the ADC: conversion, these are conversion or sequence complete interrupt for each of the sequencer, a threshold-comparison out-of-range interrupt, and a data overrun interrupt. The two conversion/sequence-complete interrupts can also serve as DMA triggers. The threshold and data overrun interrupts share a slot in the NVIC.

These interrupts may be combined into one request on some chips if there is a limited number of interrupt slots. This register contains the interrupt-enable bits for each interrupt.

In this register, threshold events selected in the ADCMPINTENn bits are described as follows:

- Disabled: Threshold comparisons on channel n will not generate an ADC threshold-compare interrupt/DMA trigger.

- Outside threshold: A conversion result on channel n which is outside the range specified by the designated HIGH and LOW threshold registers will set the channel n THCMP flag in the FLAGS register and generate an ADC threshold-compare interrupt/DMA trigger.
- Crossing threshold: Detection of a threshold crossing on channel n will set the channel n THCMP flag in the FLAGS register and generate an ADC threshold-compare interrupt/DMA trigger.

NOTE

Overrun and threshold-compare interrupts related to a particular channel will occur regardless of which sequence was in progress at the time the conversion was performed or what trigger caused the conversion.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved													ADCMPINTEN7		ADCMPIN...
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADCMPIN...	ADCMPINTEN5		ADCMPINTEN4		ADCMPINTEN3		ADCMPINTEN2		ADCMPINTEN1		ADCMPINTEN0		OVR_INT...	Reserved	SEQA_IN...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	u	0

Fields

Field	Function
31-19 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
18-17: ADCMPINTEN7 16-15: ADCMPINTEN6 14-13: ADCMPINTEN5 12-11: ADCMPINTEN4 10-9: ADCMPINTEN3 8-7: ADCMPINTEN2 6-5: ADCMPINTEN1 4-3: ADCMPINTEN0	Threshold Comparison Interrupt Enable for Channel n 00b - Disabled. 01b - Outside threshold. 10b - Crossing threshold. 11b - Reserved.
2 OVR_INTEN	Overrun Interrupt Enable. 0b - Disabled. The overrun interrupt is disabled. 1b - Enabled. The overrun interrupt is enabled. Detection of an overrun condition on any of the 8 channel data registers will cause an overrun interrupt/DMA trigger. In addition, if the MODE bit for a particular sequence is 0, then an overrun in the global data register for that sequence will also cause this interrupt/DMA trigger to be asserted.
1 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
0 SEQA_INTEN	Sequence A interrupt Enable 0b - Disabled. The sequence A interrupt/DMA trigger is disabled. 1b - Enabled. The sequence A interrupt/DMA trigger is enabled and will be asserted either upon completion of each individual conversion performed as part of sequence A, or upon completion of the entire A sequence of conversions, depending on the MODE bit in the SEQ_CTRL register.

17.1.12 ADC Flags Register (FLAGS)

Offset

Register	Offset
FLAGS	68h

Function

This register contains the interrupt/DMA trigger flags along with the individual overrun flags that contribute to an overrun interrupt and the component threshold-comparison flags that contribute to that interrupt. Note that the combination of the threshold and overrun interrupts have a slot in the NVIC. The sequencer also has a slot in the NVIC.

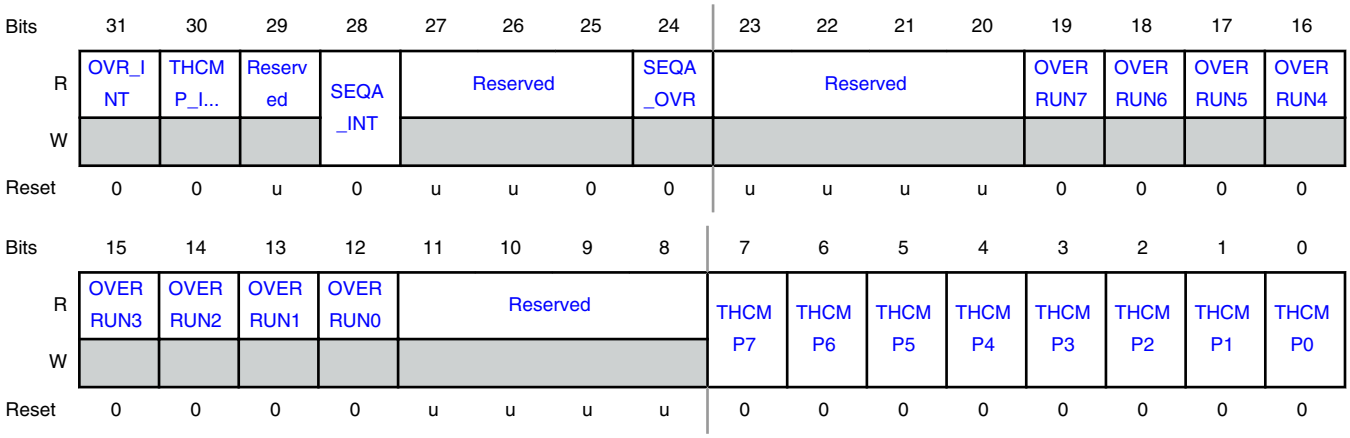
The channel OVERRUN flags mirror those appearing in the individual DAT registers for each channel and indicate a data overrun in each of those registers.

Likewise, the SEQA_OVR bit mirrors the OVERRUN bit in the global data registers (SEQ_GDAT).

NOTE

The SEQA_INT conversion/sequence-complete flag also serves as a DMA trigger.

Diagram



Fields

Field	Function
31 OVR_INT	Overrun Interrupt Flag Any overrun bit in any of the individual channel data registers will cause this interrupt. In addition, if the MODE bit in either of the SEQ_CTRL registers is 0 then the OVERRUN bit in the corresponding SEQ_GDAT register will also cause this interrupt. This interrupt must be enabled in the INTEN register. This bit will be cleared when all of the individual overrun bits have been cleared via reading the corresponding data registers.
30 THCMP_INT	Threshold Comparison Interrupt This bit will be set if any of the THCMP flags in the lower bits of this register are set to 1 (due to an enabled out-of-range or threshold-crossing event on any channel). Each type of threshold comparison interrupt on each channel must be individually enabled in the INTEN register to cause this interrupt. This bit will be cleared when all of the individual threshold flags are cleared via writing 1s to those bits.
29 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
28 SEQA_INT	Sequence Interrupt/DMA Trigger If the MODE bit in the SEQ_CTRL register is 0, this flag will mirror the DATAVALID bit in the sequence A global data register (SEQ_GDAT), which is set at the end of every ADC conversion performed as part of sequence A. It will be cleared automatically when the SEQ_GDAT register is read. If the MODE bit in the SEQ_CTRL register is 1, this flag will be set upon completion of an entire A sequence. In this case it must be cleared by writing a 1 to this SEQA_INT bit. This interrupt must be enabled in the INTEN register.
27-25 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
24 SEQA_OVR	SEQ_GDAT Overrun Status Mirror Mirrors the global OVERRUN status flag in the register
23-20 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
19-12 OVERRUNn	Overrun Mirrors the OVERRRUN status flag from the result register for ADC channel n
11-8 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
7-0 THCMPn	Threshold Comparison Event on Channel n. Set to 1 upon either an out-of-range result or a threshold-crossing result if enabled to do so in the INTEN register. This bit is cleared by writing a 1.

17.1.13 ADC Startup Register (STARTUP)

Offset

Register	Offset
STARTUP	6Ch

Function

This register is typically used only by the ADC API. ADC clock should be selected and running at full frequency prior to writing to this register.

Diagram

Fields

Field	Function
31-1 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
0 ADC_ENA	ADC Enable Bit This bit can only be set to a 1 by software. It is cleared automatically whenever the ADC is powered down. This bit must not be set until at least 10 microseconds after the ADC is powered up (typically by altering a system-level ADC power control bit).

17.1.14 Second ADC Control Register (GPADC_CTRL0)

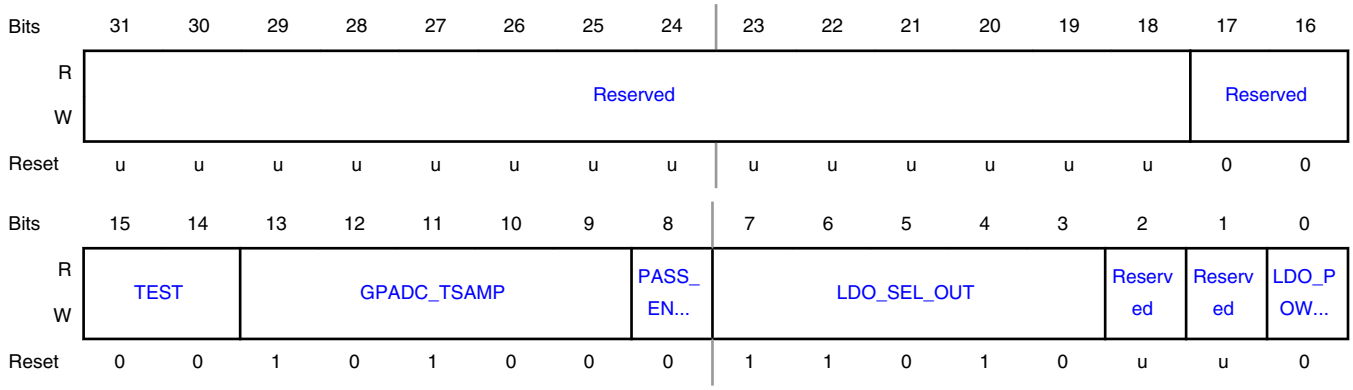
Offset

Register	Offset
GPADC_CTRL0	70h

Function

A further control register for the ADC that sets the gain mode, can account for the source impedance of signals connecting to the ADC. Additionally, control of the LDO within the ADC sub-system is managed by this register. This will be configured by the software API for the ADC.

Diagram



Fields

Field	Function
31-18 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
17-16 —	RESERVED Reserved. User software should write zeroes to reserved bits.
15-14 TEST	ADC Gain Mode Selection 00b - Normal functional mode (DIV4 mode). Input range is 0 to 3.6V, although max input voltage is affected by supply voltage of the device. 01b - Not used 10b - ADC in unity gain mode. (DIV1 mode). Input range is 0 to 0.9V. Voltages above this may damage the device. 11b - Not used.
13-9 GPADC_TSAMP	Extend ADC Sampling Time Extend ADC sampling time according to source impedance 0x0 - Not used 0x1 - 0.621 kΩ 0x14 (default) - 55 kΩ 0x1F - 87 kΩ
8 PASS_ENABLE	Pass Mode Enable Test feature only, do not modify this field.
7-3 LDO_SEL_OUT	LDO Output Select Software driver configures correct setting. Do not modify this field.

Table continues on the next page...

Table continued from the previous page...

Field	Function
2 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
1 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
0 LDO_POWER_EN	LDO Power Enable Signal (active high). This is for the LDO within the ADC itself. There is also LDOADC controlled from PMC that is outside the ADC block. The LDOADC should have been enabled for 10usec before enabling this LDO. After enabling this LDO it is necessary to wait for 230usec, before ADC sampling is commenced, so that full accuraccy of the ADC will be obtained. Control of this field is managed by the software driver.

17.1.15 Third ADC Control Register (GPADC_CTRL1)

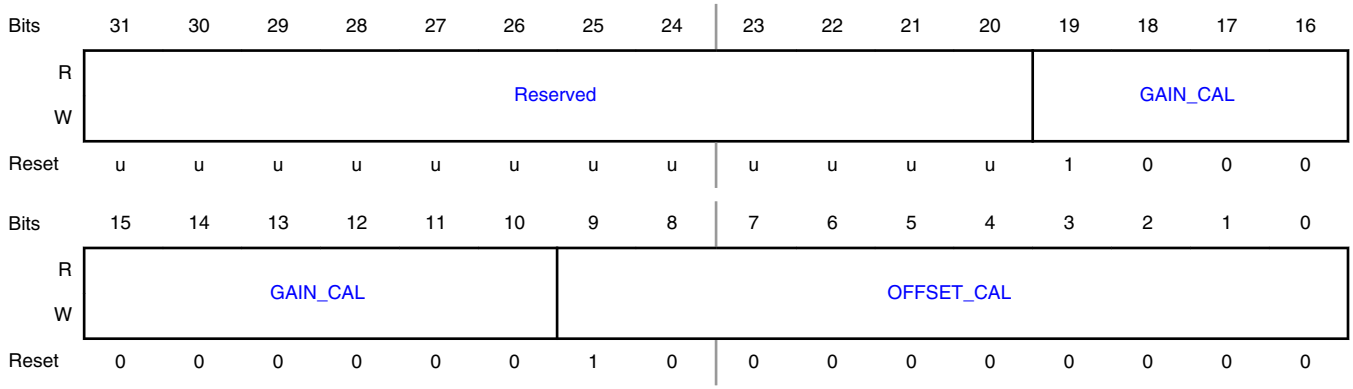
Offset

Register	Offset
GPADC_CTRL1	74h

Function

This register controls ADC internal gain and offset. During device test calibration values are calculated to trim the ADC to ensure it meets the device specification. These values are copied into this register by the ADC APIs functions.

Diagram



Fields

Field	Function
31-20 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
19-10 GAIN_CAL	Gain Cal This field is used within the ADC to compensate for any gain variation for this particular device.
9-0 OFFSET_CAL	Offset Cal This field is used within the ADC to compensate for a DC shift in values for this particular device.

Chapter 18
Flash Controller (Flash)

18.1 Flash register descriptions

18.1.1 Flash memory map

flash base address: 4000_9000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Command Register (CMD)	32	WO	0000_0000h
4h	Event Register (EVENT)	32	WO	See description
Ch	Auto Programming Register (AUTOPROG)	32	RW	See description
10h	Start Address for Next flash Command Register (STARTA)	32	RW	See description
14h	End Address for Next Flash Command Register (STOPA)	32	RW	See description
80h - 8Ch	Data Register a (DATAW0 - DATAW3)	32	RW	0000_0000h
FD8h	Clear Interrupt Enable Register (INT_CLR_ENABLE)	32	WO	See description
FDCh	Set Interrupt Enable Register (INT_SET_ENABLE)	32	WO	See description
FE0h	Interrupt Status Register (INT_STATUS)	32	RO	See description
FE4h	Interrupt Enable Register (INT_ENABLE)	32	RO	See description
FE8h	Clear Interrupt Status Register (INT_CLR_STATUS)	32	WO	See description
FECh	Set Interrupt Status Register (INT_SET_STATUS)	32	WO	See description
FFCh	Controller and Memory Module Identification Register (MODULE_ID)	32	RO	C40F_1500h

18.1.2 Command Register (CMD)

Offset

Register	Offset
CMD	0h

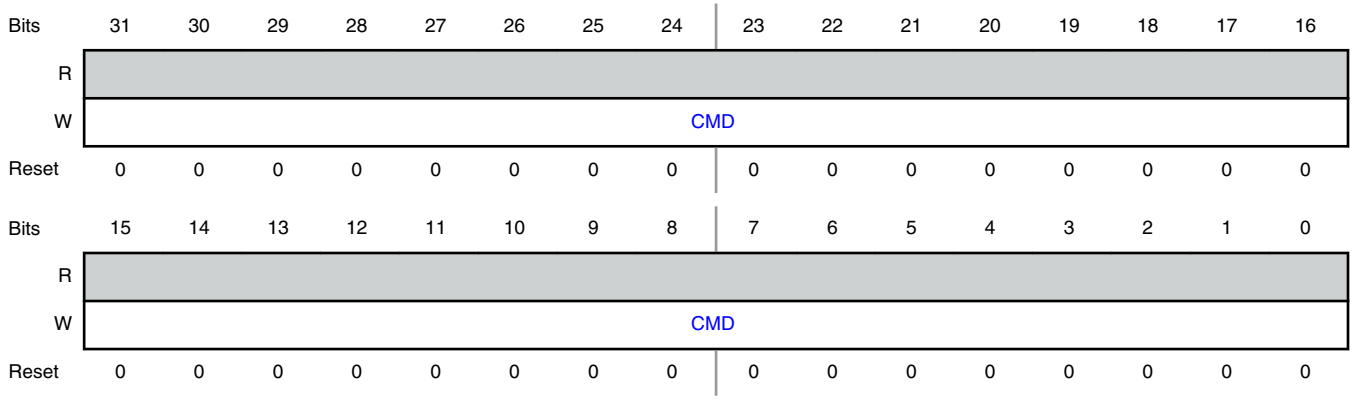
Function

The Command register is used to initiate activity in the flash controller. A “command” is any action performed by the controller, such as a mode change, programming, erasing, calculating a checksum over an address range. A command normally has parameters, such as an address or address range, data to be written, a mode specification. Parameters have to be written into corresponding registers before that the command is started. Writing parameters has no effect until the command is started.

Command execution is triggered when writing the CMD register.

When a command is executed, it sets appropriate bits in the INT_STATUS registers. Some commands also return additional information in other registers

Diagram



Fields

Field	Function
31-0 CMD	CMD

18.1.3 Event Register (EVENT)

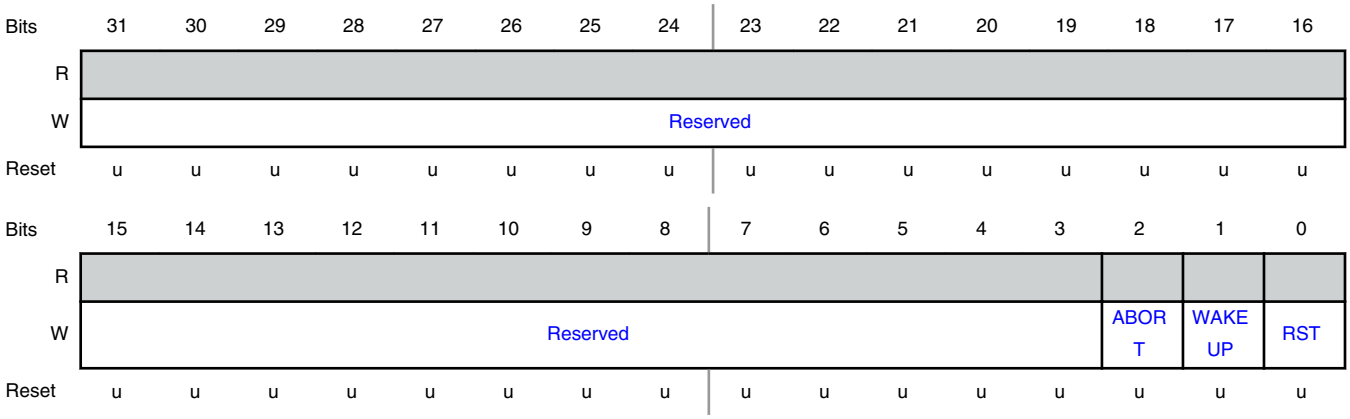
Offset

Register	Offset
EVENT	4h

Function

The event register is a write-only register that allows software to cause certain events to occur in the flash controller. These events are a reset, a command abort request, wake-up from power-down. The act of writing the register with one of the bits at 1 activates the generation of the corresponding event. Such an activation might be masked if the corresponding controller is already performing an event.

Diagram



Fields

Field	Function
31-3 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
2 ABORT	Abort When bit is set, a running program/erase command is aborted.
1 WAKEUP	Wakeup When bit is set, the controller wakes up from whatever low power or powerdown mode was active. If not in a powerdown mode, this bit has no effect.
0 RST	Reset When bit is set, the controller and flash are reset.

18.1.4 Auto Programming Register (AUTOPROG)

Offset

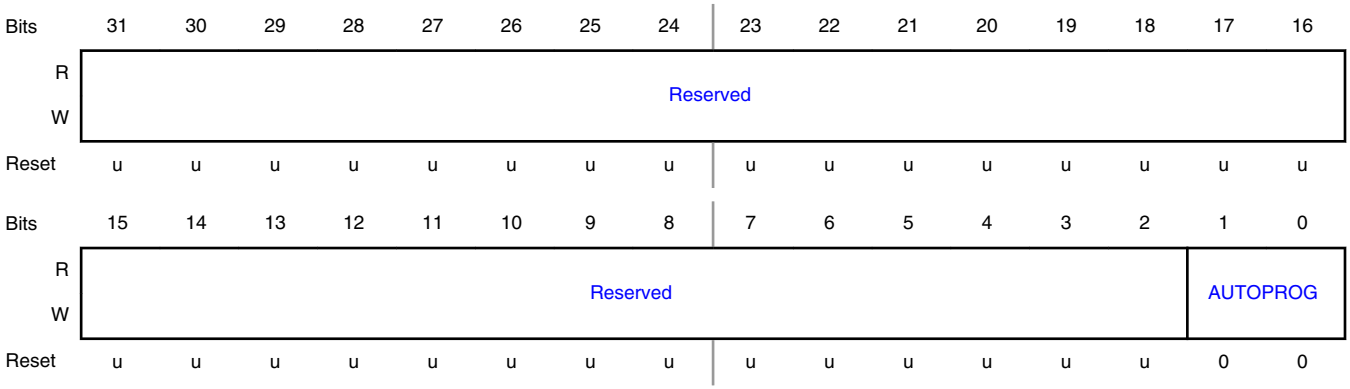
Register	Offset
AUTOPROG	Ch

Function

The auto programming register allows the user to trigger command execution automatically after an AHB write access, without the need to write to the CMD register after data has been written. Commands are executed independently of whether an AHB write cycle or an APB DATAWx register write was used to specify write data.

An auto command is triggered when writing the MSB of the memory word, [S-AWCMD]. This is because the memory word is wider than 32 bits, and must be written through multiple bus writes

Diagram



Fields

Field	Function
31-2 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
1-0 AUTOPROG	Auto Programmings Configuration 00b - Auto programming switched off. 01b - Execute write word. 10b - Execute write word then, if the last word in a page was written, program page. 11b - Reserved for future use / no action.

18.1.5 Start Address for Next flash Command Register (STARTA)

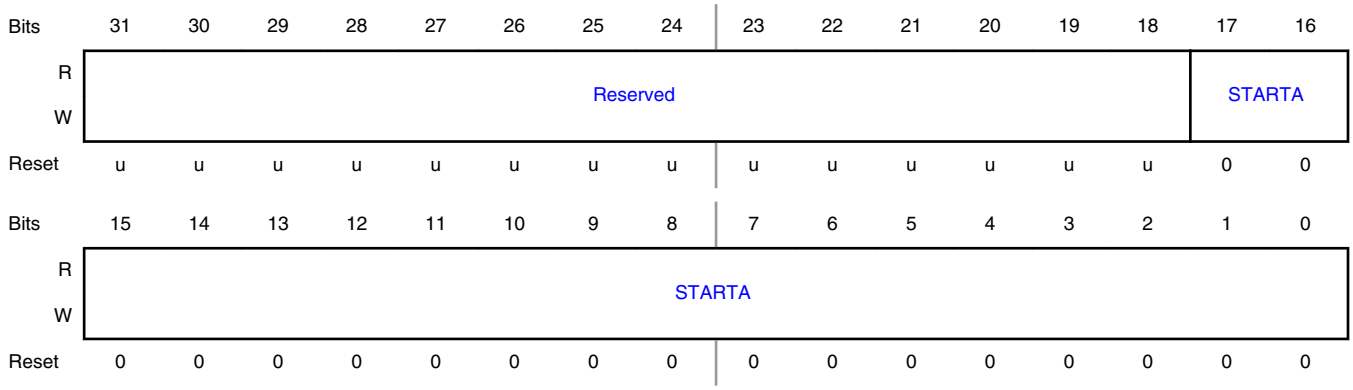
Offset

Register	Offset
STARTA	10h

Function

Address / Start address for commands that take an address (range) as a parameter. The address is in units of memory words, not bytes.

Diagram



Fields

Field	Function
31-18 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
17-0 STARTA	Address / Start Address Address / Start address for commands that take an address (range) as a parameter. The address is in units of memory words, not bytes.

18.1.6 End Address for Next Flash Command Register (STOPA)

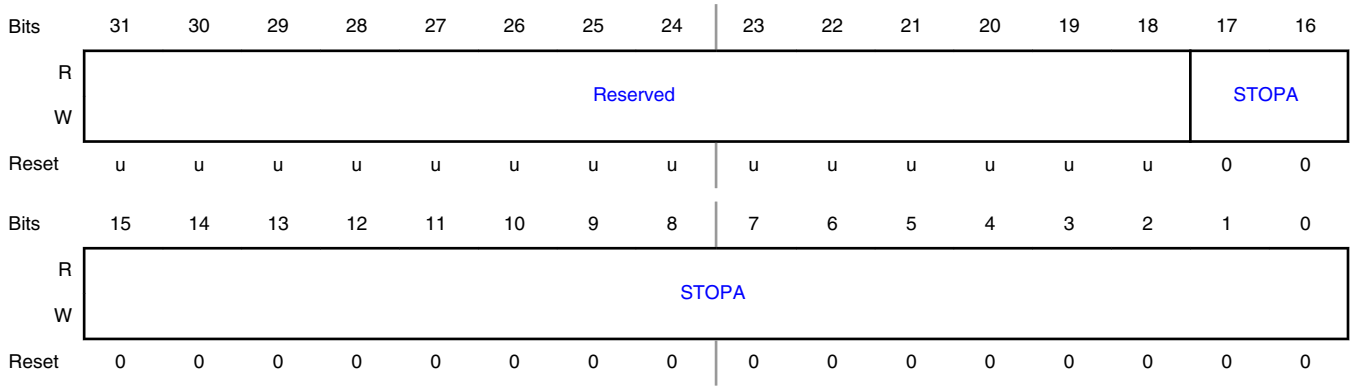
Offset

Register	Offset
STOPA	14h

Function

Stop address for commands that take an address range as a parameter (the word specified by STOPA is included in the address range). The address is in units of memory words, not bytes.

Diagram



Fields

Field	Function
31-18 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
17-0 STOPA	Stop Address for Commands Stop address for commands that take an address range as a parameter (the word specified by STOPA is included in the address range). The address is in units of memory words, not bytes.

18.1.7 Data Register a (DATAW0 - DATAW3)

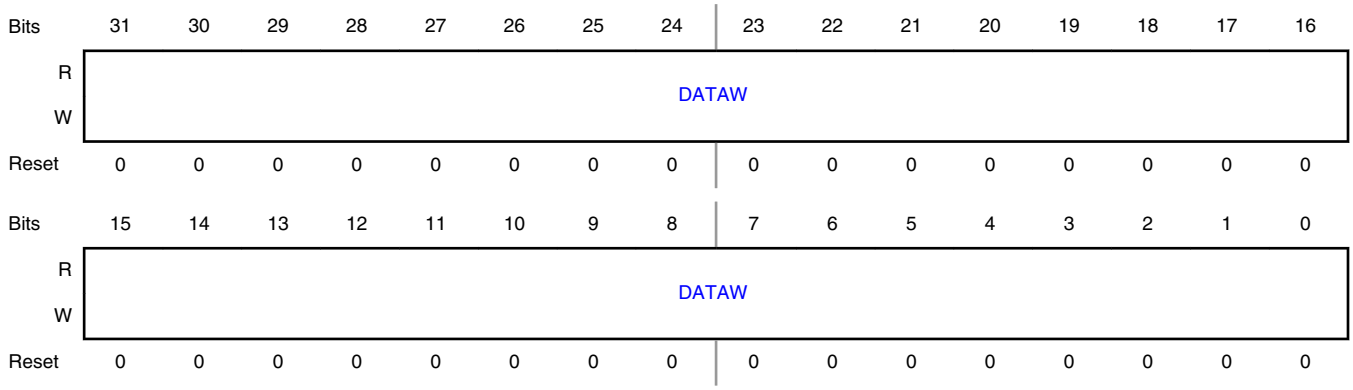
Offset

Register	Offset
DATAW0	80h
DATAW1	84h
DATAW2	88h
DATAW3	8Ch

Function

Data register for transfer of data, command parameter or command result up to 128 bits.

Diagram



Fields

Field	Function
31-0 DATAW	DATAW

18.1.8 Clear Interrupt Enable Register (INT_CLR_ENABLE)

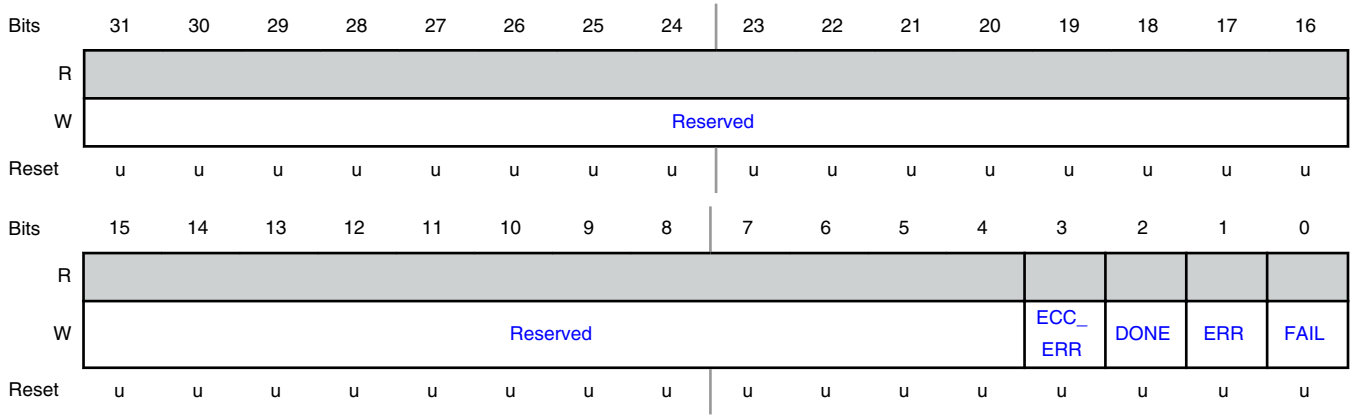
Offset

Register	Offset
INT_CLR_ENABLE	FD8h

Function

Used to clear interrupt enable bits. When a INT_CLR_ENABLE bit is written to 1, the corresponding INT_ENABLE bit is cleared.

Diagram



Fields

Field	Function
31-4 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
3 ECC_ERR	ECC_ERR Clear When a 1 is written to this field, the corresponding INT_ENABLE[ECC_ERR] bit is cleared.
2 DONE	DONE Clear When a 1 is written to this field, the corresponding INT_ENABLE[DONE] bit is cleared.
1 ERR	ERR Clear When a 1 is written to this field, the corresponding INT_ENABLE[ERR] bit is cleared.
0 FAIL	FAIL Clear When a 1 is written to this field, the corresponding INT_ENABLE[FAIL] bit is cleared.

18.1.9 Set Interrupt Enable Register (INT_SET_ENABLE)

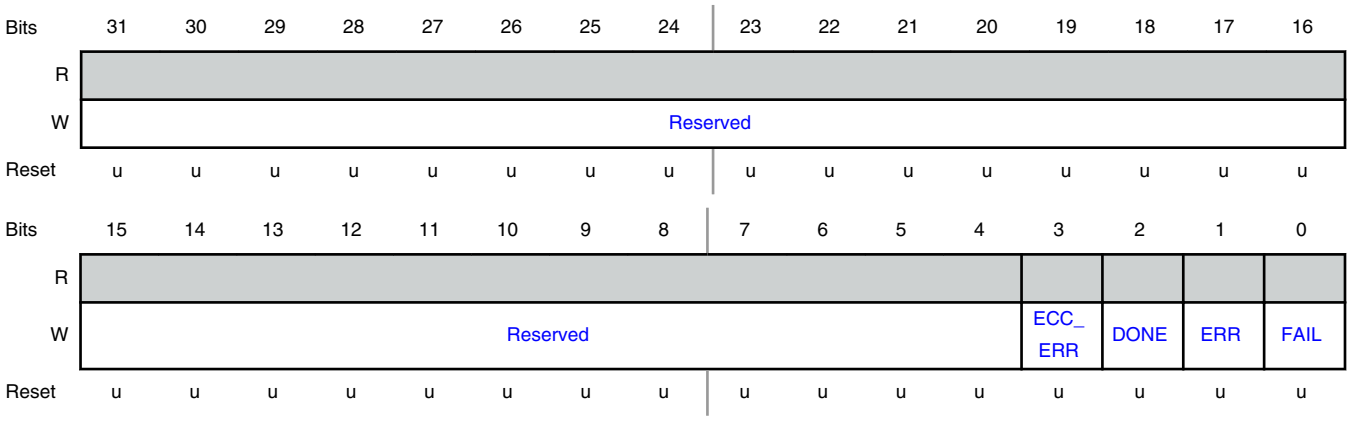
Offset

Register	Offset
INT_SET_ENABLE	FDCh

Function

Used to set interrupt enable bits. When a INT_SET_ENABLE bit is written to 1, the corresponding INT_ENABLE bit is set.

Diagram



Fields

Field	Function
31-4 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
3 ECC_ERR	ECC_ERR Set When a 1 is written to this field, the corresponding INT_ENABLE[ECC_ERR] bit is set
2 DONE	DONE Set When a 1 is written to this field, the corresponding INT_ENABLE[DONE] bit is set
1 ERR	ERR Set When a 1 is written to this field, the corresponding INT_ENABLE[ERR] bit is set
0 FAIL	FAIL Set When a 1 is written to this field, the corresponding INT_ENABLE[FAIL] bit is set

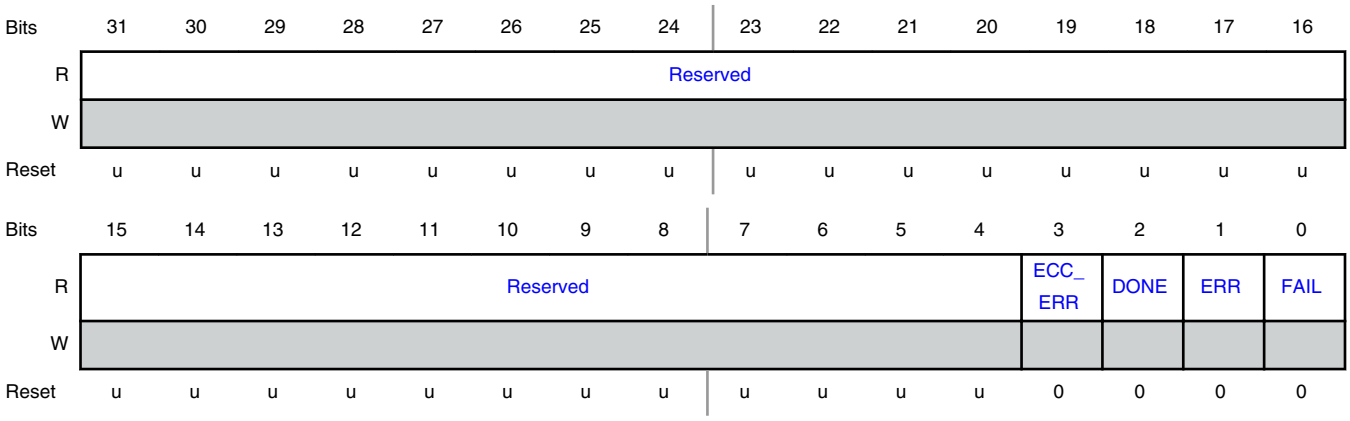
18.1.10 Interrupt Status Register (INT_STATUS)

Offset

Register	Offset
INT_STATUS	FE0h

Function
Used to read the current status of the interrupt status bits.

Diagram



Fields

Field	Function
31-4 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
3 ECC_ERR	ECC_ERR Status This status bit is set if, during a memory read operation (either a user-requested read, or a speculative read, or reads performed by a controller command), a correctable or uncorrectable error is detected by ECC decoding logic.
2 DONE	Done Status This status bit is set at the end of command execution
1 ERR	ERR Status This status bit is set if execution of an illegal command is detected. A command is illegal if it is unknown, or it is not allowed in the current mode, or it is violating access restrictions, or it has invalid parameters.
0 FAIL	FAIL Status This status bit is set if execution of a (legal) command failed. The flag can be set at any time during command execution, not just at the end.

18.1.11 Interrupt Enable Register (INT_ENABLE)

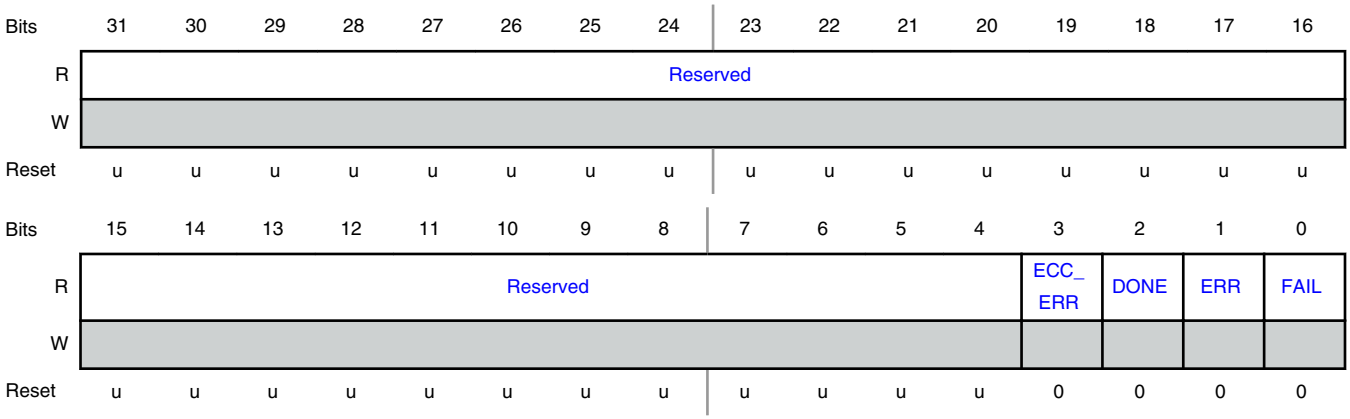
Offset

Register	Offset
INT_ENABLE	FE4h

Function

Used to read the current interrupt enable bits. An interrupt is generated where an interrupt is enabled and the corresponding status bit (INT_STATUS) is also set.

Diagram



Fields

Field	Function
31-4 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
3 ECC_ERR	ECC_ERR If this field is set, an interrupt request will be generated if the corresponding INT_STATUS[ECC_ERR] bit is high.
2 DONE	DONE If this field is set, an interrupt request will be generated if the corresponding INT_STATUS[DONE] bit is high.
1 ERR	ERR If this field is set, an interrupt request will be generated if the corresponding INT_STATUS[ERR] bit is high.
0 FAIL	FAIL If this field is set, an interrupt request will be generated if the corresponding INT_STATUS[FAIL] bit is high.

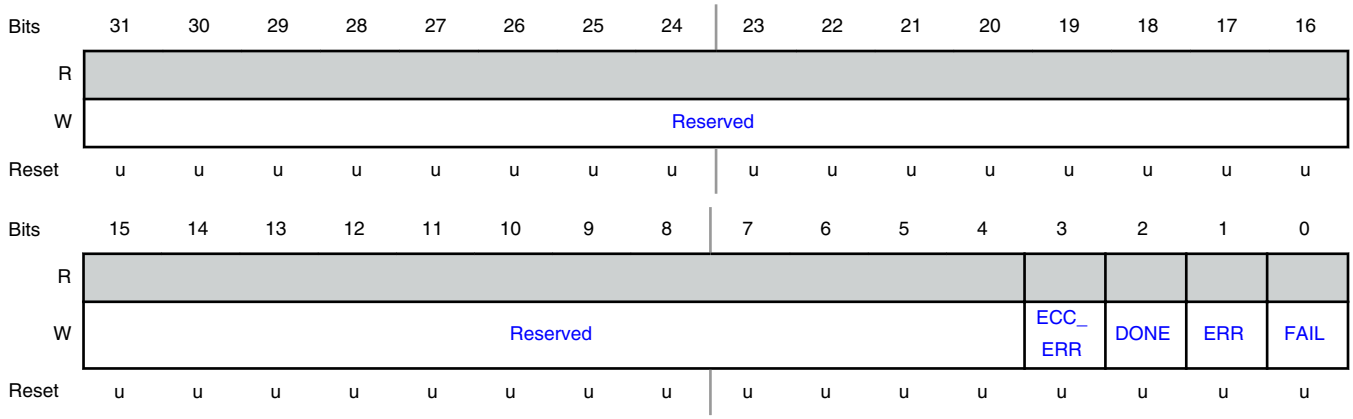
18.1.12 Clear Interrupt Status Register (INT_CLR_STATUS)

Offset

Register	Offset
INT_CLR_STATUS	FE8h

Function
Used to clear interrupt status bits. When a INT_CLR_STATUS bit is written to 1, the corresponding INT_STATUS bit is cleared.

Diagram



Fields

Field	Function
31-4 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
3 ECC_ERR	ECC_ERR Status Clear When a 1 is written to this field, the corresponding INT_STATUS[ECC_ERR] bit is cleared
2 DONE	DONE Status Clear When a 1 is written to this field, the corresponding INT_STATUS[DONE] bit is cleared
1 ERR	ERR Status Clear When a 1 is written to this field, the corresponding INT_STATUS[ERR] bit is cleared
0 FAIL	FAIL Status Clear When a 1 is written to this field, the corresponding INT_STATUS[FAIL] bit is cleared

18.1.13 Set Interrupt Status Register (INT_SET_STATUS)

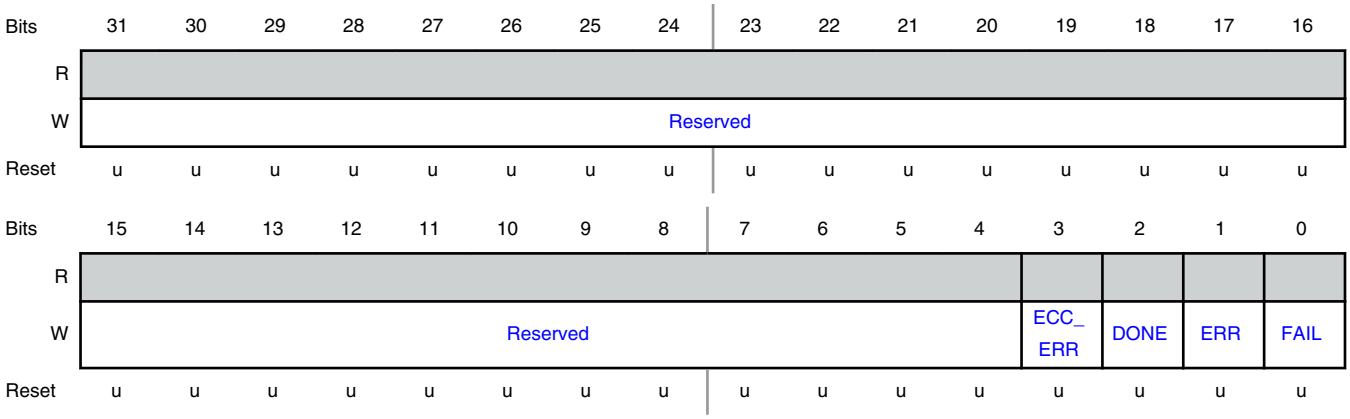
Offset

Register	Offset
INT_SET_STATUS	FECh

Function

Used to set interrupt status bits. When a INT_SET_STATUS bit is written to 1, the corresponding INT_STATUS bit is set. This can be used to test software by creating status conditions.

Diagram



Fields

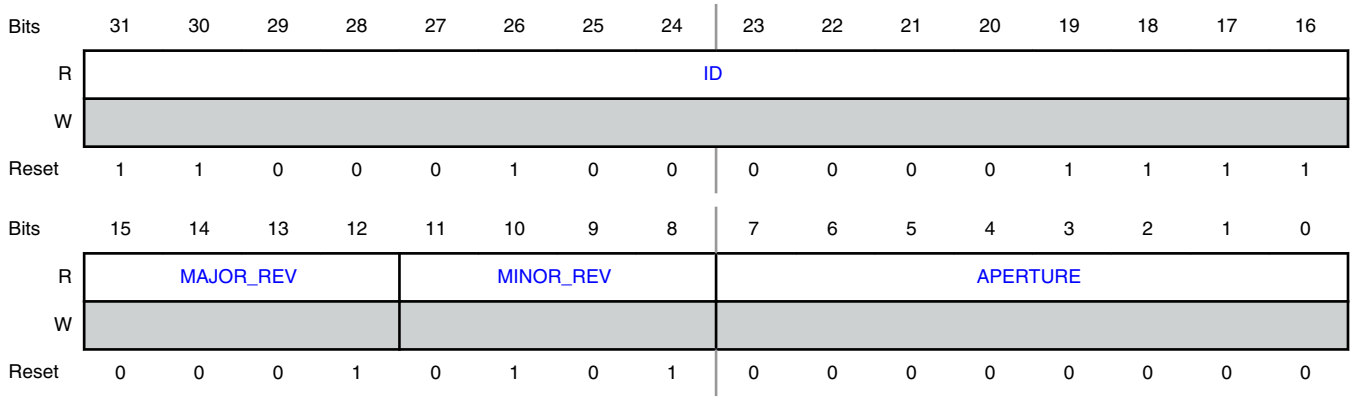
Field	Function
31-4 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
3 ECC_ERR	ECC_ERR Status Set When a 1 is written to this field, the corresponding INT_STATUS[ECC_ERR] bit is set
2 DONE	DONE Status Set When a 1 is written to this field, the corresponding INT_STATUS[DONE] bit is set
1 ERR	ERR Status Set When a 1 is written to this field, the corresponding INT_STATUS[ERR] bit is set
0 FAIL	FAIL Status Set When a 1 is written to this field, the corresponding INT_STATUS[FAIL] bit is set

18.1.14 Controller and Memory Module Identification Register (MODULE_ID)

Offset

Register	Offset
MODULE_ID	FFCh

Diagram



Fields

Field	Function
31-16 ID	Identifier This is the unique identifier of the module.
15-12 MAJOR_REV	Major Revision Major revision implies software modifications
11-8 MINOR_REV	Minor Revision Minor revision with no software consequences
7-0 APERTURE	Aperture Aperture number minus 1 of consecutive packets 4 Kbytes reserved for this IP

Chapter 19
Hash-Crypt Peripheral for SHA1, SHA2 (HASH)

19.1 HASH register descriptions

19.1.1 HASH memory map

HASH base address: 4008_F000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Control Register (CTRL)	32	RW	See description
4h	Status Regsiter (STATUS)	32	RO	See description
8h	Interrupt Enable and Set Register (INTENSET)	32	RW	See description
Ch	Interrupt Clear Register (INTENCLR)	32	WO	See description
10h	Setup Master to access Memory Register (MEMCTRL)	32	RW	See description
14h	Address to Start Memory Access Register (MEMADDR)	32	RW	0000_0000h
20h - 3Ch	Input Data Register a (INDATA0 - INDATA7)	32	WO	See description
40h - 5Ch	DIGESTa or OUTDa Register (DIGEST0 - DIGEST7)	32	RO	0000_0000h
90h	Mask register (MASK)	32	RW	0000_0000h
FFCh	IP Identifier (ID)	32	RO	EA00_0000h

19.1.2 Control Register (CTRL)

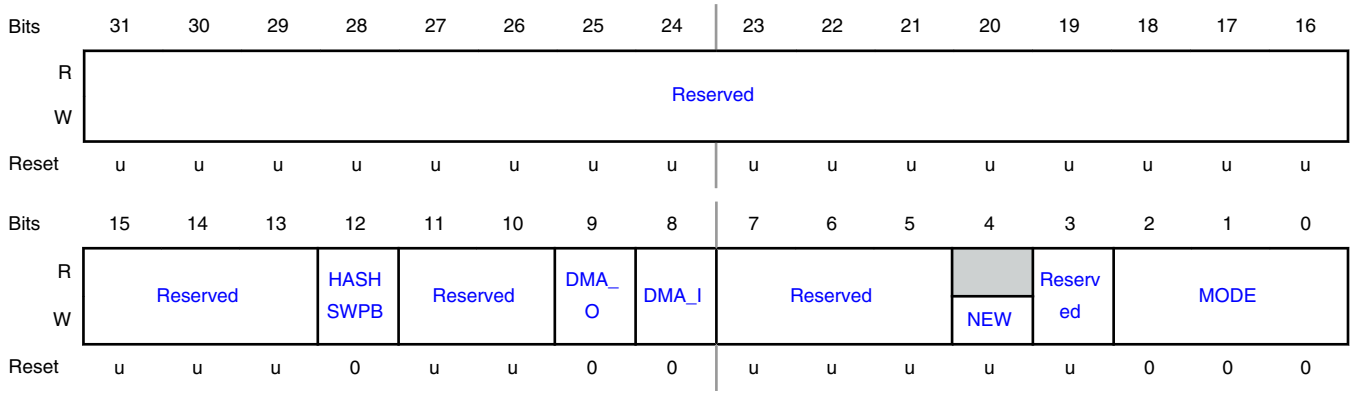
Offset

Register	Offset
CTRL	0h

Function

This register is configured with the operation to perform. The block is enabled once the MODE is selected to any of the available engines (e.g. SHA1, SHA2-256). The operational use is to Write the NEW field to 1, and then data can be pumped into INDATA (and/or its aliases) register.

Diagram



Fields

Field	Function
31-13 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
12 HASHSWPB	Swap Bytes for SHA Hashing If 1, it swaps bytes in the word for SHA hashing. The default is byte order and LSB is the first byte, but this allows swapping to MSB first.
11-10 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
9 DMA_O	Use DMA to drain Digest/output Written 1 to use DMA to drain Digest/output. If both DMA_I and DMA_O are set, the application must manage the configuration of the DMA to ensure the two DMAs are correctly configured. If written to 0 the DMA is not used and the processor has to read the digest/output in response to the DIGEST interrupt.
8 DMA_I	Use DMA to Fill INDATA Written 1 to use DMA to Fill INDATA. For Hash, 16 words will be requested from the DMA and then these will be processed. Normal model is that the DMA interrupts the processor when its length expires.
7-5 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
4 NEW	Starting New Hash Written 1 to start new hash. It self clears. Note, following the setting of NEW, the WAITING Status bit will clear for a cycle while some initialization occurs.

Table continues on the next page...

Table continued from the previous page...

Field	Function
3 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
2-0 MODE	Operational Mode 000b - Disabled 001b - SHA1 010b - SHA2-256 Others - Not valid

19.1.3 Status Regsiter (STATUS)

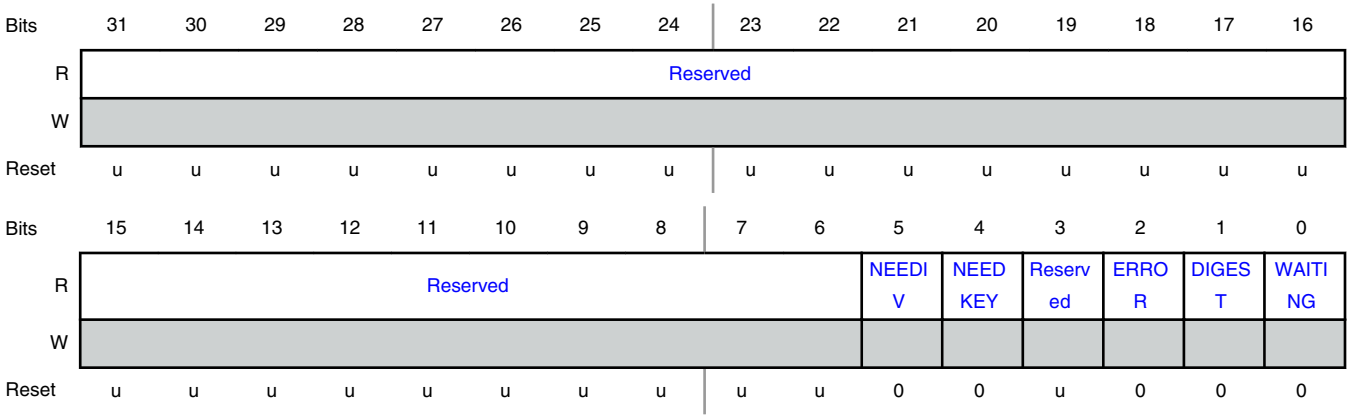
Offset

Register	Offset
STATUS	4h

Function

The status register shows when the block is waiting for data and when it has results (which may be partial). These bits correspond to both interrupts and DMA (in the case of data).

Diagram



Fields

Field	Function
31-6 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
5 NEEDIV	Block Need IV/NONE Indicate the block wants an IV/NONE to be written in (set along with WAITING). 0b - No IV is needed, because either written already or not needed. 1b - IV needed and INDATA will be accepted as IV.
4 NEEDKEY	Block Need Key Indicate the block wants the key to be written in (set along with WAITING). 0b - No key is needed and writes will not be treated as Key. 1b - Key is needed and INDATA will be accepted as Key. Will also set WAITING.
3 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
2 ERROR	Error If 1, an error occurred. For normal uses, this is due to an attempted overrun: INDATA was written when it was not appropriate. Write 1 to clear.
1 DIGEST	DIGEST Ready If 1, a DIGEST is ready and waiting and there is no active next block already started. This is cleared when any data is written, when NEW is written, or when the block is disabled.
0 WAITING	Waiting Status 0b - Not waiting for data to be written in. Also may indicate if activity has been disabled or if the block is busy. Note that for cryptographic uses, this is not set if IsLast is set nor will it set until at least 1 word is read of the output. 1b - Waiting for data to be written in (16 words)

19.1.4 Interrupt Enable and Set Register (INTENSET)

Offset

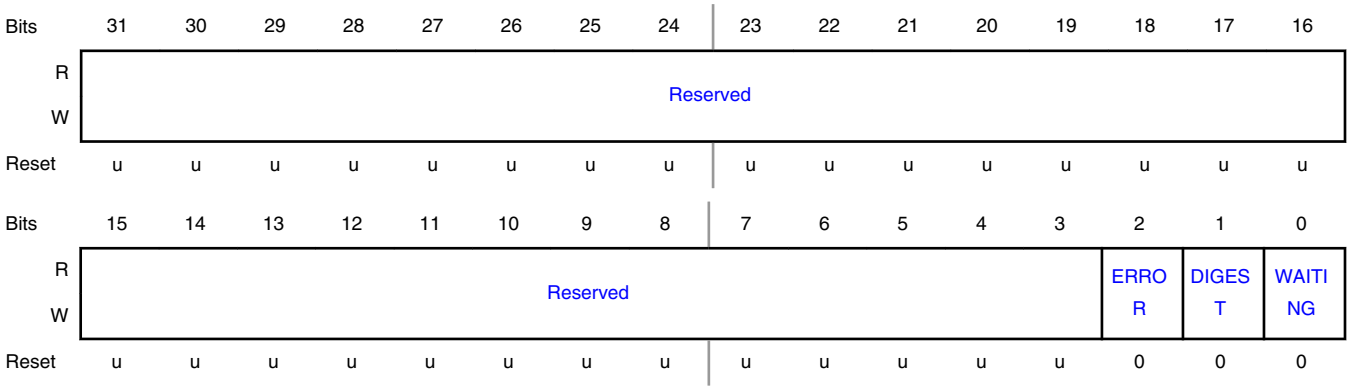
Register	Offset
INTENSET	8h

Function

This register is used to mask-enable interrupt sources to cause processor interrupts. The interrupts can be used for DMA operations or controls over registers.

Write a 1 to a valid field to enable that interrupt. When reading the register it will show which interrupts are enabled. To clear interrupt enables use the INTENCLR register.

Diagram



Fields

Field	Function
31-3 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
2 ERROR	Error 0b - Not interrupt on Error. 1b - Interrupt on Error. Write 1 to set this bit.
1 DIGEST	Digest 0b - Not interrupt when Digest is ready. 1b - Interrupt when Digest is ready. Write 1 to set this bit.
0 WAITING	Waiting 0b - Not interrupt when waiting. 1b - Interrupt when waiting. Write 1 to set this bit.

19.1.5 Interrupt Clear Register (INTENCLR)

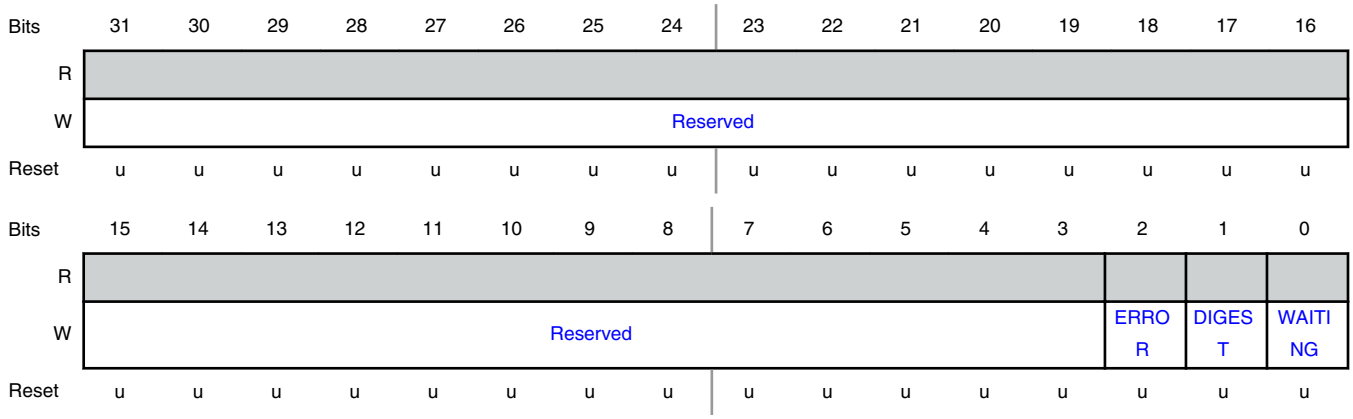
Offset

Register	Offset
INTENCLR	Ch

Function

The Interrupt Clear register is used to clear interrupts mask-enabled by the INTENSET register.

Diagram



Fields

Field	Function
31-3 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
2 ERROR	Error Write 1 to clear correspnding bit in INTENSET.
1 DIGEST	Digest Write 1 to clear correspnding bit in INTENSET.
0 WAITING	Waiting Write 1 to clear correspnding bit in INTENSET.

19.1.6 Setup Master to access Memory Register (MEMCTRL)

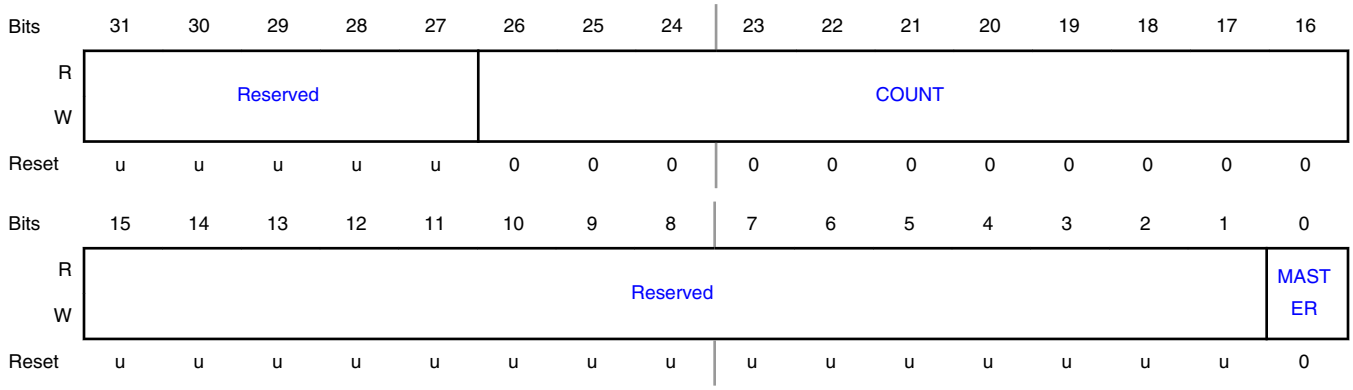
Offset

Register	Offset
MEMCTRL	10h

Function

The MEMCTRL register allows setting up the Hashing block to master the bus to read memory for hashing. It can be used to read 512-bit blocks from Flash or RAM (or whatever system allows) for hashing. The starting location must be word aligned and the length may be up to 128 KB (2K blocks).

Diagram



Fields

Field	Function
31-27 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
26-16 COUNT	Count Number of 512-bit blocks to copy starting at MEMADDR. This field will decrement after each block is copied, ending in 0. For Hash, the DIGEST interrupt will occur when it reaches 0. If a bus error occurs, it will stop with this field set to the block that failed 0x0 - Done, nothing to process 0x1-0x3FF - Number of blocks to hash
15-1 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
0 MASTER	Enables Mastering 0b - Mastering is not used and the normal DMA or Interrupt based model is used with INDATA. 1b - Mastering is enabled and neither of the DMA or INDATA is used.

19.1.7 Address to Start Memory Access Register (MEMADDR)

Offset

Register	Offset
MEMADDR	14h

Function

The MEMADDR register holds the base address for MEMCTRL. It must only point to valid locations per the part (eg. Flash and one or more of the RAMs) and must be word aligned.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BASEADDR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BASEADDR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Function
31-0 BASEADDR	Base Address Address base to start copying from, word aligned (so bits 1:0 must be 0). This field will advance as it processes the words. If it fails with a bus error, the register will contain the failing word.

19.1.8 Input Data Register a (INDATA0 - INDATA7)

Offset

For a = 0 to 7:

Register	Offset
INDATAa	20h + (a × 4h)

Function

Input of 16 words at a time to load up buffer. This register is aliased 16 times in the offset range 0x20 to 0x3C which may allow for optimised writing of the data, e.g. using a Store multiple (STM) instruction.

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	DATA															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	DATA															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Function
31-0 DATA	Data Write next word in little-endian form. The hash requires big endian word data, but this block swaps the bytes automatically. That is, SHA assumes the data coming in is treated as bytes (e.g. abcd) and since the Arm core treats abcd as a word as 0x64636261, the block will swap the word to restore into big endian

19.1.9 DIGESTa or OUTDa Register (DIGEST0 - DIGEST7)

Offset

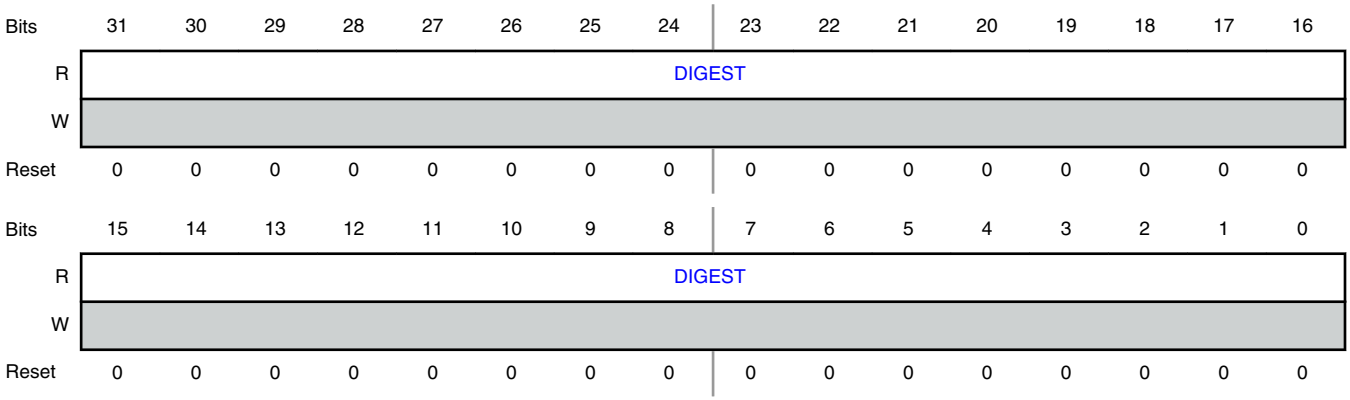
For a = 0 to 7:

Register	Offset
DIGESTa	40h + (a × 4h)

Function

DIGEST or OUTD0, 5 or 8 bytes of output data, depending upon mode. This register gives access to up to 256 bits of DIGEST/ output data at offset from 0x40 to 0x5C.

Diagram



Fields

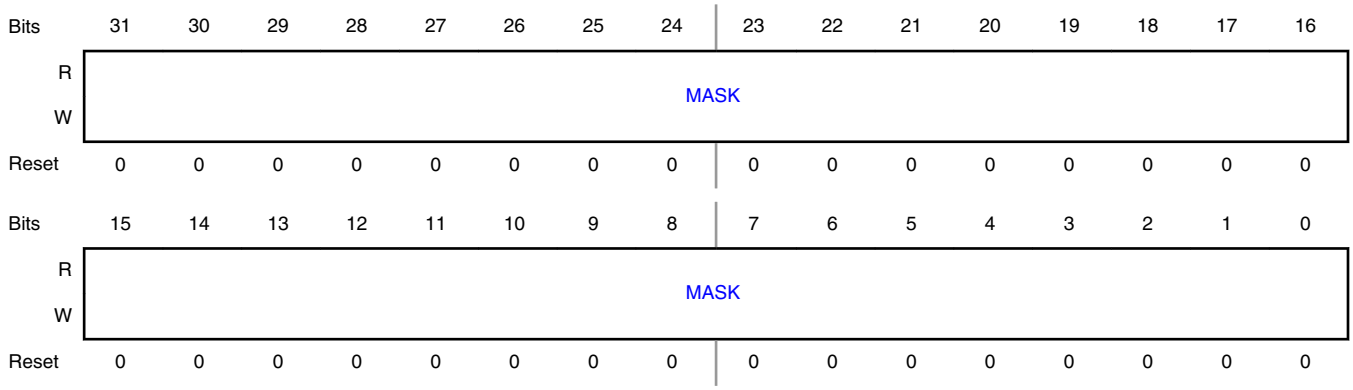
Field	Function
31-0 DIGEST	8 Entry DIGEST/OUTPUT Array All 256 bits are valid for SHA2-256. Only 160 bits are valid for SHA1.

19.1.10 Mask register (MASK)

Offset

Register	Offset
MASK	90h

Diagram



Fields

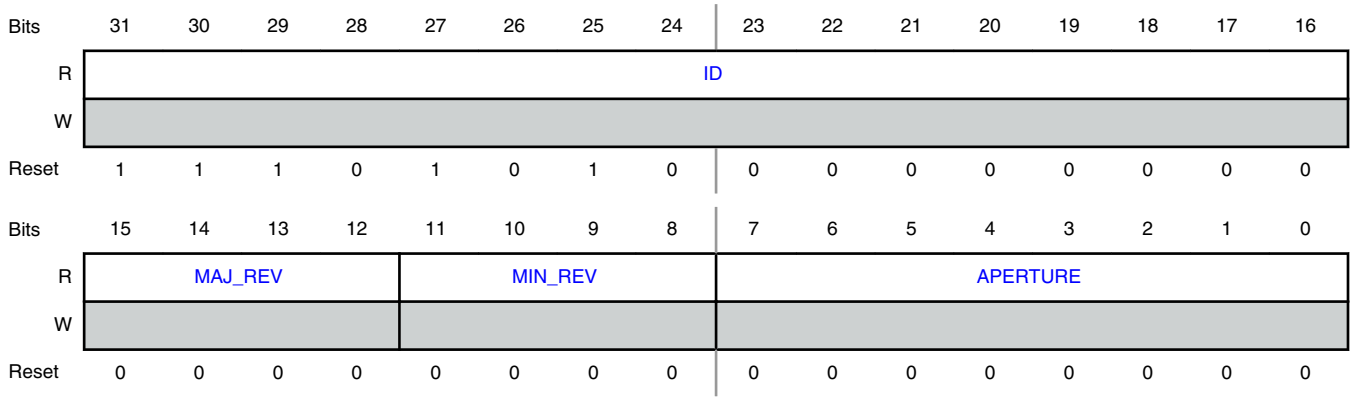
Field	Function
31-0 MASK	MASK Reserved, do not modify this register.

19.1.11 IP Identifier (ID)

Offset

Register	Offset
ID	FFCh

Diagram



Fields

Field	Function
31-16 ID	Identifier This is the unique identifier of the module
15-12 MAJ_REV	Major Revision Major revision implies software modifications
11-8 MIN_REV	Minor Revision Minor revision with no software consequences
7-0 APERTURE	Aperture Aperture number minus 1 of consecutive packets 4 KB reserved for this IP

Chapter 20
SPI Flash Interface (SPIFI)

20.1 SPIFI register descriptions

20.1.1 SPIFI memory map

SPIFI base address: 4008_4000h

Offset	Register	Width (In bits)	Access	Reset value
0h	SPIFI Control Register (CTRL)	32	RW	See description
4h	SPIFI Command Register (CMD)	32	RW	0000_0000h
8h	SPIFI Address Register (ADDR)	32	RW	0000_0000h
Ch	SPIFI Intermediate Data Register (IDATA)	32	RW	0000_0000h
10h	SPIFI Cache Limit Register (CLIMIT)	32	RW	0800_0000h
14h	SPIFI Data Register (DATA)	32	RW	0000_0000h
18h	SPIFI Memory Command Register (MCMD)	32	RW	See description
1Ch	SPIFI Status Register (STAT)	32	RW	See description

20.1.2 SPIFI Control Register (CTRL)

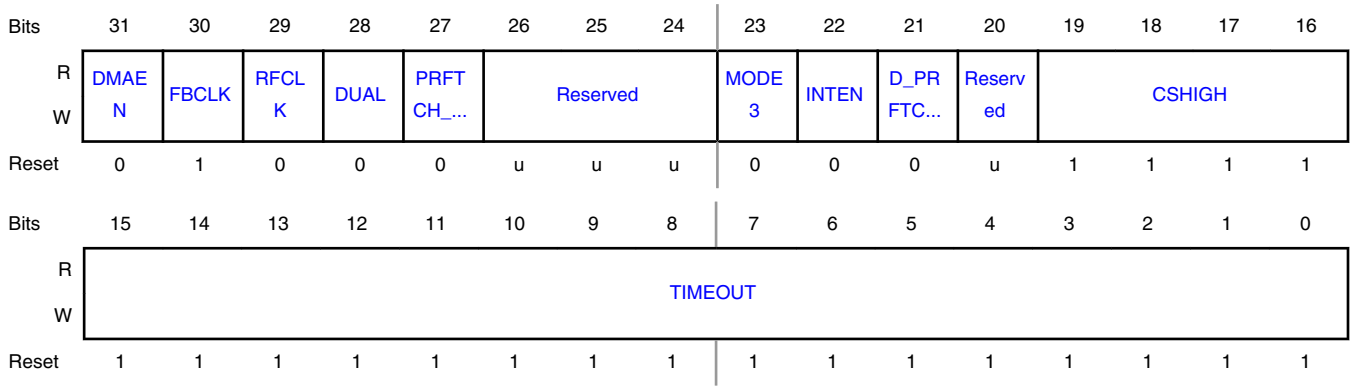
Offset

Register	Offset
CTRL	0h

Function

This register controls the overall operation of the SPIFI, and should be written before any commands are initiated.

Diagram



Fields

Field	Function
31 DMAEN	Enable DMA Request Output A 1 in this bit enables the DMA Request output from the SPIFI. Set this bit only when a DMA channel is used to transfer data in peripheral mode. Do not set this bit when a DMA channel is used for memory-to-memory transfers from the SPIFI memory area. DMAEN should only be used in Command mode.
30 FBCLK	Feedback Clock Select Remark: MODE3, RFCLK, and FBCLK should not all be 1, because in this case there is no final falling edge on SPIFI_CLK on which to sample the last data bit of the frame. 0b - Internal clock. The SPIFI samples read data using an internal clock. 1b - Feedback clock. Read data is sampled using a feedback clock from the SPIFI_CLK pin. This allows slightly more time for each received bit.
29 RFCLK	Select Active Clock Edge for Input Data Remark: MODE3, RFCLK, and FBCLK should not all be 1, because in this case there is no final falling edge on SPIFI_CLK on which to sample the last data bit of the frame. 0b - Rising edge. Read data is sampled on rising edges on the clock. 1b - Falling edge. Read data is sampled on falling edges of the clock, allowing a full serial clock of of time in order to maximize the serial clock frequency.
28 DUAL	Select Dual Protocol 0b - Quad protocol. This protocol uses IO3:0. 1b - Dual protocol. This protocol uses IO1:0.
27 PRFTCH_DIS	Cache Prefetching Disable The SPIFI includes an internal cache. 0b - Enable prefetching of cache lines. 1b - Disable prefetching of cache lines.

Table continues on the next page...

Table continued from the previous page...

Field	Function
26-24 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
23 MODE3	SPI Mode 3 Select Remark: MODE3, RFCLK, and FBCLK should not all be 1, because in this case there is no final falling edge on SPIFI_CLK on which to sample the last data bit of the frame. 0b - SPIFI_CLK LOW. The SPIFI drives SPIFI_CLK low after the rising edge at which the last bit of each command is captured, and keeps it low while SPIFI_CSN is HIGH. 1b - SPIFI_CLK HIGH. the SPIFI keeps SPIFI_CLK high after the rising edge for the last bit of each command and while SPIFI_CSN is HIGH, and drives it low after it drives SPIFI_CSN LOW. (Known serial flash devices can handle either mode, but some devices may require a particular mode for proper operation.)
22 INTEN	Assert Interrupt Request Output If this bit is 1 when a command ends, the SPIFI will assert its interrupt request output. See STAT[INTRQ] for more details.
21 D_PRFTCH_DIS	Prefetch Conditioning This bit allows conditioning of memory mode prefetches based on the AHB HPROT (instruction/data) access information. A 1 in this field means that the SPIFI will not attempt a speculative prefetch when it encounters data accesses.
20 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
19-16 CSHIGH	Minimum SPIFI_SCN High Time Control This field controls the minimum SPIFI_SCN high time, expressed as a number of serial clock periods minus one.
15-0 TIMEOUT	Timeout This field contains the number of serial clock periods without the processor reading data in memory mode, which will cause the SPIFI hardware to terminate the command by driving the SPIFI_CSN pin high and negating the STAT[CMD] bit. (This allows the flash memory to enter a lower-power state.) If the processor reads data from the flash region after a time-out, the command in the Memory Command (MCMD) Register is issued again.

20.1.3 SPIFI Command Register (CMD)

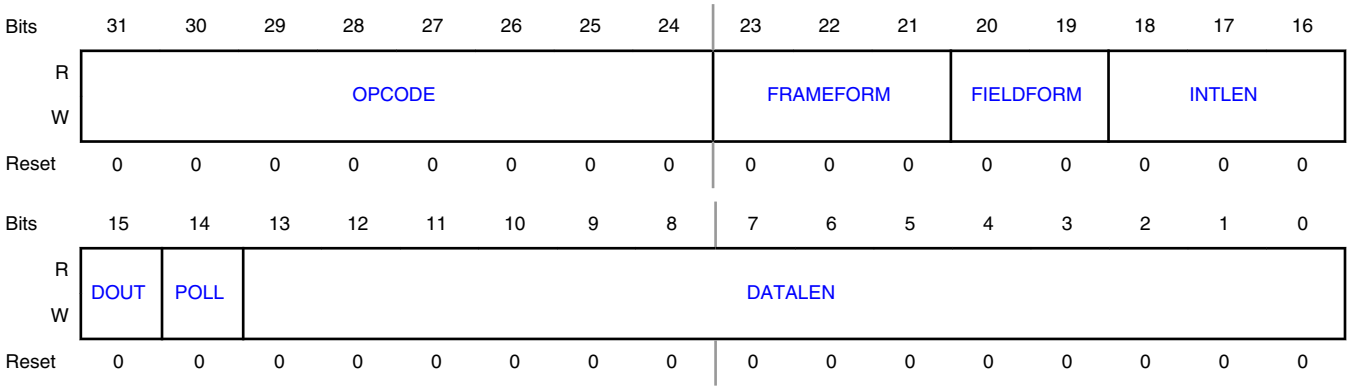
Offset

Register	Offset
CMD	4h

Function

This register may be written to when the STAT[CMD] and STAT[MCINIT] bits are 0, and under these circumstances writing initiates the transmission of a new command. For a command that contains an address and/or intermediate data, software should write to the ADDR and/or IDATA Register(s) before writing to this register. If the command contains output data, software should write it to the DATA Register after writing to this register. If the command contains input data, software can read it from the DATA Register after writing to this register.

Diagram



Fields

Field	Function
31-24 OPCODE	Opcode The opcode of the command (not used for some FRAMEFORM values).
23-21 FRAMEFORM	Opcode and Address Fields Control This field controls the opcode and address fields. 000b - Reserved. 001b - Opcode. Opcode only, no address. 010b - Opcode one byte. Opcode, least significant byte of address. 011b - Opcode two bytes. Opcode, two least significant bytes of address. 100b - Opcode three bytes. Opcode, three least significant bytes of address. 101b - Opcode four bytes. Opcode, 4 bytes of address. 110b - No opcode three bytes. No opcode, 3 least significant bytes of address. 111b - No opcode four bytes. No opcode, 4 bytes of address.
20-19 FIELDFORM	Command Field Sent This field controls how the fields of the command are sent. All fields of the command are in quad/dual format. 00b - All serial. All fields of the command are serial. 01b - Quad/dual data. Data field is quad/dual, other fields are serial. 10b - Serial opcode. Opcode field is serial. Other fields are quad/dual. 11b - All quad/dual.

Table continues on the next page...

Table continued from the previous page...

Field	Function
18-16 INTLEN	Intermediate Bytes Precede Data This field controls how many intermediate bytes precede the data. (Each such byte may require 8 or 2 SPIFI_CLK cycles, depending on whether the intermediate field is in serial, 2-bit, or 4-bit format.) Intermediate bytes are output by the SPIFI, and include post-address control information, dummy and delay bytes. See the description of the Intermediate Data (IDATA) register for the contents of such bytes.
15 DOUT	Data Direction Control If the DATALEN field is not zero, this bit controls the direction of the data. 0b - Input from serial flash. 1b - Output to serial flash.
14 POLL	Poll This bit should be written as 1 only with an opcode that • contains an input data field, • causes the serial flash device to return byte status repetitively (e.g., a Read Status command). When this bit is 1, the SPIFI hardware continues to read bytes until the test specified by the DATALEN field is met. The hardware tests the bit in each status byte selected by DATALEN[2:0], until a bit is found that is equal to DATALEN bit 3. When the test succeeds, the SPIFI captures the byte that meets this test so that it can be read from the Data Register, and terminates the command by raising SPIFI_SCN. The end-of-command interrupt can be enabled to inform software when this occurs.
13-0 DATALEN	Command Data Bytes Control Except when the POLL bit in this register is 1, this field controls how many data bytes are in the command. 0 indicates that the command does not contain a data field.

20.1.4 SPIFI Address Register (ADDR)

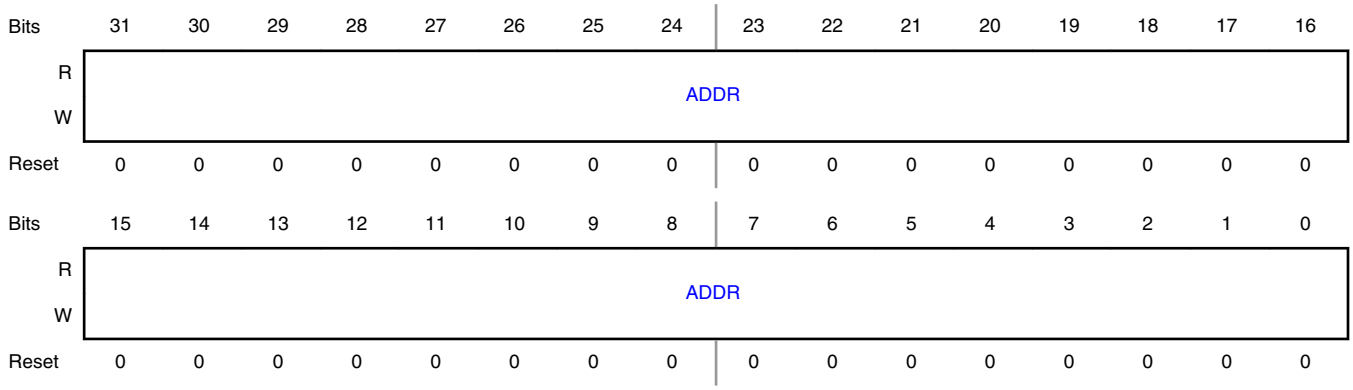
Offset

Register	Offset
ADDR	8h

Function

Before writing a command that includes an address field to the CMD register, software should write the address to this register.

Diagram



Fields

Field	Function
31-0 ADDR	Address Used as part of the SPIFI operation when required by the command in operation.

20.1.5 SPIFI Intermediate Data Register (IDATA)

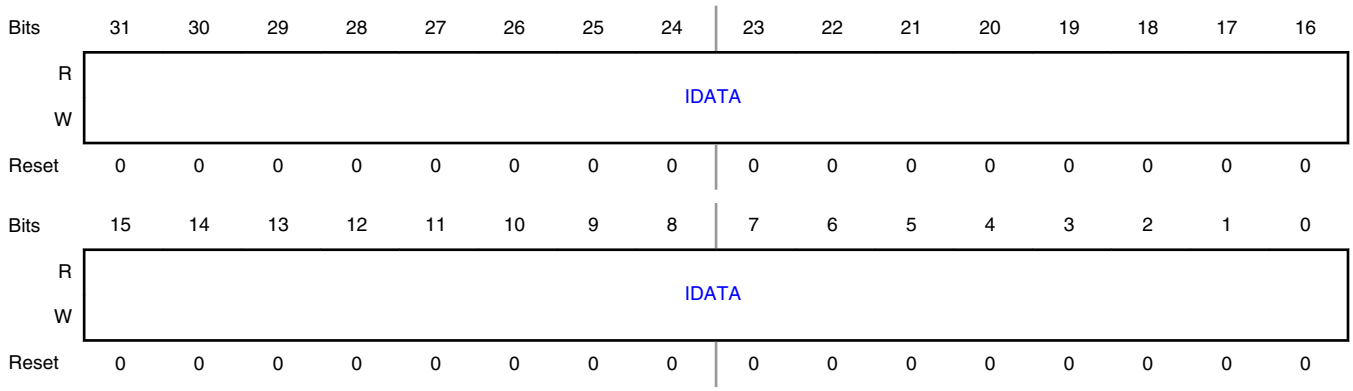
Offset

Register	Offset
IDATA	Ch

Function

This register is required with some commands.

Diagram



Fields

Field	Function
31-0 IDATA	IDATA Intermediate byte values that are output with some specific commands.

20.1.6 SPIFI Cache Limit Register (CLIMIT)

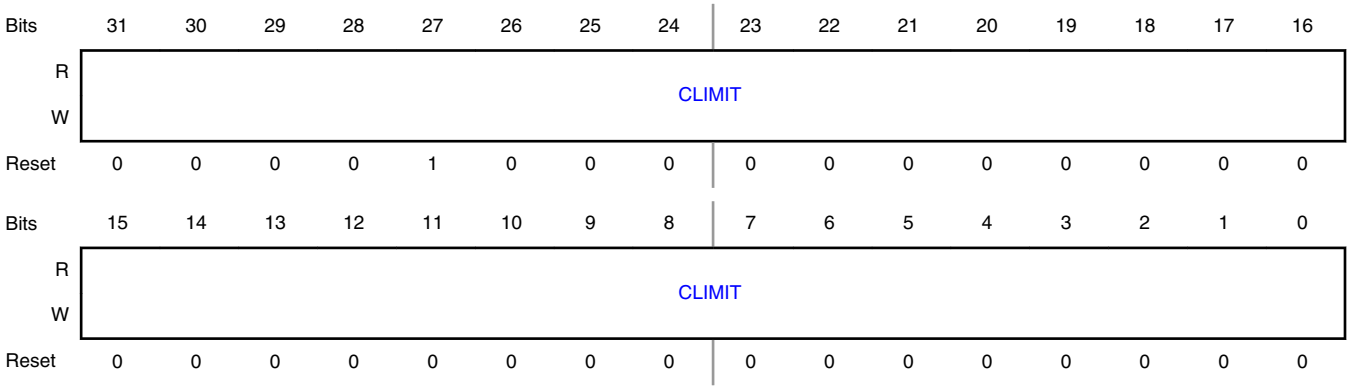
Offset

Register	Offset
CLIMIT	10h

Function

This register sets the point above which caching will not occur.

Diagram



Fields

Field	Function
31-0 CLIMIT	Cache Limit Address accesses above this setting will not get cached in the data cache.

20.1.7 SPIFI Data Register (DATA)

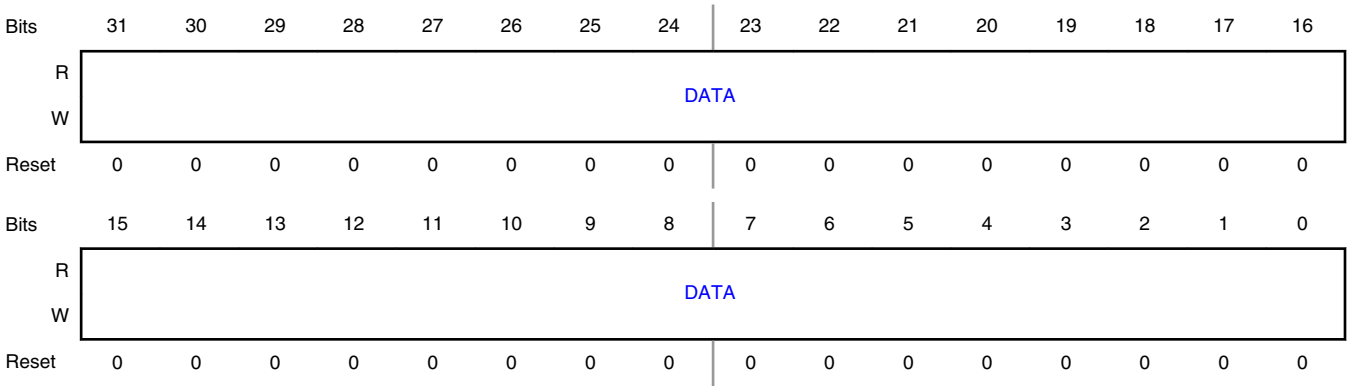
Offset

Register	Offset
DATA	14h

Function

Input or output data

Diagram



Fields

Field	Function
31-0 DATA	Data Used during the commands involving input and output data.

20.1.8 SPIFI Memory Command Register (MCMD)

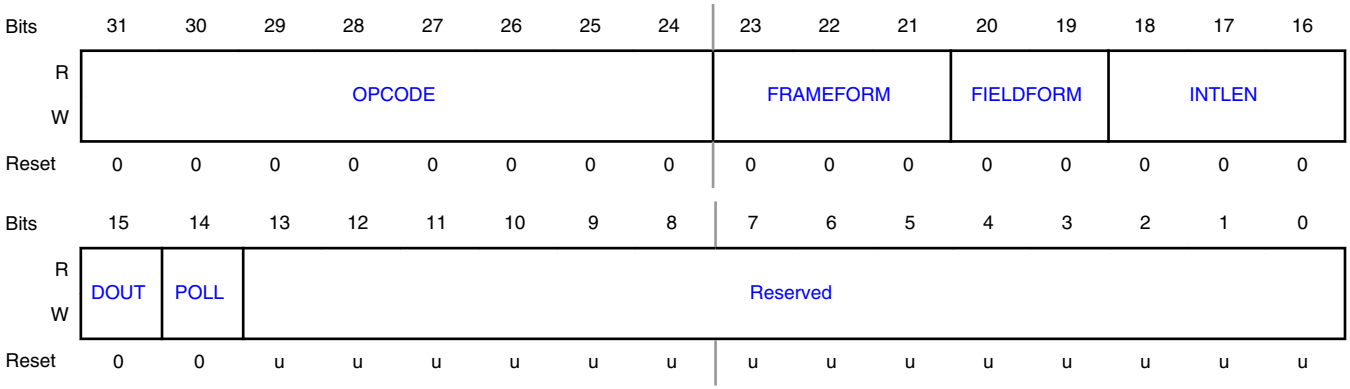
Offset

Register	Offset
MCMD	18h

Function

MCMD is used for commands that are accessing the flash memory region of the memory map.

Diagram



Fields

Field	Function
31-24 OPCODE	Opcode The opcode of the command (not used for some FRAMEFORM values).
23-21 FRAMEFORM	Opcode and Address Fields Control This field controls the opcode and address fields. 000b - Reserved. 001b - Opcode. Opcode only, no address. 010b - Opcode one byte. Opcode, least significant byte of address. 011b - Opcode two bytes. Opcode, two least significant bytes of address. 100b - Opcode three bytes. Opcode, three least significant bytes of address. 101b - Opcode four bytes. Opcode, 4 bytes of address. 110b - No opcode three bytes. No opcode, 3 least significant bytes of address. 111b - No opcode four bytes. No opcode, 4 bytes of address.
20-19 FIELDFORM	Command Field Sent This field controls how the fields of the command are sent. All fields of the command are in quad/dual format. 00b - All serial. All fields of the command are serial. 01b - Quad/dual data. Data field is quad/dual, other fields are serial. 10b - Serial opcode. Opcode field is serial. Other fields are quad/dual. 11b - All quad/dual.
18-16 INTLEN	Intermediate Bytes Precede Data This field controls how many intermediate bytes precede the data. (Each such byte may require 8 or 2 SPIFI_CLK cycles, depending on whether the intermediate field is in serial, 2-bit, or 4-bit format.) Intermediate bytes are output by the SPIFI, and include post-address control information, dummy and delay bytes. See the description of the Intermediate Data (IDATA) register for the contents of such bytes.
15 DOUT	DOUT This bit should be written as 0.
14 POLL	Poll This bit should be written as 0.
13-0 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

20.1.9 SPIFI Status Register (STAT)

Offset

Register	Offset
STAT	1Ch

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	VERSION								Reserved							
Reset	0	0	0	0	0	0	1	0	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved										INTRQ	RESET	Reserved		CMD	MCINIT
Reset	u	u	u	u	u	u	u	u	u	u	0	0	u	u	0	0

Fields

Field	Function
31-24 VERSION	Version
23-6 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
5 INTRQ	SPIFI Interrupt Request Status This bit reflects the SPIFI interrupt request. Write a 1 to this bit to clear it. This bit is set when the CMD field was previously 1 and has been cleared due to the deassertion of SPIFI_SCN.
4 RESET	Reset Status Write a 1 to this bit to abort a current command or memory mode. This bit is cleared when the hardware is ready for a new command to be written to the Command register.
3-2 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
1 CMD	Command Register Written Status This bit is 1 when the Command register is written. It is cleared by a hardware reset, a write to the RESET bit in this register, or the deassertion of SPIFI_SCN which indicates that the command has completed communication with the external SPI Flash device.
0 MCINIT	Software Write Memory Command Status This bit is set when software successfully writes the Memory Command register, and is cleared by Reset or by writing a 1 to the RESET bit in this register.

Chapter 21
True Random Number Generator (RNG)

21.1 RNG register descriptions

21.1.1 RNG memory map

RNG base address: 4000_D000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Random Number Register (RANDOM_NUMBER)	32	RO	0000_0000h
8h	Counter Value Register (COUNTER_VAL)	32	RO	See description
Ch	Counter Configure Register (COUNTER_CFG)	32	RW	See description
10h	On Line Test Configuration Register (ONLINE_TEST_CFG)	32	RW	See description
14h	Online Test Results Register (ONLINE_TEST_VAL)	32	RO	See description
FF4h	Powerdown Mode and Reset Control Register (POWERDOWN)	32	RW	See description
FFCh	IP Identifier Register (MODULEID)	32	RO	A0B8_3200h

21.1.2 Random Number Register (RANDOM_NUMBER)

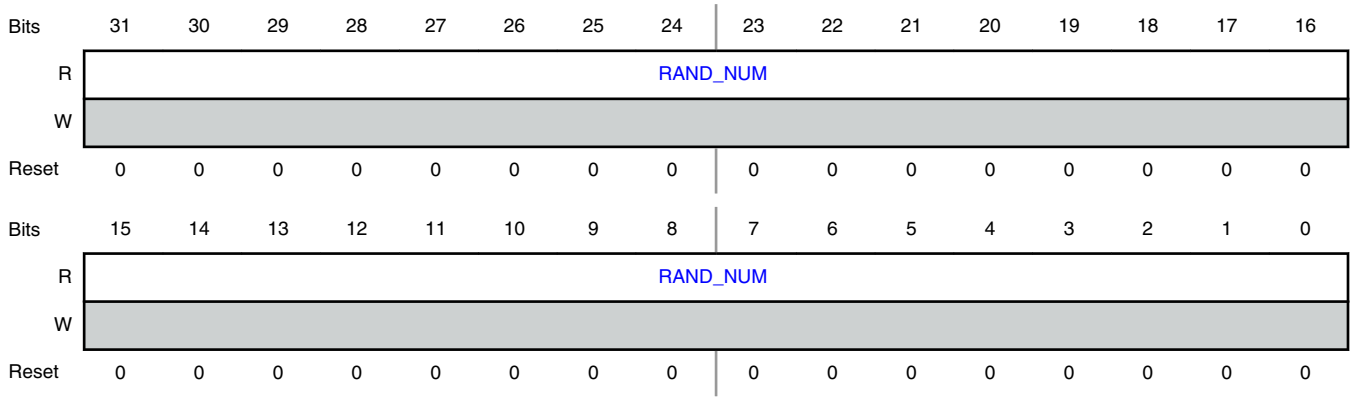
Offset

Register	Offset
RANDOM_NUMBER	0h

Function

This register is used to access the generated random numbers.

Diagram



Fields

Field	Function
31-0 RAND_NUM	Random Number This field contains a random 32-bit number which is computed on demand, at each time it is read. Weak cryptographic post-processing is used to maximize throughput. The block will start computing before the first register access and so the reset value is not relevant.

21.1.3 Counter Value Register (COUNTER_VAL)

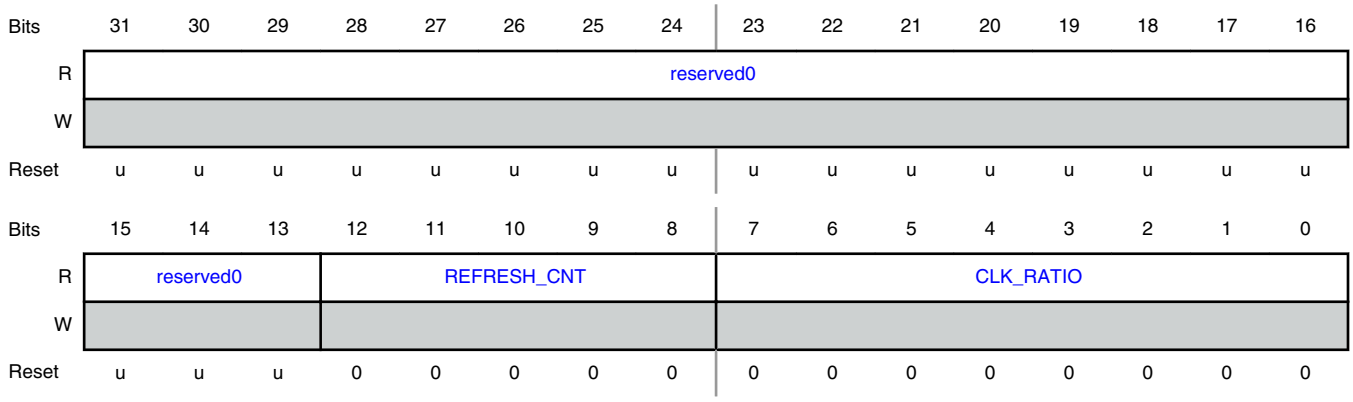
Offset

Register	Offset
COUNTER_VAL	8h

Function

This reigster shows information about the random process.

Diagram



Fields

Field	Function
31-13 reserved0	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
12-8 REFRESH_CNT	Refresh Counter Incremented (till max possible value) each time COUNTER was updated since last reading to any *_NUMBER. This gives an indication on 'entropy refill'. **NOTE** There is no linear accumulation of entropy: entropy refill is about 4 bits each time 'refresh_cnt' reaches its maximum value. See user manual for further details on how to benefit from linear entropy accumulation using a specific procedure.
7-0 CLK_RATIO	Clock Ratio This field configures the ratio between the internal clocks frequencies and the register clock frequency for evaluation and certification purposes. Internal clock frequencies are half the incoming ones: COUNTER_VAL = round[(intFreq/2)/ regFreq*256*(1<<(4*shift4x))] MODULO 256 If shitf4x==0, intFreq ~= regFreq*COUNTER_VAL/256*2. Use COUNTER_CFG[CLOCK_SEL] to select the clock to measure, in the range from 1 to 5.

21.1.4 Counter Configure Register (COUNTER_CFG)

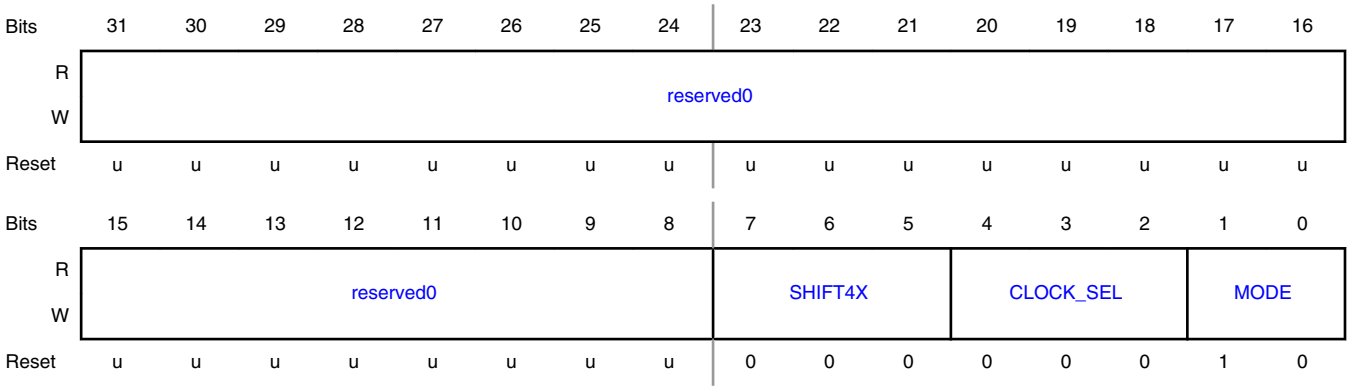
Offset

Register	Offset
COUNTER_CFG	Ch

Function

Register linked to the comupting of Statistics, not required for normal operation.

Diagram



Fields

Field	Function
31-8 reserved0	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
7-5 SHIFT4X	Shift This field is used to add precision to clock_ratio and determine 'entropy refill'. Supported range is 0-4 Used as well for ONLINE_TEST
4-2 CLOCK_SEL	Clock Selection Select the internal clock on which to compute statistics. 1 is for first one, 2 for second one, . And 0 is for a XOR of results from all clocks 000b - XOR of results from all clocks 001b - XTAL32M 010b - FRO32M 011b-111b Not valid
1-0 MODE	Mode 00b - Disabled 01b - Update once. Will return to 00 once done 10b - Free running: updates countinuously 11b - Reserved

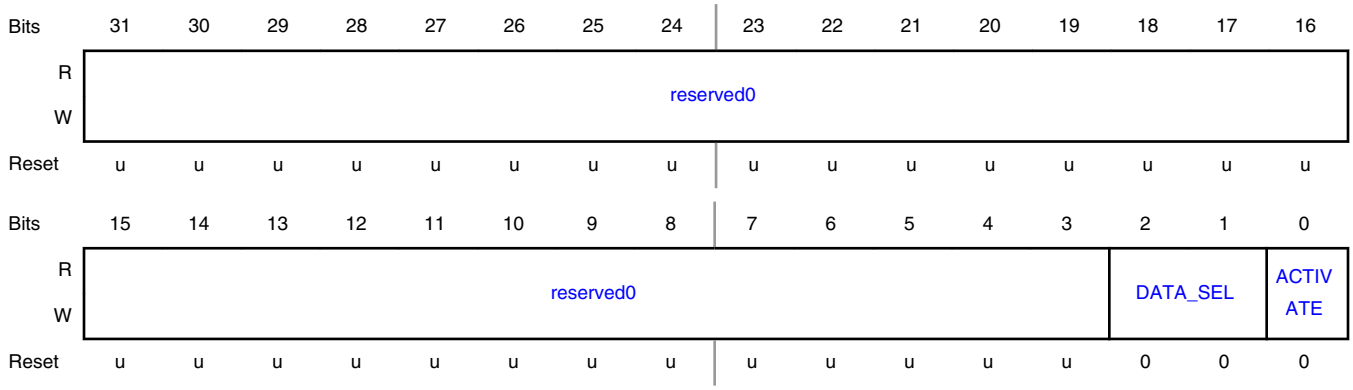
21.1.5 On Line Test Configuration Register (ONLINE_TEST_CFG)

Offset

Register	Offset
ONLINE_TEST_CFG	10h

Function
This register configures settings used in performing the measurements for randomness.

Diagram



Fields

Field	Function
31-3 reserved0	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
2-1 DATA_SEL	Data Select Selects source on which to apply online test. 00b - LSB of COUNTER: raw data from one or all sources of entropy. ACTIVATE field should be set to 'Disabled' before changing this field. 01b - Not valid. 10b - RANDOM_NUMBER. 11b - Not valid
0 ACTIVATE	Activate 0b - Disabled 1b - Activated. Update rhythm for VAL depends on COUNTER_CFG if DATA_SEL is set to 00b (LSB of COUNTER). Otherwise ONLINE_TEST_VAL is updated each time RANDOM_NUMBER is read

21.1.6 Online Test Results Register (ONLINE_TEST_VAL)

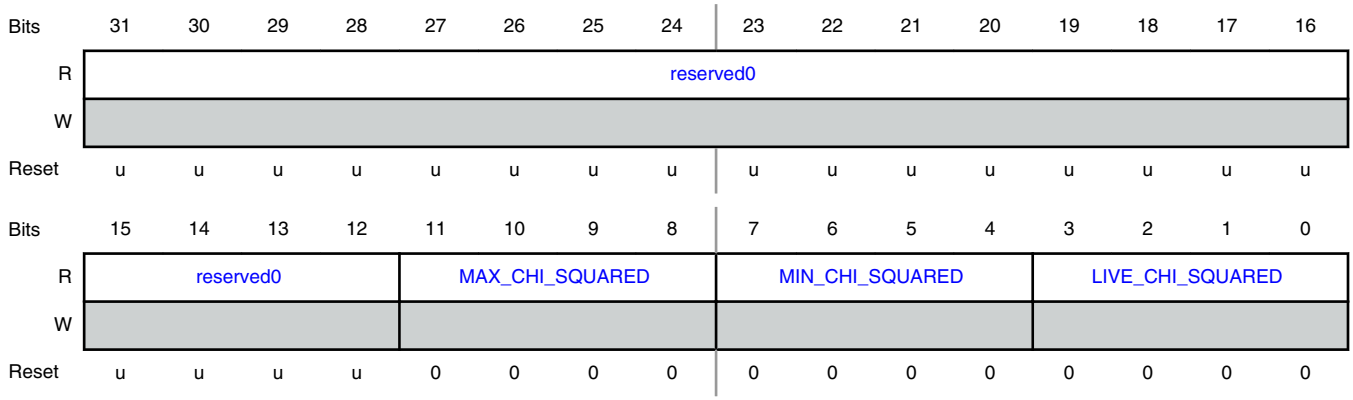
Offset

Register	Offset
ONLINE_TEST_VAL	14h

Function

This register is used to access the results of the randomness measurements.

Diagram



Fields

Field	Function
31-12 reserved0	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
11-8 MAX_CHI_SQUARED	LIVE_CHI_SQUARED Maximum Value Maximum value of LIVE_CHI_SQUARED since the last reset of this field. This field is reset when ONLINE_TEST_CFG[ACTIVATE]=0
7-4 MIN_CHI_SQUARED	LIVE_CHI_SQUARED Minimum Value Minimum value of LIVE_CHI_SQUARED since the last reset of this field. This field is reset when ONLINE_TEST_CFG[ACTIVATE]=0
3-0 LIVE_CHI_SQUARED	This value is updated as described in field 'activate' This value is updated as described in field ONLINE_TEST_CFG[ACTIVATE]. This value is a statistic value that indicates the quality of entropy generation. Low value means good, high value means no good. If ONLINE_TEST_CFG[DATA_SEL]<10, increase 'shift4x' till 'chi' is correct and poll 'refresh_cnt' before reading RANDOM_NUMBER.

21.1.7 Powerdown Mode and Reset Control Register (POWERDOWN)

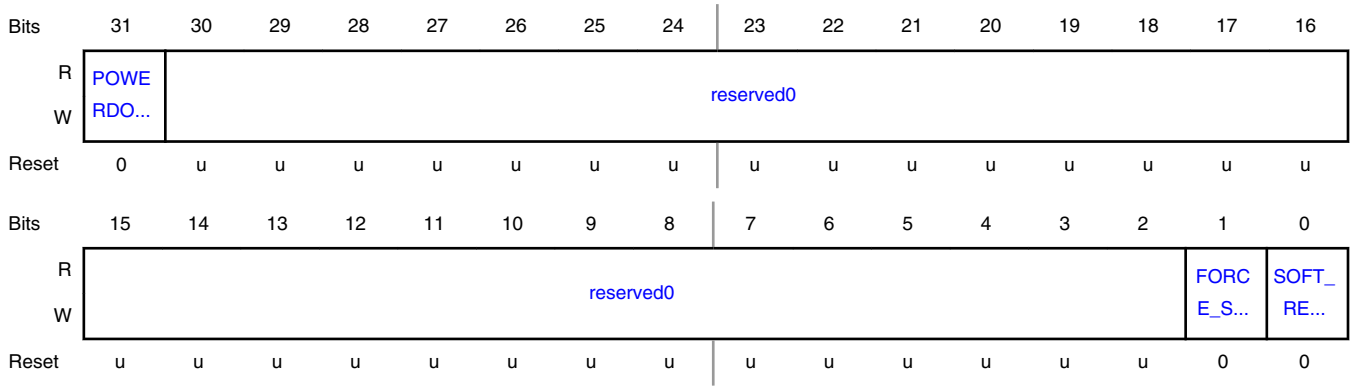
Offset

Register	Offset
POWERDOWN	FF4h

Function

This register is used to control reset and active state of the random number generator module. Generally, it is not necessary to use this register. The module is relatively small and so there is normally no need to reset or disable the module.

Diagram



Fields

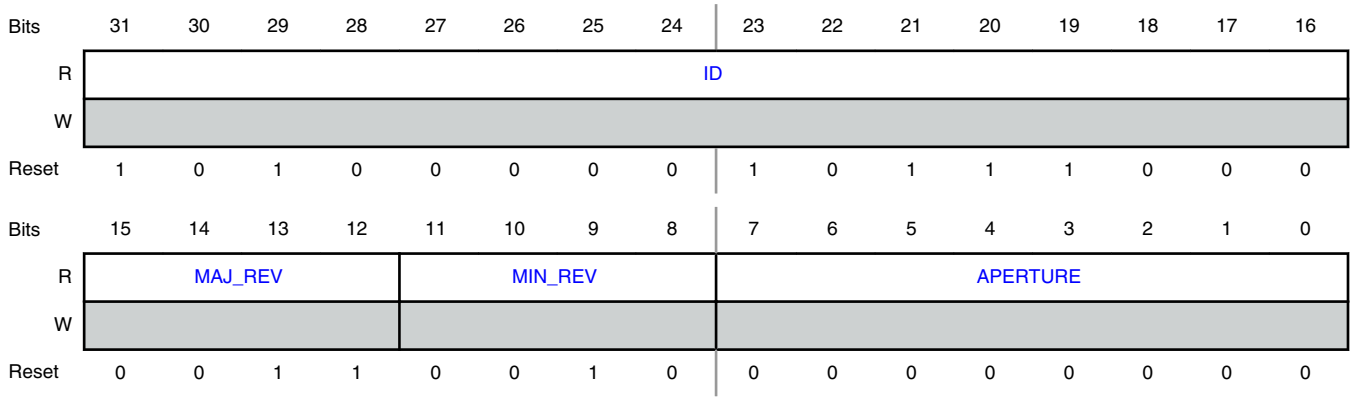
Field	Function
31 POWERDOWN	Power Down When set, all accesses to standard registers are blocked
30-2 reserved0	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
1 FORCE_SOFT_RESET	Force Software Reset When used with softreset it forces CORE_RESETN to low on acknowledge from core.
0 SOFT_RESET	Request Software Reset Request softreset that will go low automaticaly after acknowledge from core.

21.1.8 IP Identifier Register (MODULEID)

Offset

Register	Offset
MODULEID	FFCh

Diagram



Fields

Field	Function
31-16 ID	Identifier This is the unique identifier of the module.
15-12 MAJ_REV	Major Revision Major revision implies software modifications
11-8 MIN_REV	Minor Revision Minor revision without software consequences
7-0 APERTURE	Aperture Aperture number minus 1 of consecutive packets 4 KB reserved for this IP

Chapter 22
ISO 7816 Smart Card Interface (ISO7816)

22.1 ISO7816 register descriptions

22.1.1 ISO7816 memory map

ISO7816 base address: 4000_6000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Slot Select Register (SSR)	32	RW	See description
4h	Slot 1 Programmable Divider Register (LSB) (PDR1_LSB)	32	RW	See description
8h	Slot 1 Programmable Divider Register (MSB) (PDR1_MSB)	32	RW	See description
Ch	FIFO Control Register (FCR)	32	RW	See description
10h	Slot 1 Guard Time Register (GTR1)	32	RW	See description
14h	Slot 1 UART Configuration Register 1 (UCR11)	32	RW	See description
18h	Slot 1 UART Configuration Register 2 (UCR21)	32	RW	See description
1Ch	Slot 1 Clock Configuration Register (CCR1)	32	RW	See description
20h	Power Control Register (PCR)	32	RW	See description
24h	Early Answer Counter Register (ECR)	32	RW	See description
28h	Mute Card Counter RST Low Register (LSB) (MCRL_LSB)	32	RW	See description
2Ch	Mute Card Counter RST Low Register (MSB) (MCRL_MSB)	32	RW	See description
30h	Mute card Counter RST High Register (LSB) (MCRH_LSB)	32	RW	See description
34h	Mute Card Counter RST High Register (MSB) (MCRH_MSB)	32	RW	See description
3Ch	UART Receive Register / UART Transmit Register (URR_UTR)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
4Ch	Time-Out Register 1 (TOR1)	32	RW	See description
50h	Time-Out Register 2 (TOR2)	32	RW	See description
54h	Time-Out Register 3 (TOR3)	32	RW	See description
58h	Time-Out Configuration Register (TOC)	32	RW	See description
5Ch	FIFO Status Register (FSR)	32	RO	See description
60h	Mixed Status Register (MSR)	32	RO	See description
64h	UART Status Register 1 (USR1)	32	RO	See description
68h	UART Status Register 2 (USR2)	32	RO	See description

22.1.2 Slot Select Register (SSR)

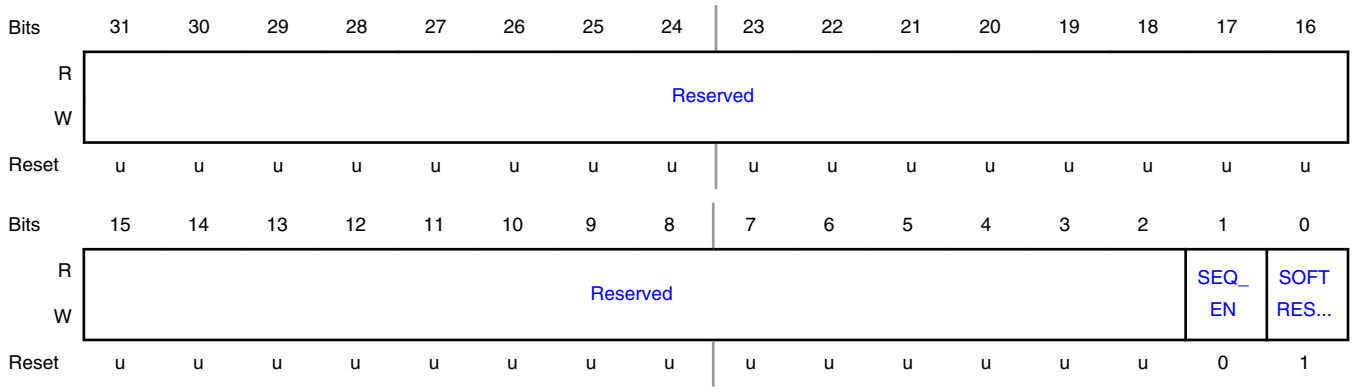
Offset

Register	Offset
SSR	0h

Function

This register controls overall reset and enable.

Diagram



Fields

Field	Function
31-2 —	RESERVED Reserved. User software must write zeroes to reserved bits. The value read from a reserved bit is not defined.
1 SEQ_EN	Sequencer Enable Set this bit to enable the sequencer. If this field is 0b, the sequencer will not respond to the Start control bit.
0 SOFTRESETN	Software Reset When set to logic 0, this bit resets the whole contact UART (software reset. If slot 1 is not activated, set to logic 1 automatically by hardware after one clock cycle; if slot 1 is activated, the hardware will deactivate the slot 1 and set to 1 after one clock clycle. Software should check whether the software reset is finished by reading SSR register before any further action.

22.1.3 Slot 1 Programmable Divider Register (LSB) (PDR1_LSB)

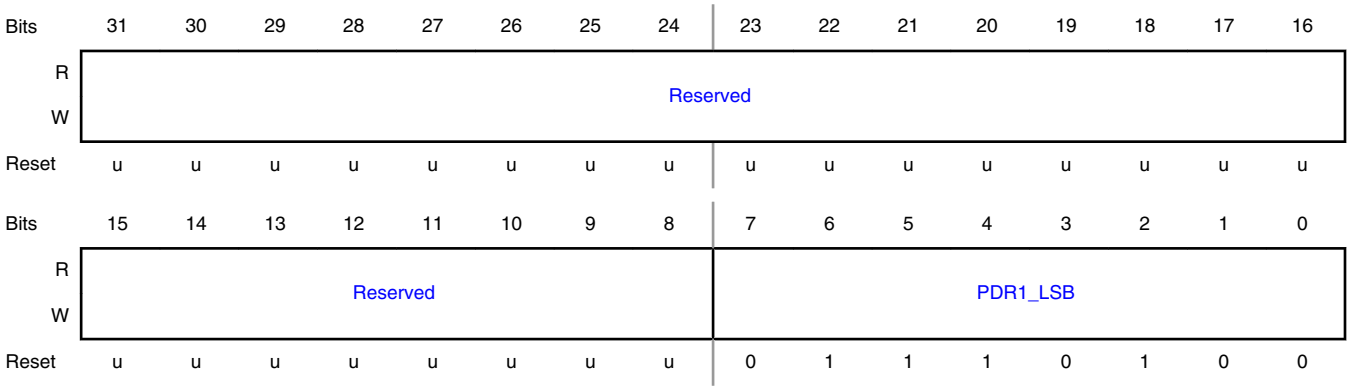
Offset

Register	Offset
PDR1_LSB	4h

Function

Least significant byte of a 16-bit counter defining the ETU. The ETU counter counts a number of cycles of the Contact Interface clock, this defines the ETU. The minimum acceptable value is 0001 0000b.

Diagram



Fields

Field	Function
31-8 —	RESERVED Reserved. User software must write zeroes to reserved bits. The value read from a reserved bit is not defined.
7-0 PDR1_LSB	PDR1_LSB

22.1.4 Slot 1 Programmable Divider Register (MSB) (PDR1_MSB)

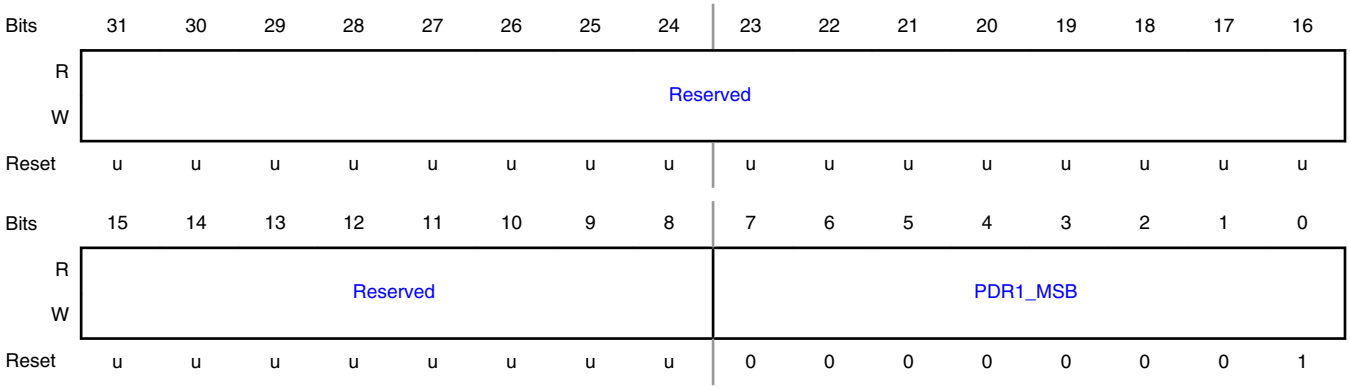
Offset

Register	Offset
PDR1_MSB	8h

Function

Most significant byte of a 16-bit counter defining the ETU. The ETU counter counts a number of cycles of the Contact Interface clock, this defines the ETU

Diagram



Fields

Field	Function
31-8 —	RESERVED Reserved. User software must write zeroes to reserved bits. The value read from a reserved bit is not defined.
7-0 PDR1_MSB	PDR1_MSB

22.1.5 FIFO Control Register (FCR)

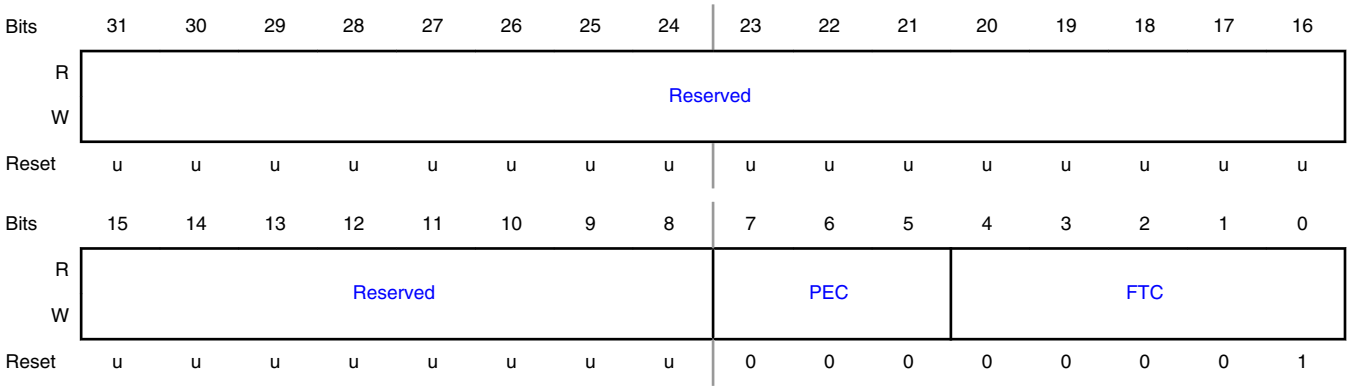
Offset

Register	Offset
FCR	Ch

Function

FIFO Control register defines the FIFO threshold (interrupt signalled by the USR1[FT] bit) and the number of repetition of character in case of Parity Error (interrupt signalled by the USR1[PE] bit).

Diagram



Fields

Field	Function
31-8 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
7-5 PEC	Parity Error Count • For protocol T = 0: Set the number of allowed repetitions in reception or transmission mode before setting USR1[PE]. — 000b - PE bit is set at logic 1 after one parity error has occurred — ... — 111b - PE bit is set at logic 1 after eight parity errors have occurred If a correct character is received before the programmed error number is reached, the error counter will be reset. If the programmed number of allowed parity errors is reached, USRE[PE] will be set at logic 1. If a transmitted character has been naked by the card, then the Contact UART will automatically retransmit it up to a number of times equal to the value programmed in bits PEC(2:0); the character will be resent at 15 ETU. If a transmitted character is considered as correct by the card after having been naked a number of times less than the value programmed in bits PEC(2:0) +1, the error counter will be reset. If a transmitted has been naked by the card a number of times equal to the value programmed in bits PEC(2:0) +1, the transmission stops and bit USR1[PE] is set at logic 1. The firmware is supposed to deactivate the card. If not, the firmware has the possibility to pursue the transmission. By reading the number of bytes present into the FIFO (ffl bits), it can determine which character has been naked PEC +1 times by the card. It will then flush the FIFO (FIFO flush bit). The next step consits in unlocking the transmission using dispe bit. By writing this bit at logic level one (and then at logic level zero if the firmware still wants to check parity errors), the transmission is unlocked. The firmware can now write bytes into the FIFO. In transmission mode, if bits PEC(2:0) are at logic 0, then the automatic retransmission is invalidated. There is no retransmission; the transmission continues with the next character sent at 13 ETU. • For protocol T = 1: The error counter has no action: bit PE is set at logic 1 at the first wrong received character.
4-0 FTC	FIFO Threshold Configuration Define the number of received or transmitted characters in the FIFO triggering the USR1[FT]. The FIFO depth is 32 bytes. In reception mode, it enables to know that a number equals to ftc(4:0) + 1 bytes have been received. In transmission mode, ftc(4:0) equals to the number of remaining bytes into the FIFO. **NOTE** In reception mode 00000 = length 1, and in transmission mode 00000 = length 0.

22.1.6 Slot 1 Guard Time Register (GTR1)

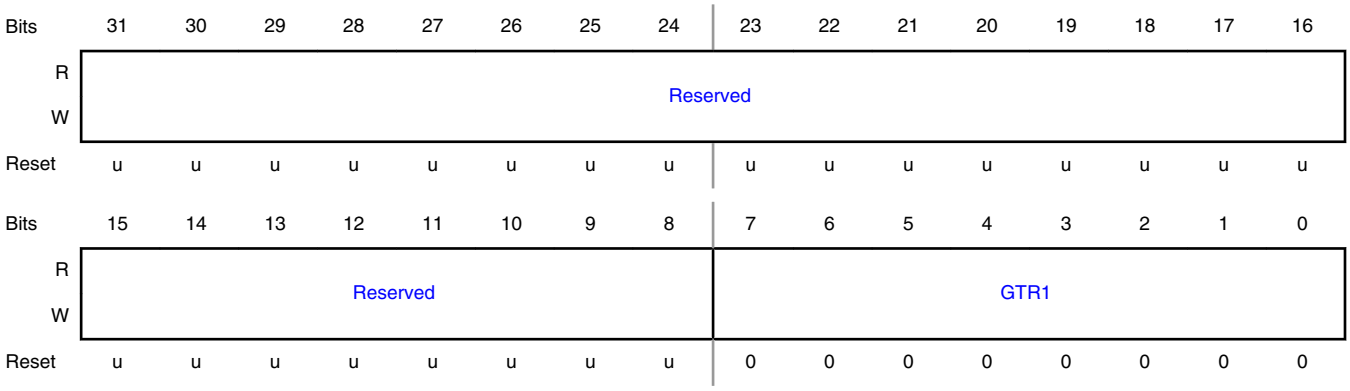
Offset

Register	Offset
GTR1	10h

Function

This configuration register is used to store the guard time given by the card during ATR.

Diagram



Fields

Field	Function
31-8 —	RESERVED Reserved. User software must write zeroes to reserved bits. The value read from a reserved bit is not defined.
7-0 GTR1	GTR1 Value used by the Contact UART notably in transmission mode. The Contact UART will wait this number of ETUs before transmitting the character. In protocol T=1, gtr = FFh means operation at 11 ETUs. In protocol T=0, gtr = FFh means operation at 12 ETUs. Otherwise, operation starts at (12 + gtr) ETUs, for protocol T=0 or T=1.

22.1.7 Slot 1 UART Configuration Register 1 (UCR11)

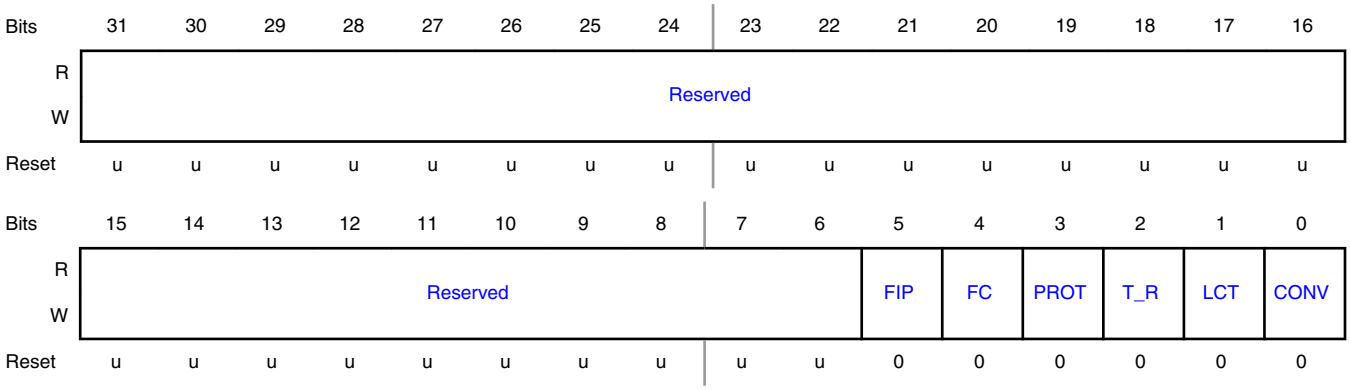
Offset

Register	Offset
UCR11	14h

Function

This configuration register defines the reception and transmission settings.

Diagram



Fields

Field	Function
31-6 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
5 FIP	Force Inverse Parity If bit FIP is set to logic 1, the Contact UART will NAK a correctly received character, and will transmit characters with wrong parity bits.
4 FC	Flow Control Enable Valid only if PROT=0.
3 PROT	Select Protocol 0b - T=0 1b - T=1
2 T_R	Transmit/Receive Defines the mode. Bit T_R is set by software for transmission mode. Bit T_R is automatically reset to logic 0 by hardware, if bit LCT has been used before transmitting the last character. **NOTE** The FIFO is flushed during the switching between reception mode and transmission mode, any remaining bytes are lost. 0b - Reception 1b - Transmission

Table continues on the next page...

Table continued from the previous page...

Field	Function
1 LCT	Last Character to Transmit Bit LCT is set to logic 1 by software before writing the last character to be transmitted in register URR_UTR. It allows automatic change to reception mode. It is reset to logic 0 by hardware at the end of a successful transmission after 11.75 ETUs in protocol T = 0 and after 10.75 ETUs in protocol T = 1. When bit LCT is being reset to logic 0, bit T_R is also reset to logic 0 and the UART is ready to receive a character. LCT bit can be set to logic 1 by software not only when writing the last character to be transmitted but also during the transmission or even at the beginning of the transmission. It will be taken into account when the FIFO becomes empty, which implies for the software to be able to regularly re-load the FIFO when transmitting more than 32 bytes to ensure there is at least one byte into the FIFO as long as the transmission is not finished. Else, a switch to reception mode will prematurely occur before having transmitted all the bytes.
0 CONV	Convention Bit CONV is set to logic 1 if the convention is direct. Bit CONV is either automatically written by hardware according to the convention detected during ATR, or by software if the bit UCR21[AUTOCONV] is set to logic 1.

22.1.8 Slot 1 UART Configuration Register 2 (UCR21)

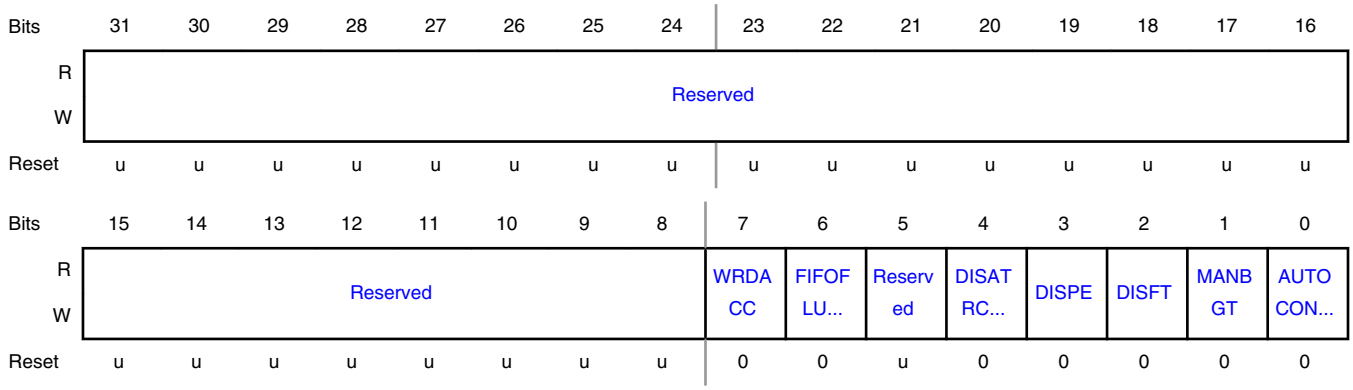
Offset

Register	Offset
UCR21	18h

Function

This configuration register defines the reception and transmission settings.

Diagram



Fields

Field	Function
31-8 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
7 WRDACC	FIFO Word Access 0b - FIFO supports byte access (read and write). 1b - FIFO supports word (4 bytes) access (read and write), access failure is indicated by bit USR2[WRDACCERR].
6 FIFOFLUSH	FIFO Flush When set to logic 1, the FIFO is flushed whatever the mode (reception or transmission) is. It can be used before any reception or transmission of characters but not while receiving or transmitting a character. It is reset to logic 0 by hardware after one clk_ip cycle.
5 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
4 DISATRCOUNTER	Disable ATR Counter When set to logic 1, the bits EARLY and MUTE in USR1 register will not generate interrupt. This bit should be set before activating.
3 DISPE	Disable Parity Error Interrupt When set to logic 1, the parity is not checked in both reception and transmission modes, the USR1[PE] bit will not generate interrupt.
2 DISFT	Disable Fifo Threshold Interrupt When set to logic 1, the USR1[FT] bit will not generate interrupt.
1 MANBGT	Manual BGT When set to logic 1, BGT is managed by software, else by hardware.
0 AUTOCONVN	Automatically Detected Convention If bit AUTOCONV = 1, the convention is set by software using UCR11[CONV] bit. If the bit is reset to logic 0, the configuration is automatically detected on the first received character and the bit automatically set after convention detection.

22.1.9 Slot 1 Clock Configuration Register (CCR1)

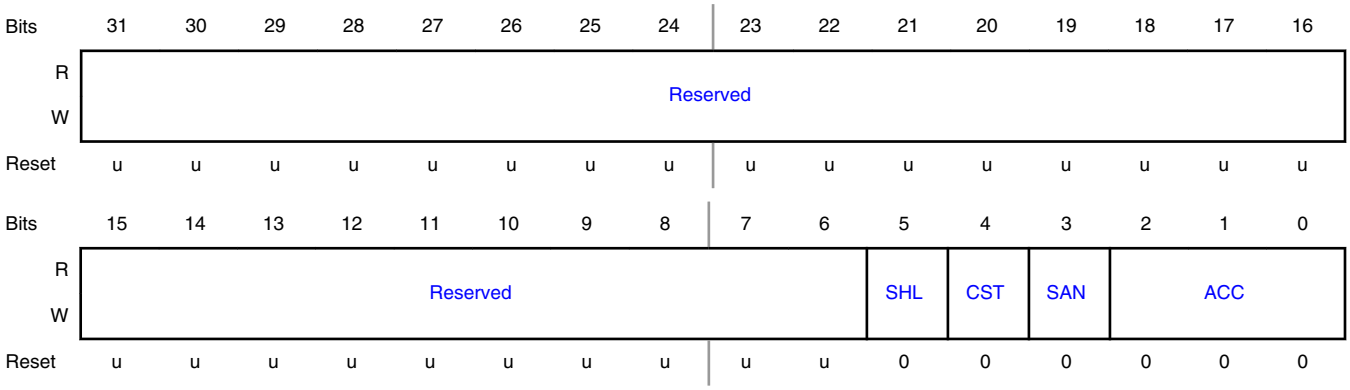
Offset

Register	Offset
CCR1	1Ch

Function

This configuration register defines the card clock frequency.

Diagram



Fields

Field	Function
31-6 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
5 SHL	Stop HIGH or LOW If operating in asynchronous mode (SAN=0) and the clock is stopped (CST=1) then the clock will stop low if SHL=0, and it will stop high if SHL=1. If operating in synchronous mode (SAN=1) then the clock will stop low if SHL=0 and it will stop high if SHL=1.
4 CST	Clock Stop In the case of an asynchronous card, bit CST defines whether the clock to the card is stopped or not; if bit CST is reset to logic 0, then the clock is determined by bits ACC0, ACC1 and ACC2.
3 SAN	Synchronous/Asynchronous Card When set to logic 1, the Contact UART supports synchronous card. The Contact UART is then bypassed, only bit 0 of registers URR_UTR is connected to pin I/O. In this case, the card clock is controlled by bit SHL, and RST card is controlled by PCR[RSTIN] bit. When set to logic 0, the Contact UART supports asynchronous card. Dynamic change (while activated) is not supported. The choice should be done before activating the card.

Table continues on the next page...

Table continued from the previous page...

Field	Function
2-0 ACC	Asynchronous Card Clock All frequency changes are synchronous, thus ensuring that no spikes or unwanted pulse widths occur during changes. This field configures the card clock frequency used by the Contact UART. 000b - Card clock frequency = fclk_ip. 001b - Card clock frequency = fclk_ip /2. 010b - Card clock frequency = fclk_ip /3. 011b - Card clock frequency = fclk_ip /4. 100b - Clock frequency = fclk_ip /5. 101b - Card clock frequency = fclk_ip /6. 110b - Card clock frequency = fclk_ip /8. 111b - Card clock frequency = fclk_ip /16.

22.1.10 Power Control Register (PCR)

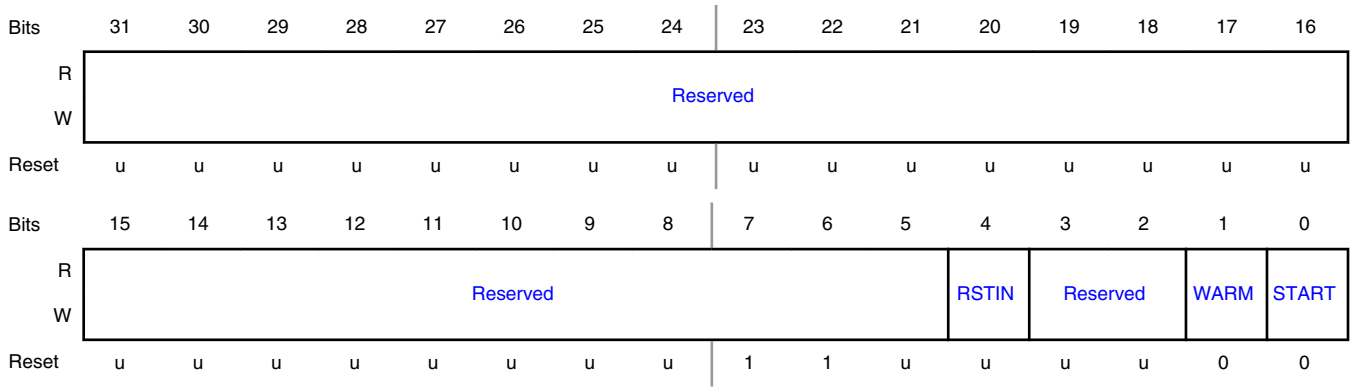
Offset

Register	Offset
PCR	20h

Function

This configuration register enables to start or stop card sessions

Diagram



Fields

Field	Function
31-5 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
4 RSTIN	Reset Signal Control Used in synchronous mode to control the RST signal.
3-2 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
1 WARM	Warm Enable a warm reset in the ATR counter by setting this bit.
0 START	Start Set to start a card session.

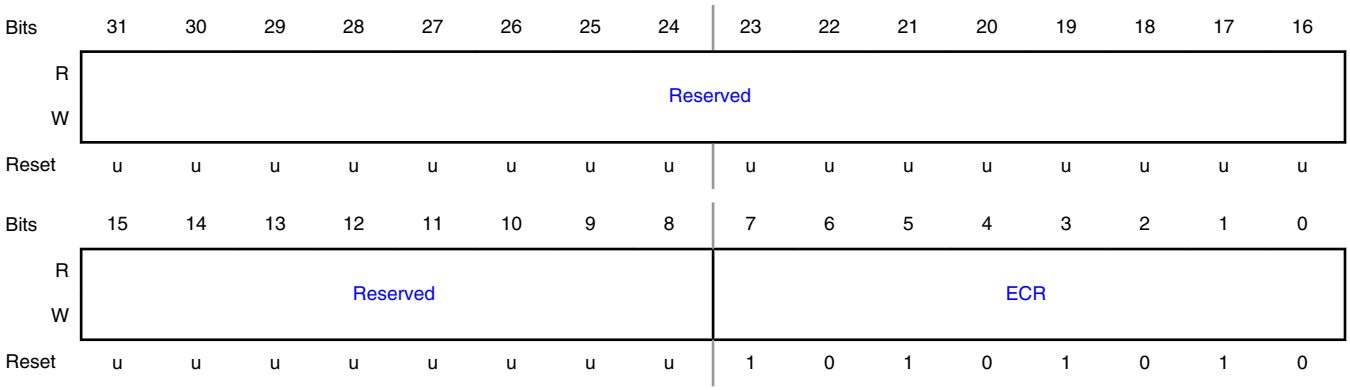
22.1.11 Early Answer Counter Register (ECR)

Offset

Register	Offset
ECR	24h

Function
This configuration register enables to program the value of a 8-bit counter used to check whether the card has answered too early.

Diagram



Fields

Field	Function
31-8 —	RESERVED Reserved. User software must write zeroes to reserved bits. The value read from a reserved bit is not defined.
7-0 ECR	Early Answer Counter The card should not respond too quickly. The early answer counter register sets the threshold for deciding when a response is too early.

22.1.12 Mute Card Counter RST Low Register (LSB) (MCRL_LSB)

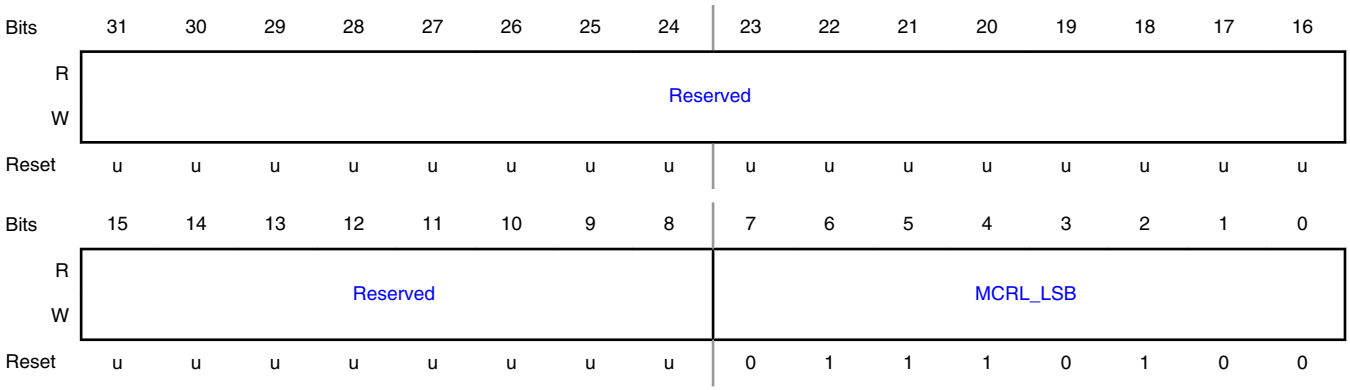
Offset

Register	Offset
MCRL_LSB	28h

Function

This configuration register is the least significant byte of a 16-bit counter used to check whether the card is mute.

Diagram



Fields

Field	Function
31-8 —	RESERVED Reserved. User software must write zeroes to reserved bits. The value read from a reserved bit is not defined.
7-0 MCRL_LSB	Least Significant Byte of Mute Card Counter Value A response should be received from the card before the mute card counter expires.

22.1.13 Mute Card Counter RST Low Register (MSB) (MCRL_MSB)

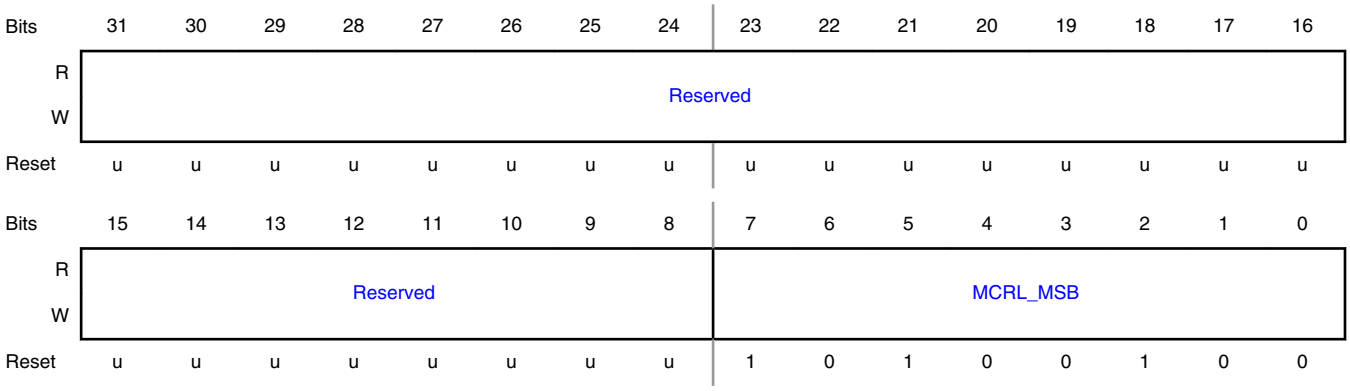
Offset

Register	Offset
MCRL_MSB	2Ch

Function

This configuration register is the most significant byte of a 16-bit counter used to check whether the card is mute.

Diagram



Fields

Field	Function
31-8 —	RESERVED Reserved. User software must write zeroes to reserved bits. The value read from a reserved bit is not defined.
7-0 MCRL_MSB	Most Significant Byte of Mute Card Counter value A response should be received from the card before the mute card counter expires.

22.1.14 Mute card Counter RST High Register (LSB) (MCRH_LSB)

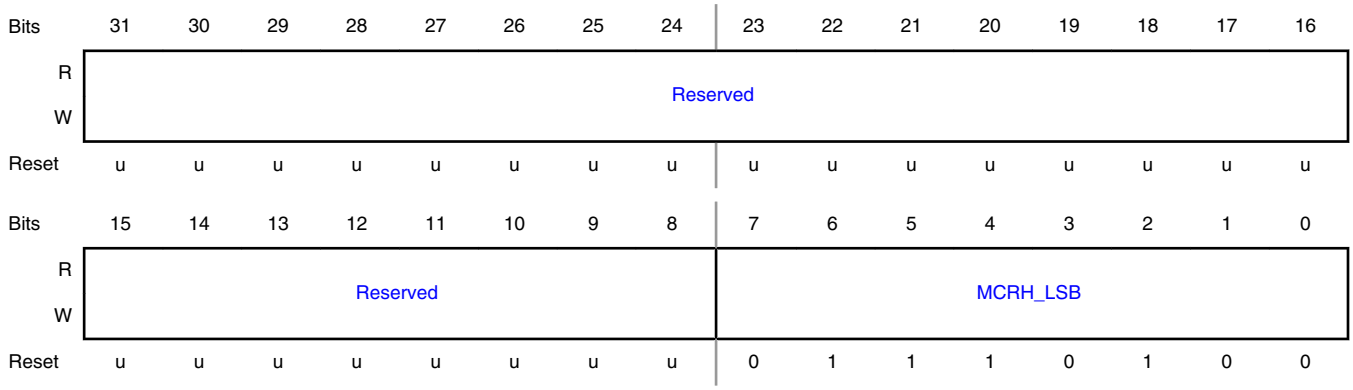
Offset

Register	Offset
MCRH_LSB	30h

Function

The MCRH, mute card reset high, value is a 16-bit value comprised of the {MCRH_MSB, MCRH_LSB} register values and is the mute card timeout setting for use when reset is high. A card should respond before this timeout, otherwise it will be deemed mute.

Diagram



Fields

Field	Function
31-8 —	RESERVED Reserved. User software must write zeroes to reserved bits. The value read from a reserved bit is not defined.
7-0 MCRH_LSB	MCRH_LSB

22.1.15 Mute Card Counter RST High Register (MSB) (MCRH_MSB)

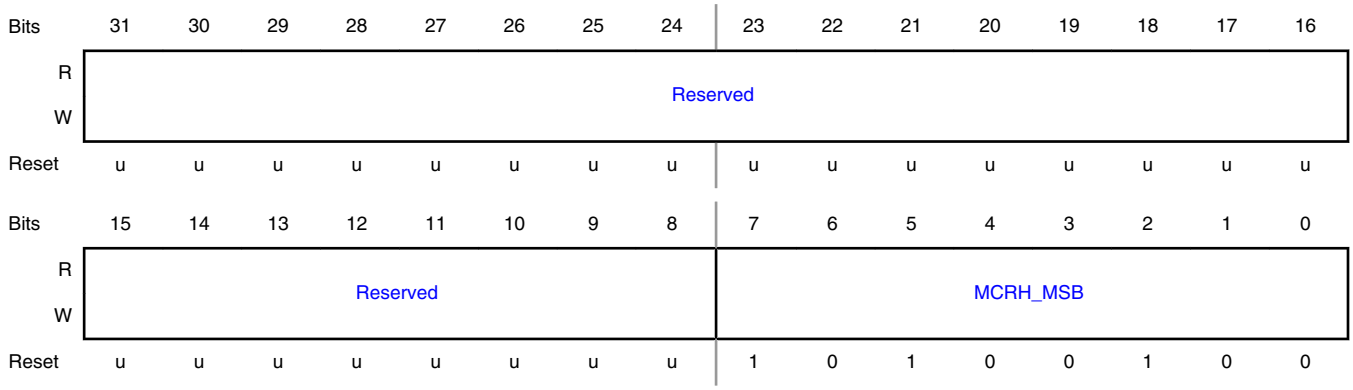
Offset

Register	Offset
MCRH_MSB	34h

Function

The MCRH, mute card reset high, value is a 16-bit value comprised of the {MCRH_MSB, MCRH_LSB} register values and is the mute card timeout setting for use when reset is high. A card should respond before this timeout, otherwise it will be deemed mute.

Diagram



Fields

Field	Function
31-8 —	RESERVED Reserved. User software must write zeroes to reserved bits. The value read from a reserved bit is not defined.
7-0 MCRH_MSB	MCRH_MSB

22.1.16 UART Receive Register / UART Transmit Register (URR_UTR)

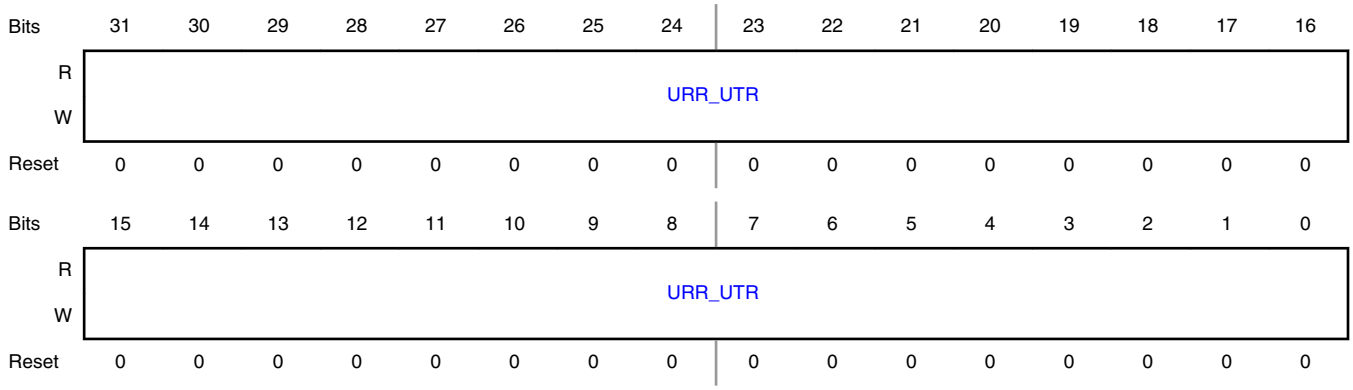
Offset

Register	Offset
URR_UTR	3Ch

Function

Values written here will be transmiited. Values received over the UART can be read here.

Diagram



Fields

Field	Function
31-0 URR_UTR	URR_UTR

22.1.17 Time-Out Register 1 (TOR1)

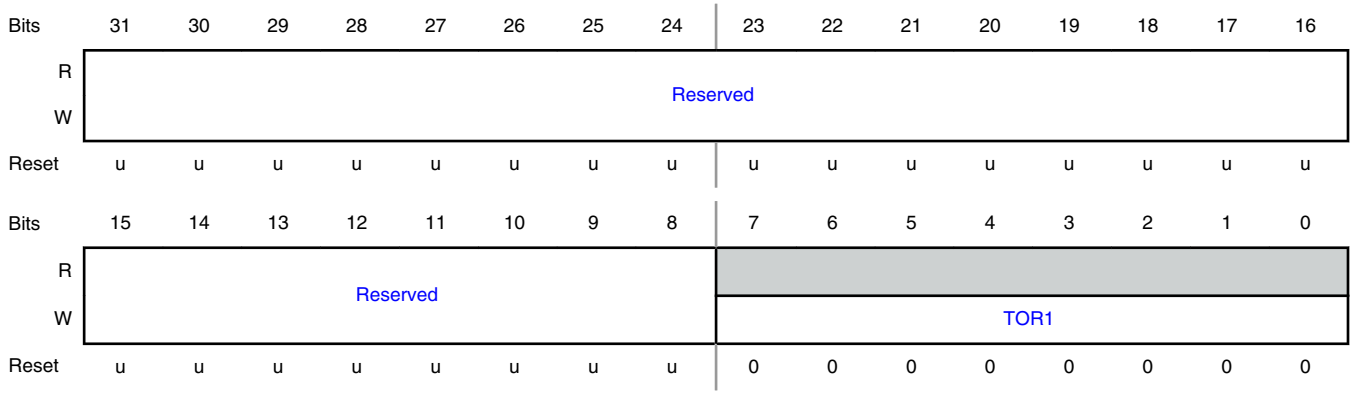
Offset

Register	Offset
TOR1	4Ch

Function

This configuration register enables to program the value of a 8-bit ETU counter used to check some timings (CWT, BWT, etc). Exact function depends on settings in TOC.

Diagram



Fields

Field	Function
31-8 —	RESERVED Reserved. User software must write zeroes to reserved bits. The value read from a reserved bit is not defined.
7-0 TOR1	Timeout Register 1 setting This value is used within the timers and timeout functionality of the module.

22.1.18 Time-Out Register 2 (TOR2)

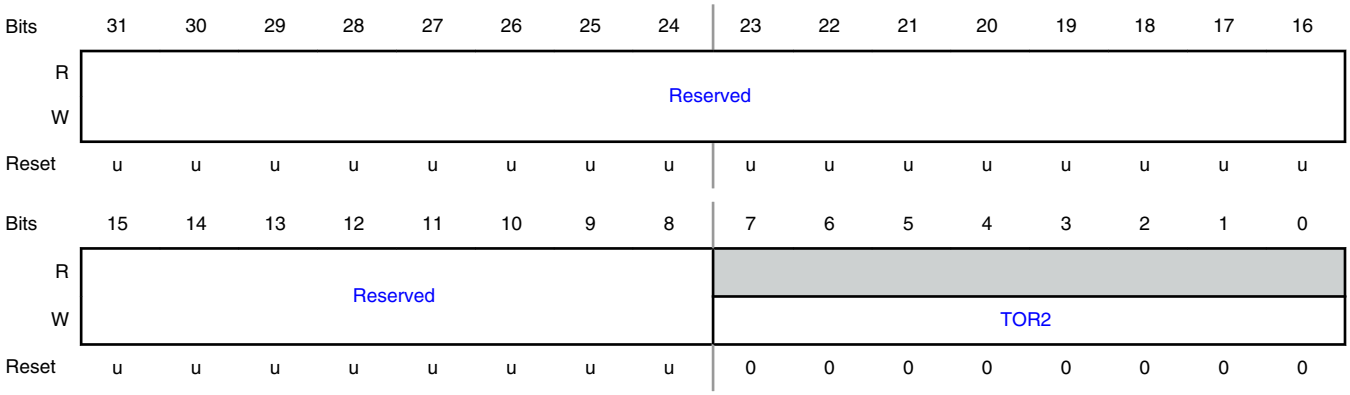
Offset

Register	Offset
TOR2	50h

Function

This configuration register enables to program the value of a 8-bit ETU counter used to check some timings (CWT, BWT, etc). Exact function depends on settings in TOC.

Diagram



Fields

Field	Function
31-8 —	RESERVED Reserved. User software must write zeroes to reserved bits. The value read from a reserved bit is not defined.
7-0 TOR2	Timeout Register 2 setting This value is used within the timers and timeout functionality of the module.

22.1.19 Time-Out Register 3 (TOR3)

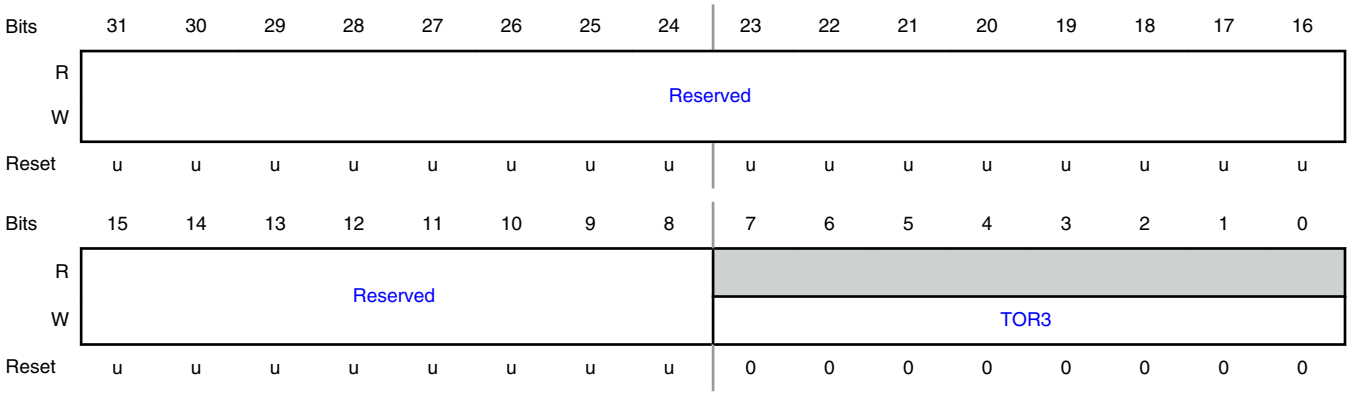
Offset

Register	Offset
TOR3	54h

Function

This configuration register enables to program the value of a 8-bit ETU counter used to check some timings (CWT, BWT,etc). Exact function depends on settings in TOC.

Diagram



Fields

Field	Function
31-8 —	RESERVED Reserved. User software must write zeroes to reserved bits. The value read from a reserved bit is not defined.
7-0 TOR3	Timeout Register 3 setting This value is used within the timers and timeout functionality of the module.

22.1.20 Time-Out Configuration Register (TOC)

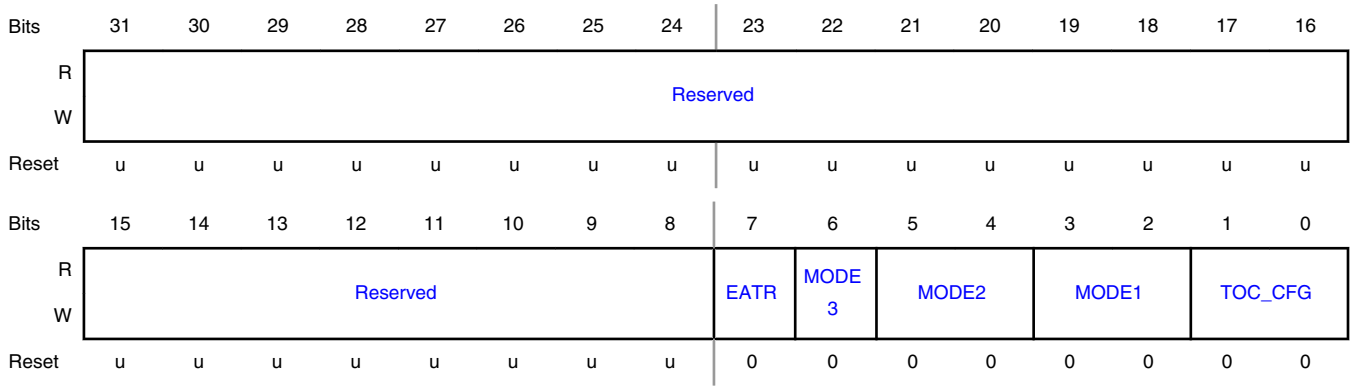
Offset

Register	Offset
TOC	58h

Function

This configuration register is used for setting different configurations of the time-out counter.

Diagram



Fields

Field	Function
31-8 —	RESERVED Reserved. User software must write zeroes to reserved bits. The value read from a reserved bit is not defined.
7 EATR	Auto Stop Mode Enable Enables an auto stop mode.
6 MODE3	Timeout Counter Mode 3 Configuration Configures the mode of time out counter 3.
5-4 MODE2	Timeout Counter Mode 2 Configuration Configures the mode of time out counter 2.
3-2 MODE1	Timeout Counter Mode 1 Configuration Configures the mode of time out counter 1.
1-0 TOC_CFG	Timeout Counter Configuration Configures the operating mode of the time out counters.

22.1.21 FIFO Status Register (FSR)

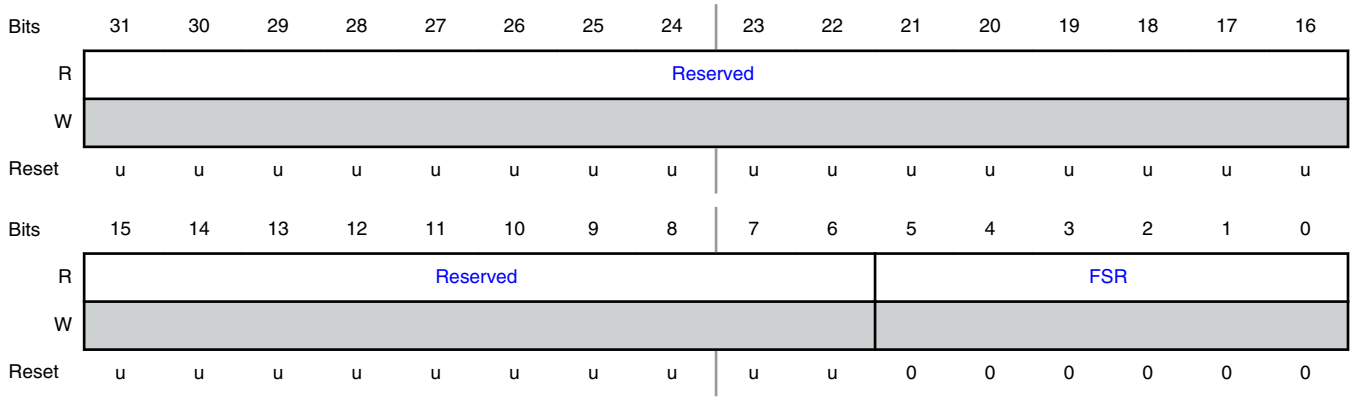
Offset

Register	Offset
FSR	5Ch

Function

This register shows the FIFO fill level.

Diagram



Fields

Field	Function
31-6 —	RESERVED Reserved. User software must write zeroes to reserved bits. The value read from a reserved bit is not defined.
5-0 FSR	FIFO Fill Level Indicate the number of bytes in the transmit or receive FIFO. If UCR11[T_R] field is set, the value reported is the fill level of the transmit FIFO, otherwise it is the fill level of the receive FIFO.

22.1.22 Mixed Status Register (MSR)

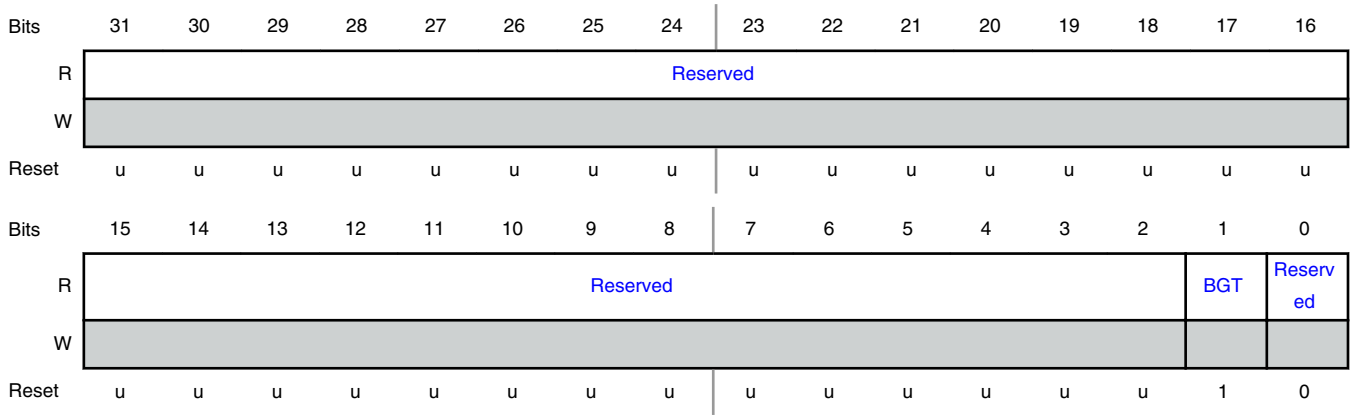
Offset

Register	Offset
MSR	60h

Function

This status register shows the block guard time status. It is intended for polling; it doesn't generate any interrupt.

Diagram



Fields

Field	Function
31-2 —	RESERVED Reserved. User software must write zeroes to reserved bits. The value read from a reserved bit is not defined.
1 BGT	Block Guard Time Status In asynchronous mode, this bit reads as 0 when the the controller is within the Block Guard Time. After the end of this time period, this status bit reads as 1. In synchronous mode, this always reads as 1.
0 —	RESERVED Reserved. User software must write zeroes to reserved bits. The value read from a reserved bit is not defined.

22.1.23 UART Status Register 1 (USR1)

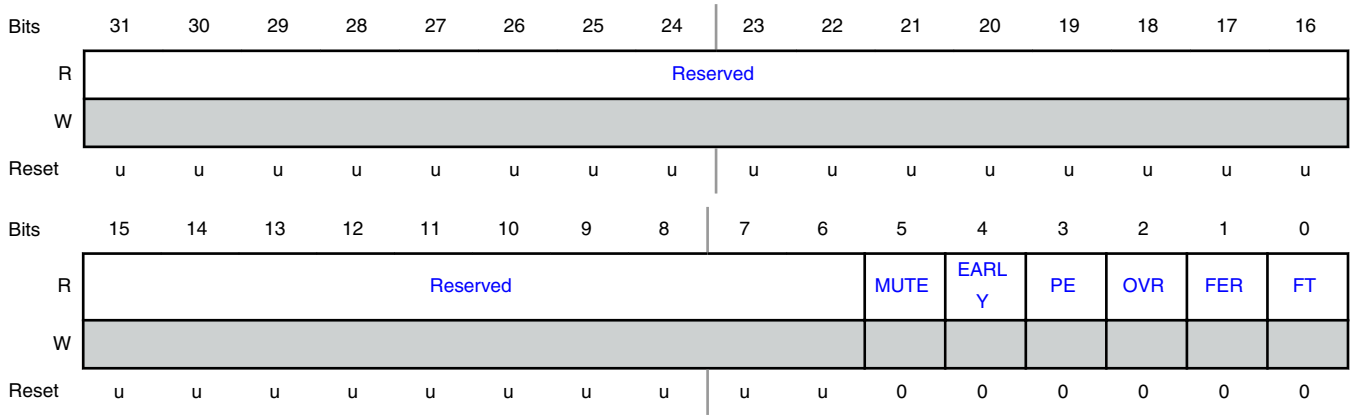
Offset

Register	Offset
USR1	64h

Function

The USR1 and USR2 provide the interrupt register with status information: these bits coming from the Contact UART core are used to manage the reception and transmission of characters. Read this register enables to know what is the cause of the interrupt. The bits are set to logic 1 by hardware and set to logic 0 by reading (with a hardware mechanism avoiding the loss of incoming interrupt while reading).

Diagram



Fields

Field	Function
31-6 —	RESERVED Reserved. User software must write zeroes to reserved bits. The value read from a reserved bit is not defined.
5 MUTE	Mute Status Set when a card does not respond before the mute timeout period. Status bit is cleared on a read.
4 EARLY	Card Answer Early Status Set when a card answers too early. Status bit is cleared on a read.
3 PE	Parity Error Status Set when a parity error is detected in reception or transmission. Status bit is cleared on a read.
2 OVR	Overrun Status Set when the FIFO is full and a new character is received. Status bit is cleared on a read.
1 FER	Framing Error Status I/O was low at 10,25 ETU. Status bit is cleared on a read.
0 FT	FIFO Threshold Passed Status FIFO threshold has been passed status bit. In reception, the FIFO contains FTC+1 or more bytes. In transmission, the FIFO contains FTC or less bytes. Status bit is cleared on a read.

22.1.24 UART Status Register 2 (USR2)

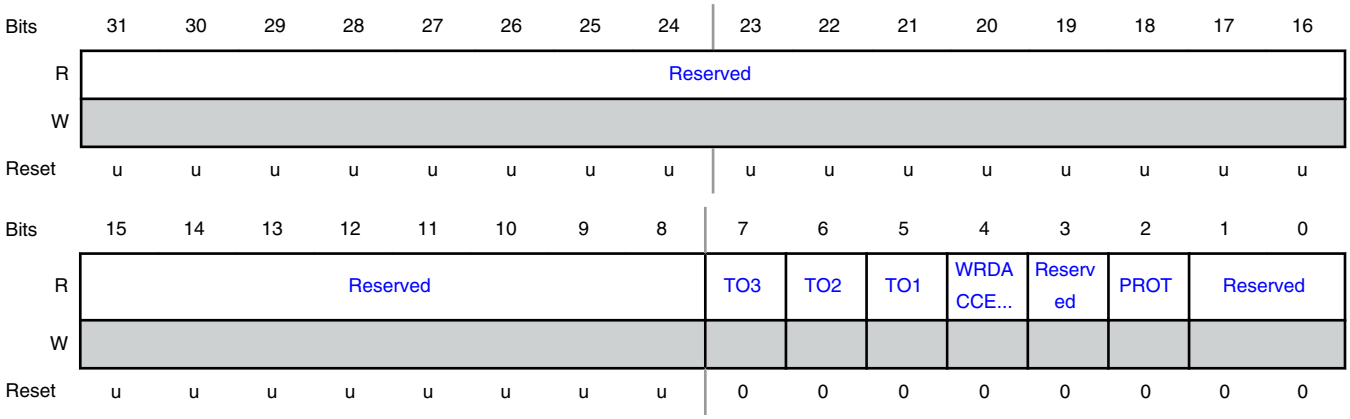
Offset

Register	Offset
USR2	68h

Function

Together USR1 and USR2 provide the interrupt register: these bits coming from the Contact UART core are used to manage the reception and transmission of characters. Read this register enables to know what is the cause of the interrupt. The bits are set to logic 1 by hardware and set to logic 0 by reading (with a hardware mechanism avoiding the loss of incoming interrupt while reading).

Diagram



Fields

Field	Function
31-8 —	RESERVED Reserved. User software must write zeroes to reserved bits. The value read from a reserved bit is not defined.
7 TO3	Timer 3 Status Timer3 has finished its count.
6 TO2	Timer 2 Status Timer2 has finished its count.
5 TO1	Timer 1 Status Timer1 has finished its count.
4 WRDACCERR	FIFO Word Access Error In transmission, an attempt was made to write a word to the FIFO when there was not enough space for the word. In reception an attempt was made to read a word from the FIFO when there was not a full word to read.
3 —	RESERVED Reserved. User software must write zeroes to reserved bits. The value read from a reserved bit is not defined.
2 PROT	Sequencer Fault Occured.

Table continues on the next page...

Table continued from the previous page...

Field	Function
1-0 —	RESERVED Reserved. User software must write zeroes to reserved bits. The value read from a reserved bit is not defined.

Chapter 23
Async System Configuration (ASYNC_SYSCON)

23.1 ASYNC_SYSCON register descriptions

23.1.1 ASYNC_SYSCON memory map

ASYNC_SYSCON base address: 4002_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Asynchronous Peripherals Reset Control Register (ASYNCPRESETCTRL)	32	RW	See description
4h	ASYNCPRESETCTRL Bits Set Register (ASYNCPRESETCTRLSET)	32	WO	See description
8h	ASYNCPRESETCTRL Bits Clear Register (ASYNCPRESETCTRLCLR)	32	WO	See description
10h	Asynchronous Peripherals Clock Control Register (ASYNCAPBCLKCTRL)	32	RW	See description
14h	ASYNCAPBCLKCTRL Bits Set Register (ASYNCAPBCLKCTRLSET)	32	WO	See description
18h	ASYNCAPBCLKCTRL Bits Clear Register (ASYNCAPBCLKCTRLCLR)	32	WO	See description
20h	Asynchronous APB Clock Source Select Register (ASYNCAPBCLKSELA)	32	RW	See description
A0h	Temperature Sensor Control Register (TEMPSENSORCTRL)	32	RW	See description
A4h	NFC Tag Pads Control Register (NFCTAGPADSCTRL)	32	RW	See description
A8h	XTAL 32 MHz LDO Control Register (XTAL32MLDOCTRL)	32	RW	See description
ACh	32 MHz XTAL Control Register (XTAL32MCTRL)	32	RW	See description
B0h	Analog Interfaces (PMU and Radio) Identity Registers (ANALOGID)	32	RO	See description
B4h	Radio Analog Modules Status Register (RADIOSTATUS)	32	RO	See description
BCh	DC Bus Control (DCBUSCTRL)	32	RW	See description
C0h	Frequency Measure Register (FREQMECTRL)	32	RW	0000_0000h

Table continues on the next page...

Table continued from the previous page...

Offset	Register	Width (In bits)	Access	Reset value
C8h	NFCTAG VDD Output Control Register (NFCTAG_VDD)	32	RW	See description
CCh	Full IC Reset Request (from Software application) Register (SWRESETCTRL)	32	WO	See description

23.1.2 Asynchronous Peripherals Reset Control Register (ASYNCPRESETCTRL)

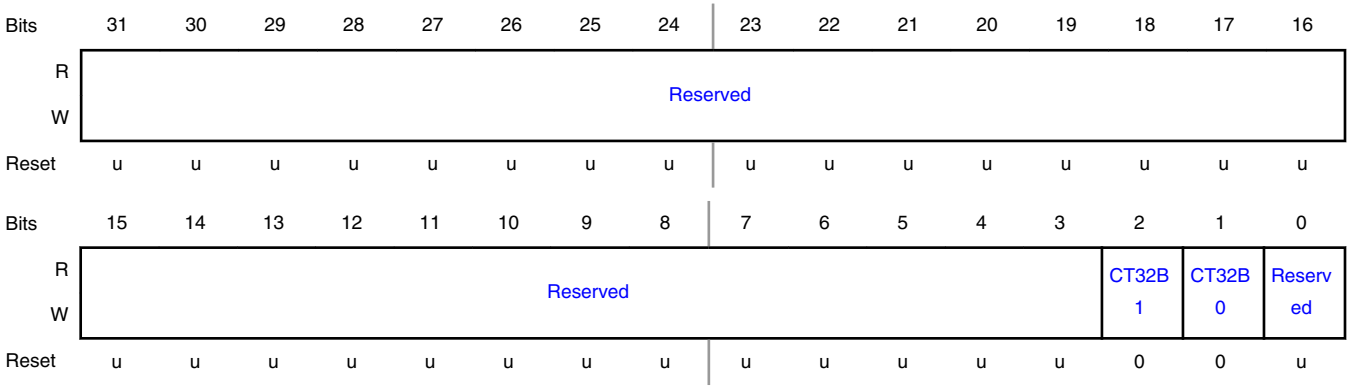
Offset

Register	Offset
ASYNCPRESETCTRL	0h

Function

The ASYNCPRESETCTRL register allows software to reset specific peripherals attached to the async APB bridge. Writing a 0 to any assigned bit in this register clears the reset and allows the specified peripheral to operate. Writing a 1 asserts the reset.

Diagram



Fields

Field	Function
31-3 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
2 CT32B1	CT32B1 Reset Control 0b - Clear reset to Counter/Timer CTIMER1/CT32B1. 1b - Assert reset to Counter/Timer CTIMER1/CT32B1.
1 CT32B0	CT32B0 Reset Control 0b - Clear reset to Counter/Timer CTIMER0/CT32B0. 1b - Assert reset to Counter/Timer CTIMER0/CT32B0.
0 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

23.1.3 ASYNCPRESETCTRL Bits Set Register (ASYNCPRESETCTRLSET)

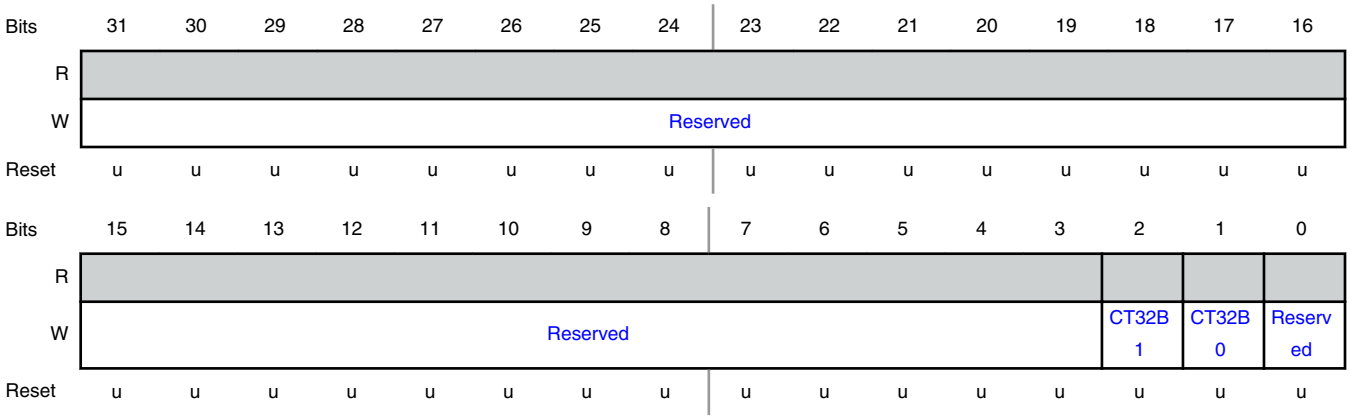
Offset

Register	Offset
ASYNCPRESETCTRLSET	4h

Function

Set bits in ASYNCPRESETCTRL. Writing ones to this register sets the corresponding bit or bits in the ASYNCPRESETCTRL register, if they are implemented

Diagram



Fields

Field	Function
31-3 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
2 CT32B1	ASYNCPRESETCTRL[CT32B1] Set Writing 1 to this bit sets the ASYNCPRESETCTRL[CT32B1] 0b - No effect. 1b - Set the ASYNCPRESETCTRL[CT32B1].
1 CT32B0	ASYNCPRESETCTRL[CT32B0] Set Writing 1 to this field sets the ASYNCPRESETCTRL[CT32B0] 0b - No effect. 1b - Set the ASYNCPRESETCTRL[CT32B0].
0 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

23.1.4 ASYNCPRESETCTRL Bits Clear Register (ASYNCPRESETCTRLCLR)

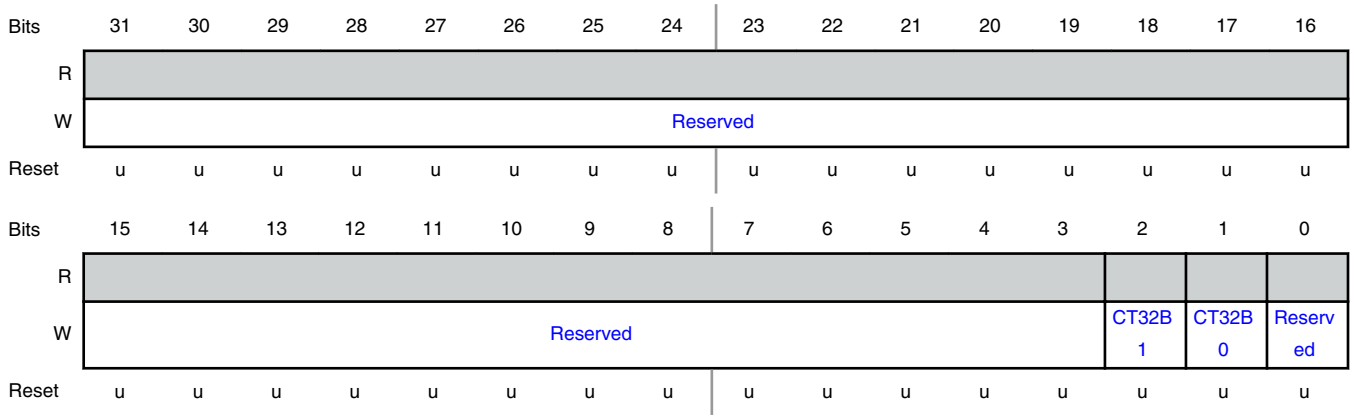
Offset

Register	Offset
ASYNCPRESETCTRLCLR	8h

Function

Clear bits in ASYNCPRESETCTRL. Writing ones to this register clears the corresponding bit or bits in the ASYNCPRESETCTRL register, if they are implemented

Diagram



Fields

Field	Function
31-3 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
2 CT32B1	ASYNCPRESETCTRL[CT32B1] Clear Writing 1 to this field clears the ASYNCPRESETCTRL[CT32B1] 0b - No effect. 1b - Clear the ASYNCPRESETCTRL[CT32B1].
1 CT32B0	ASYNCPRESETCTRL[CT32B0] Clear Writing 1 to this field clears the ASYNCPRESETCTRL[CT32B0]. 0b - No effect. 1b - Clear the ASYNCPRESETCTRL[CT32B0].
0 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

23.1.5 Asynchronous Peripherals Clock Control Register (ASYNCAPBCLKCTRL)

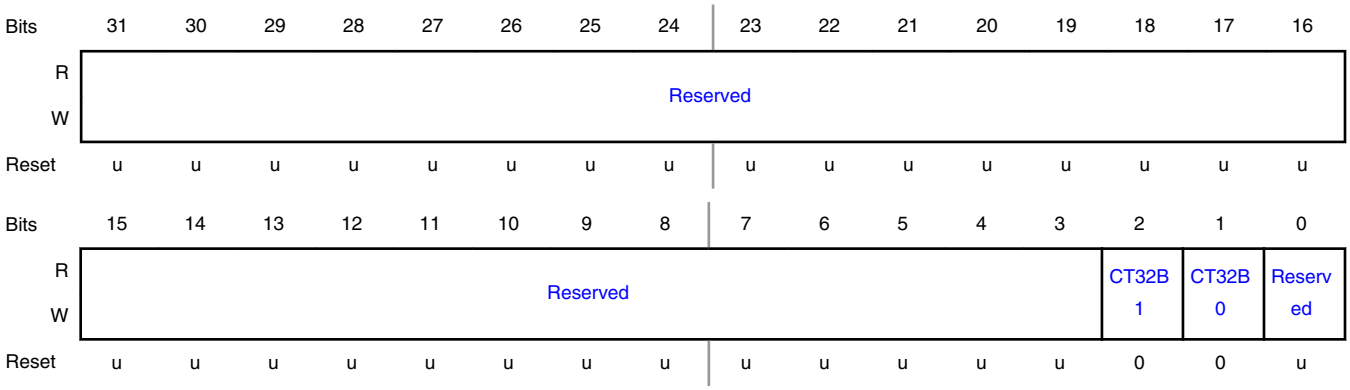
Offset

Register	Offset
ASYNCAPBCLKCTRL	10h

Function

This register controls how the clock selected for the asynchronous APB peripherals is divided to provide the clock to the asynchronous peripherals.

Diagram



Fields

Field	Function
31-3 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
2 CT32B1	CT32B1 Clock Control 0b - Disable clock to Counter/Timer CTIMER1/CT32B1. 1b - Enable clock to Counter/Timer CTIMER1/CT32B1.
1 CT32B0	CT32B0 Clock Control 0b - Disable clock to Counter/Timer CTIMER0/CT32B0. 1b - Enable clock to Counter/Timer CTIMER0/CT32B0.
0 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

23.1.6 ASYNCAPBCLKCTRL Bits Set Register (ASYNCAPBCLKCTRLSET)

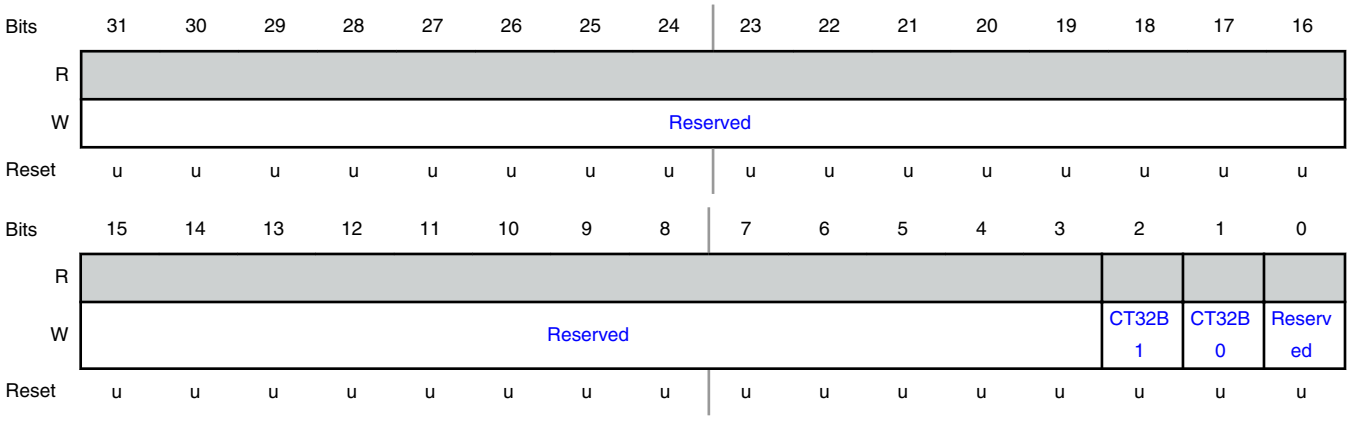
Offset

Register	Offset
ASYNCAPBCLKCTRLSET	14h

Function

Set bits in ASYNCAPBCLKCTRL. Writing ones to this register sets the corresponding bit or bits in the ASYNCAPBCLKCTRL register, if they are implemented

Diagram



Fields

Field	Function
31-3 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
2 CT32B1	ASYNCAPBCLKCTRL[CT32B1] Set Writing 1 to this register sets the ASYNCAPBCLKCTRL[CT32B1]. 0b - No effect. 1b - Set the ASYNCAPBCLKCTRL[CT32B1].
1 CT32B0	ASYNCAPBCLKCTRL[CT32B0] Set Writing 1 to this field sets the ASYNCAPBCLKCTRL[CT32B0] 0b - No effect. 1b - Set the ASYNCAPBCLKCTRL[CT32B0].
0 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

23.1.7 ASYNCAPBCLKCTRL Bits Clear Register (ASYNCAPBCLKCTRLCLR)

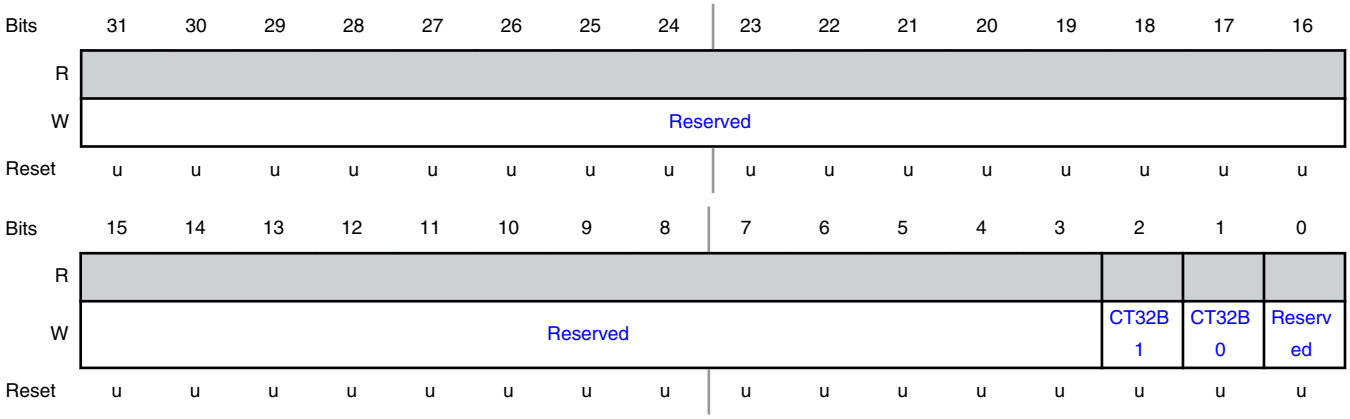
Offset

Register	Offset
ASYNCAPBCLKCTRLCLR	18h

Function

Clear bits in ASYNCAPBCLKCTRL. Writing ones to this register sets the corresponding bit or bits in the ASYNCAPBCLKCTRL register, if they are implemented

Diagram



Fields

Field	Function
31-3 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
2 CT32B1	ASYNCAPBCLKCTRL[CT32B1] Clear Writing 1 to this register clears the ASYNCAPBCLKCTRL[CT32B1] 0b - No effect. 1b - Clear the ASYNCAPBCLKCTRL[CT32B1].
1 CT32B0	ASYNCAPBCLKCTRL[CT32B0] Clear Writing 1 to this register clears the ASYNCAPBCLKCTRL[CT32B0]. 0b - No effect. 1b - Clear the ASYNCAPBCLKCTRL[CT32B0].

Table continues on the next page...

Table continued from the previous page...

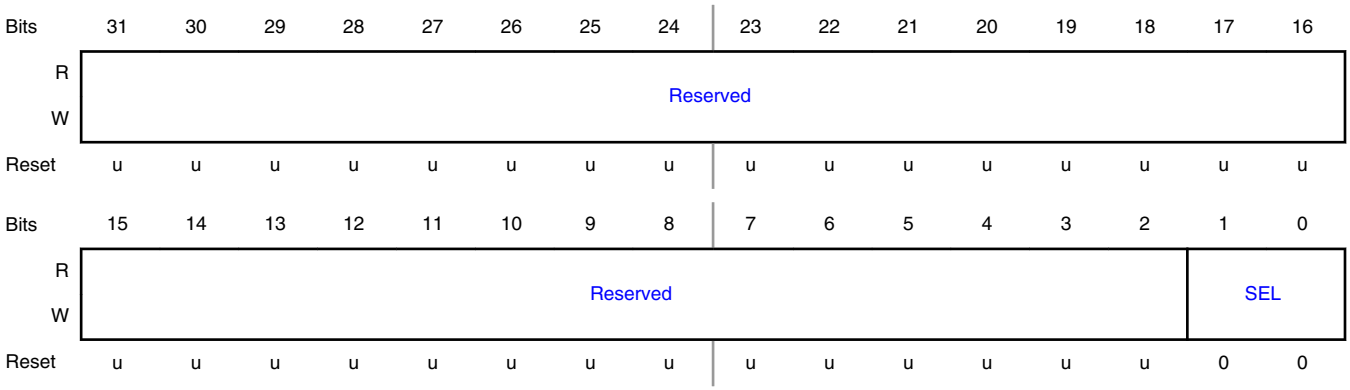
Field	Function
0 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

23.1.8 Asynchronous APB Clock Source Select Register (ASYNCAPBCLKSELA)

Offset

Register	Offset
ASYNCAPBCLKSELA	20h

Diagram



Fields

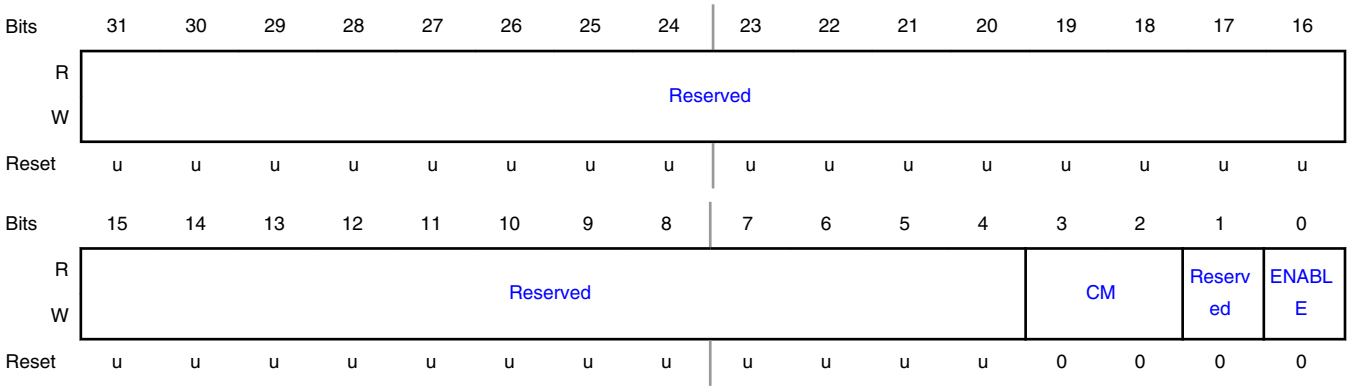
Field	Function
31-2 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
1-0 SEL	Clock Source Select Clock source for modules beyond asynchronous Bus bridge: ASYNC_SYSCON itself, counter/timers CTIMER 0/1. 00b - System bus clock 01b - 32 MHz crystal oscillator (XTAL32M). 10b - 32 MHz free running oscillator (FRO32M). 11b - 48 MHz free running oscillator (FRO48M).

23.1.9 Temperature Sensor Control Register (TEMPSENSORCTRL)

Offset

Register	Offset
TEMPSENSORCTRL	A0h

Diagram



Fields

Field	Function
31-4 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
3-2 CM	Temerature Sensor Common Mode Output Voltage Selection Set to 10b for proper use of the temperature sensor. 00b - High negative offset added. 01b - Intermediate negative offset added. 10b - No offset added. 11b - Low positive offset added.
1 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
0 ENABLE	Temperature Sensor Enable 0b - Diabled. 1b - Enabled.

23.1.10 NFC Tag Pads Control Register (NFCTAGPADSCTRL)

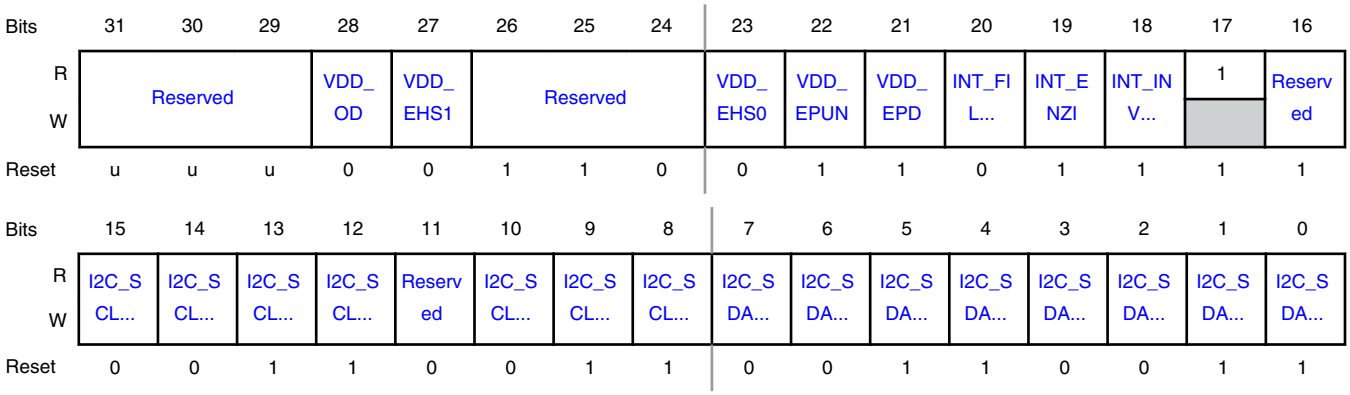
Offset

Register	Offset
NFCTAGPADSCTRL	A4h

Function

This register configures NFC tag pads control for I²C interface to internal NFC Tag (T parts only): I²C interface + 1 interrupt/ field detect input pad + NTAG VDD output pad.

Diagram



Fields

Field	Function
31-29 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
28 VDD_OD	NTAG VDD Open-drain Mode Control 0b - Normal. Normal push-pull output. 1b - Open-drain. Simulated open-drain output (high drive disabled).
27 VDD_EHS1	NTAG VDD IO Driver Slew Rate MSB VDD_EHS1 and VDD_EHS0 set IO cell speed when enabled as an output. It is rcommended to use default value (00b) 00b Low speed. 01b Nominal speed. 10b Fast speed. 11b High speed.

Table continues on the next page...

Table continued from the previous page...

Field	Function
26-24 —	RESERVED Reserved.
23 VDD_EHS0	NTAG VDD IO Driver Slew Rate LSB VDD_EHS1 and VDD_EHS0 set IO cell speed when enabled as an output. Recommendation is to use default value (00b) 00b Low speed. 01b Nominal speed. 10b Fast speed. 11b High speed.
22 VDD_EPUN	NTAG_VDD Enable Weak Pull-up on IO Pad Active Low
21 VDD_EPD	NTAG VDD Enable Weak Pull-down on IO Pad
20 INT_FILTEROFF	Reserved NTAG INT/FD IO cell always filters signal. This field should not be modified.
19 INT_ENZI	Reserved NTAG INT/FD IO cell always enabled. This field should not be modified.
18 INT_INVERT	NTAG INT/FD Input Polarity 0b - Input function is not inverted. 1b - Input function is inverted.
17 —	RESERVED Reserved. This field should not be modified.
16 —	RESERVED Reserved. This field should not be modified.
15 I2C_SCL_OD	I2C_SCL Open-drain Mode Control 0b - Normal. Normal push-pull output. 1b - Open-drain. Simulated open-drain output (high drive disabled).
14 I2C_SCL_EHS1	I2C_SCL IO Driver Slew Rate MSB Recommended setting is 0 to select slow slew rate.

Table continues on the next page...

Table continued from the previous page...

Field	Function
13 I2C_SCL_FILTEROFF	I2C_SCL Input Glitch Filter Control This field should not be modified.
12 I2C_SCL_ENZI	I2C_SCL Receiver Enable Active High This field should not be modified.
11 —	RESERVED Reserved. This field should not be modified.
10 I2C_SCL_EHS0	I2C_SCL IO Driver Slew Rate LSB Recommended setting is 0 to select slow slew rate.
9 I2C_SCL_EPUN	I2C_SCL Enable Weak Pull up on IO Pad, Active Low 0b - Pullup disabled. 1b - Pullup enabled.
8 I2C_SCL_EPD	I2C_SCL Enable Weak Pull Down on IO Pad 0b - Pull down disabled. 1b - Pull down enabled.
7 I2C_SDA_OD	I2C_SDA Controls Open-drain Mode 0b - Normal. Normal push-pull output. 1b - Open-drain. Simulated open-drain output (high drive disabled).
6 I2C_SDA_EHS1	I2C_SDA IO Driver Slew Rate MSB Recommended setting is 0 to select slow slew rate.
5 I2C_SDA_FILTEROFF	I2C_SDA Input Glitch Filter Control 0b - Filter enabled. Short noise pulses are filtered out. 1b - Filter disabled. No input filtering is done.
4 I2C_SDA_ENZI	I2C_SDA Receiver Enable Active High
3 I2C_SDA_INVERT	I2C_SDA Input Polarity 0b - Input function is not inverted. 1b - Input function is inverted.

Table continues on the next page...

Table continued from the previous page...

Field	Function
2 I2C_SDA_EHS 0	I2C_SDA IO Driver Slew Rate LSB Recommended setting is 0 to select slow slew rate.
1 I2C_SDA_EPU N	I2C_SDA Enable Weak Pull Up on IO Pad, Active Low 0b - Pullup disabled. 1b - Pullup enabled.
0 I2C_SDA_EPD	I2C_SDA Enable Weak Pull Down on IO Pad 0b - Pull down disabled. 1b - Pull down enabled.

23.1.11 XTAL 32 MHz LDO Control Register (XTAL32MLDOCTRL)

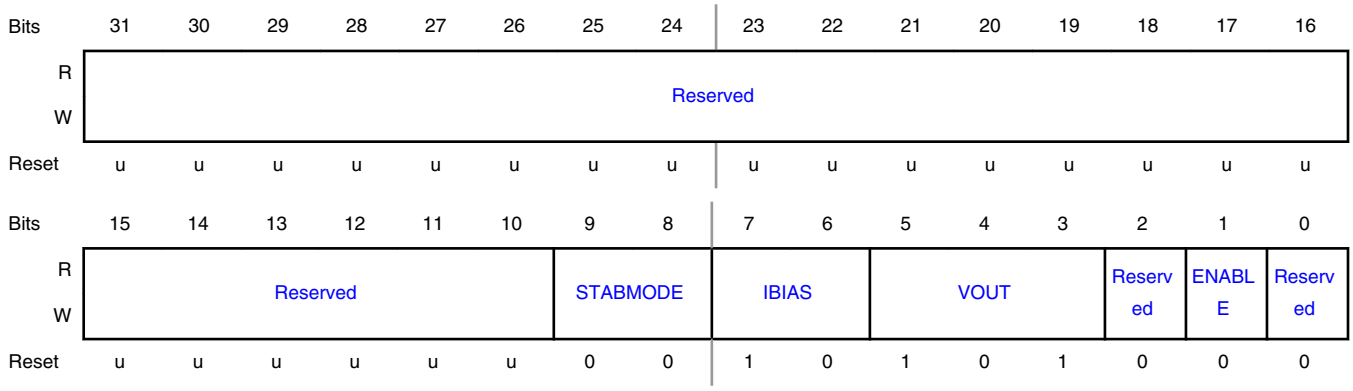
Offset

Register	Offset
XTAL32MLDOCTRL	A8h

Function

Control of the 32 MHz XTAL LDO. The XTAL will be auto-started on a power-up and settings in SYSCON_XTAL32MCTRL may need modifying before the full control by this register is possible.

Diagram



Fields

Field	Function
31-10 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
9-8 STABMODE	Stability Configuration This setting is managed by software API.
7-6 IBIAS	Adjust Biasing Current This setting is managed by software API.
5-3 VOUT	Adjust Output Voltage Level This setting is managed by software API.
2 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
1 ENABLE	Enable LDO Enable the LDO when set. Setting managed by software API.
0 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

23.1.12 32 MHz XTAL Control Register (XTAL32MCTRL)

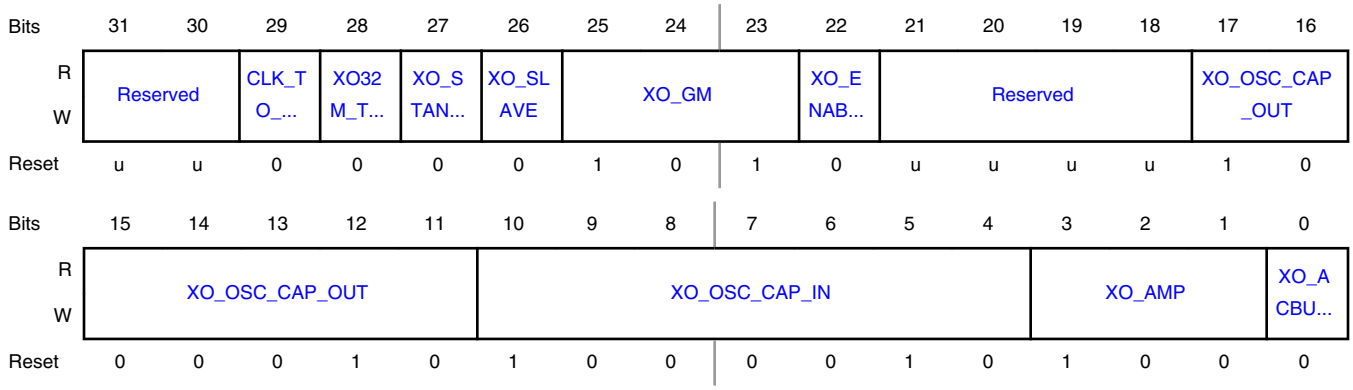
Offset

Register	Offset
XTAL32MCTRL	ACh

Function

Control of the 32 MHz XTAL. The XTAL will be auto-started on a power-up and settings in SYSCON_XTAL32MCTRL may need modifying before the full control by this register is possible.

Diagram



Fields

Field	Function
31-30 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
29 CLK_TO_GPADC_ENABLE	Enable 16 MHz Clock to General Purpose ADC Do not modify contents of this field, managed by API functions.
28 XO32M_TO_MCU_ENABLE	Enable 32MHz Clock to MCU and Clock Generators Do not modify contents of this field, managed by API functions.
27 XO_STANDALONE_ENABLE	Selection of the LDO and Core XO Reference Biasing Sources Do not modify contents of this field, managed by API functions.
26 XO_SLAVE	XTAL in Slave Mode Do not modify contents of this field, managed by API functions.
25-23 XO_GM	Gm Value for XTAL Do not modify contents of this field, managed by API functions.
22 XO_ENABLE	Enable Signal for 32 MHz XTAL Manual enable for 32 MHz XTAL. 0b - Disabled 1b - Enabled
21-18 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

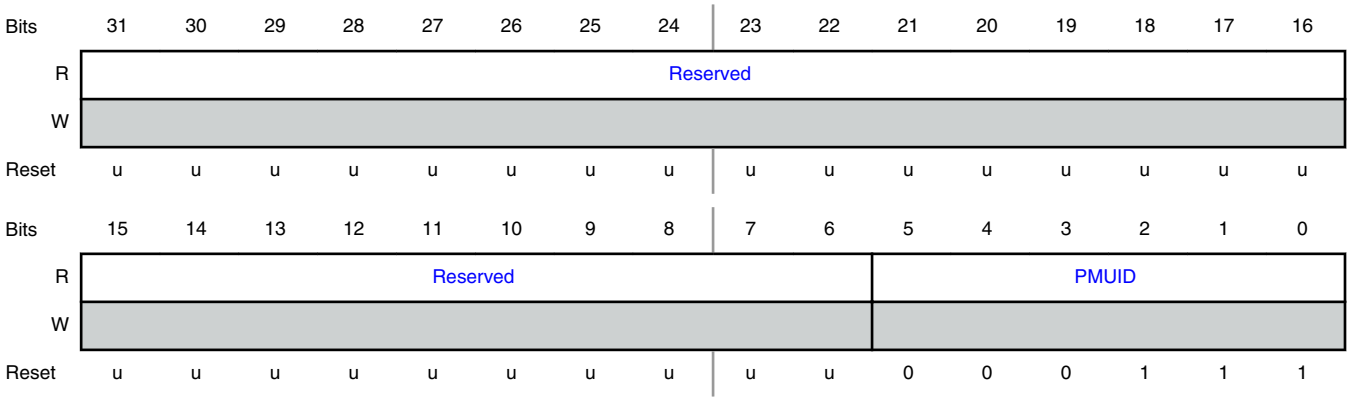
Field	Function
17-11 XO_OSC_CAP_OUT	Internal Capacitor Selection for XTAL_N. Internal Capacitor selection setting. Setting should be based on required capacitance value and trimming data. Recommended to use API function to manage this setting.
10-4 XO_OSC_CAP_IN	Internal Capacitor Selection for XTAL_P Internal Capacitor selection setting. Setting should be based on required capacitance value and trimming data. Recommended to use API function to manage this setting.
3-1 XO_AMP	Amplitude Selection Do not modify contents of this field, managed by API functions.
0 XO_ACBUF_PASS_ENABLE	Bypass enable of XTAL AC buffer enable in pll and top level Do not modify contents of this field, managed by API functions.

23.1.13 Analog Interfaces (PMU and Radio) Identity Registers (ANALOGID)

Offset

Register	Offset
ANALOGID	B0h

Diagram



Fields

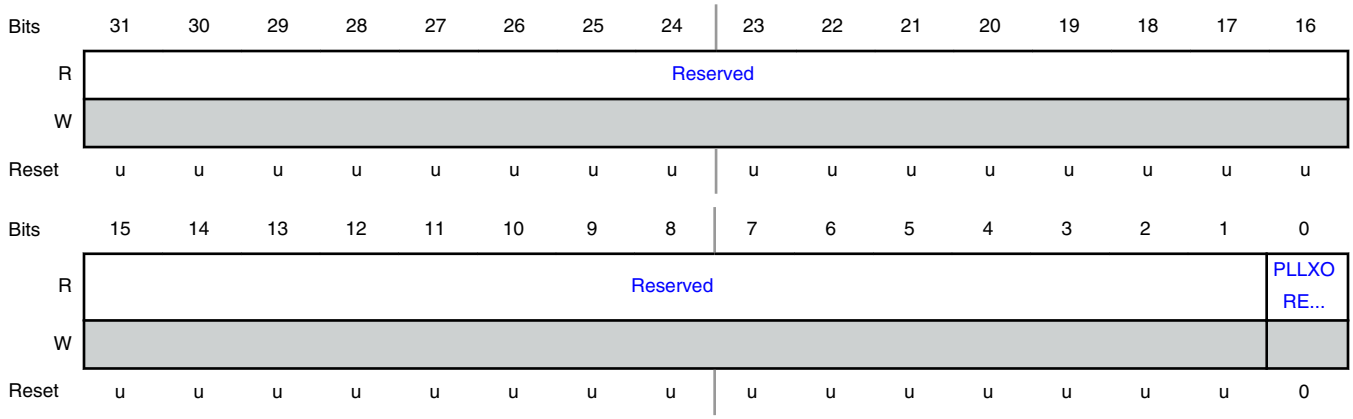
Field	Function
31-6 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
5-0 PMUID	PMU Identity This field indicates a version of the PMU.

23.1.14 Radio Analog Modules Status Register (RADIOSTATUS)

Offset

Register	Offset
RADIOSTATUS	B4h

Diagram



Fields

Field	Function
31-1 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
0 PLLXOREADY	32M XTAL Oscillator Status Output Value Value of status output by 32 MHz XTAL oscillator. Asserted to indicate that the clock is active. **NOTE** The quality of the 32 MHz clock may improve even after this is asserted. Additionally, if settings are changed, such as ibias control, this status flag will probably remain asserted even though changes to the clock signal occur.

23.1.15 DC Bus Control (DCBUSCTRL)

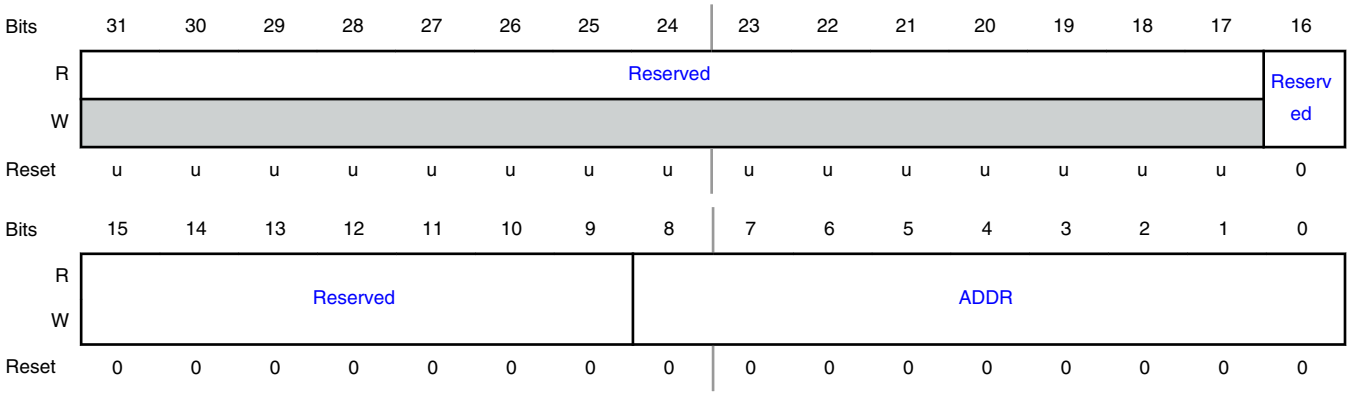
Offset

Register	Offset
DCBUSCTRL	BCh

Function

DC Bus can be used during device test and evaluation; it is not for use in application code.

Diagram



Fields

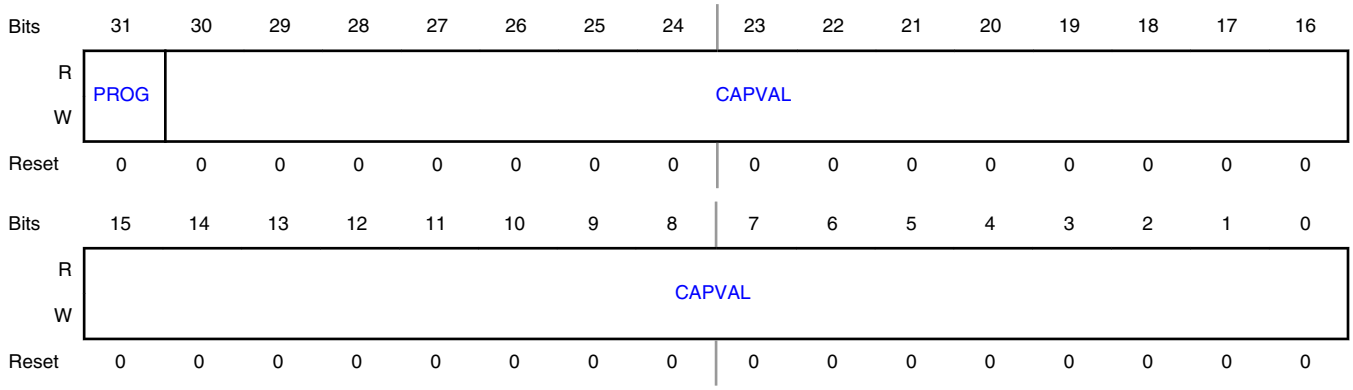
Field	Function
31-17 —	Reserved
16-9 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
8-0 ADDR	ADDR[8] should be set to 1 before entering power down to prevent the risk of a small amount of leakage current during power down. This setting is managed within the power APIs.

23.1.16 Frequency Measure Register (FREQMECTRL)

Offset

Register	Offset
FREQMECTRL	C0h

Diagram



Fields

Field	Function
31 PROG	Initiate Frequency Measurement Cycle Set this bit to 1 to initiate a frequency measurement cycle. Hardware clears this bit when the measurement cycle has completed and there is valid capture data in the CAPVAL field (bits 13:0).
30-0 CAPVAL	Frequency Measure Control and Status The function differs for a read and write operation. CAPVAL: FREQMECTRL[30:0] (Read-only) : Store the target counter result from the last frequency measure activiation, this is used in the calculation of the unknown clock frequency of the reference or target clock. SCALE: FREQMECTRL[4:0] (Write-only) : Define the count duration, 2^{SCALE}-1, that reference counter counts during measurement. Note that the value is 2 giving a minimum count 2^2-1 = 3. The result of freq_me_plus can be calculated as follows: freq_targetclk =freq_refclk x (CAPVAL+1) / (2^{SCALE}-1).

23.1.17 NFCTAG VDD Output Control Register (NFCTAG_VDD)

Offset

Register	Offset
NFCTAG_VDD	C8h

Diagram

Fields

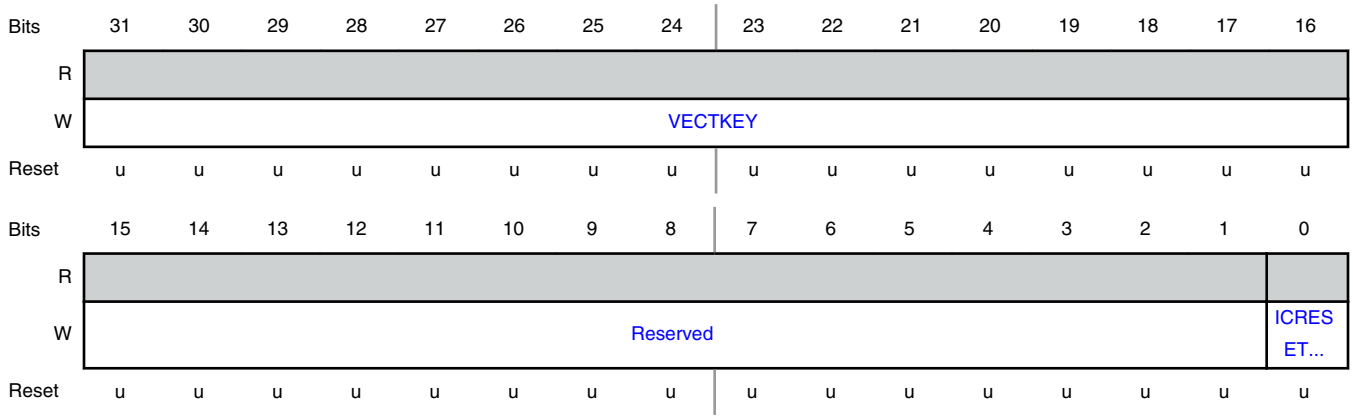
Field	Function
31-2 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
1 NFCTAG_VDD_OE	Output Enable for NFC Tag Vdd IO Cell
0 NFCTAG_VDD_OUT	NFC Tag Vdd IO Output Output value for the NFC Tag Vdd IO, if enabled with NFCTAG_VDD_OE.

23.1.18 Full IC Reset Request (from Software application) Register (SWRESETCTRL)

Offset

Register	Offset
SWRESETCTRL	CCh

Diagram



Fields

Field	Function
31-16 VECTKEY	Register Key On write, write 0x05FA to VECTKEY, otherwise the write is ignored.
15-1 —	RESERVED Reserved. User software should write zeroes to reserved bits.
0 ICRESETREQ	IC Reset Request This bit is only valid if VECTKEY is set correctly. Additionally, the software reset also requires PMC_CTRL[SWRRESETENABLE] to be set. 0b - No effect. 1b - Request a full IC reset level reset.

Chapter 24
Advanced Encryption Standard (AES)

24.1 AES register descriptions

24.1.1 AES memory map

AES base address: 4008_6000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Configuration Register (CFG)	32	RW	See description
4h	Command Register (CMD)	32	RW	See description
8h	Status Register (STAT)	32	RO	See description
Ch	Counter Increment Register (CTR_INCR)	32	RW	0000_0000h
20h - 3Ch	Key a Register (KEY0 - KEY7)	32	WO	0000_0000h
40h - 4Ch	Input Text a Register (INTEXT0 - INTEXT3)	32	WO	0000_0000h
50h - 5Ch	Holding a Register (HOLDING0 - HOLDING3)	32	WO	0000_0000h
60h - 6Ch	Output Text a Register (OUTTEXT0 - OUTTEXT3)	32	RO	0000_0000h
70h - 7Ch	GF128 Ya Register (GF128_Y0 - GF128_Y3)	32	WO	0000_0000h
80h - 8Ch	GF128 Za Register (GF128_Z0 - GF128_Z3)	32	RO	0000_0000h
90h - 9Ch	GCM Tag a Register (GCM_TAG0 - GCM_TAG3)	32	RO	0000_0000h

24.1.2 Configuration Register (CFG)

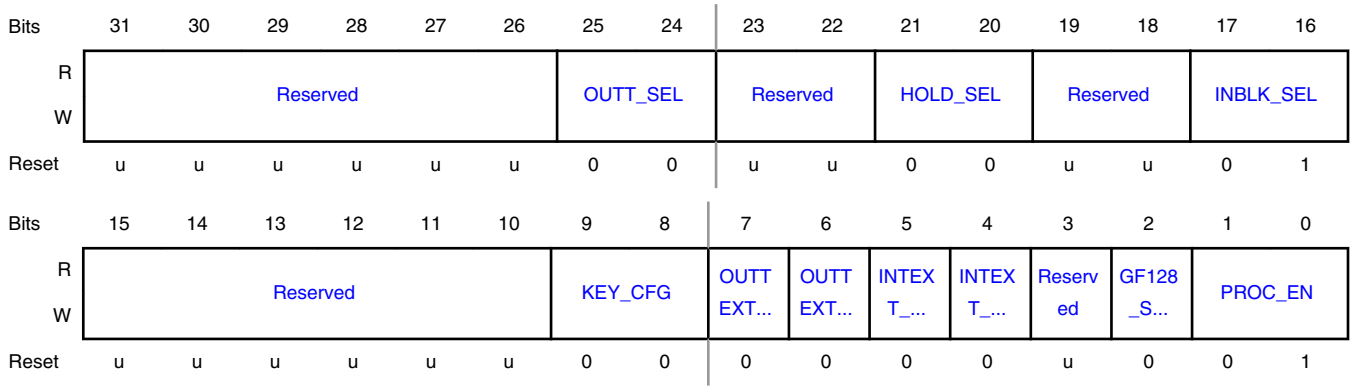
Offset

Register	Offset
CFG	0h

Function

This register Holds the configuration for processing and accepts word, short, and byte access. Must use byte access to write to bits 7:0 in order to avoid resetting parts of the engine. Alternatively, set the configuration when the block is not processing data.

Diagram



Fields

Field	Function
31-26 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
25-24 OUTT_SEL	Output Text Selection Writes to these bits perform an abort of the encrypt/decrypt logic and clear data. 00b - Output Block. 01b - Output Block XOR Input Text. 10b - Output Block XOR Holding. 11b - Reserved.
23-22 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
21-20 HOLD_SEL	Holding Select Writes to these bits perform an abort of the encrypt/decrypt logic and clear data. 00b - Counter. 01b - Input Text. 10b - Output Block. 11b - Input Text XOR Output Block.
19-18 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.

Table continues on the next page...

Table continued from the previous page...

Field	Function
17-16 INBLK_SEL	Input Block Selection Writes to these bits perform an abort of the encrypt/decrypt logic and clear data. 00b - Reserved. 01b - Input Text. 10b - Holding. 11b - Input Text XOR Holding.
15-10 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
9-8 KEY_CFG	Key Configuration Changes to these bits perform an abort of the encrypt/decrypt logic, clear data, and invalidate the key. 00b - 128 Bit Key. 01b - 192 Bit Key. 10b - 256 Bit Key. 11b - Reserved.
7 OUTTEXT_WSWAP	Output Text Word Swap If set then the output text has the words swapped.
6 OUTTEXT_BSWAP	Output Text Byte Swap If set then the output text has the bytes swapped.
5 INTEXT_WSWAP	Input Text Word Swap If set then the input text has the words swapped before it is used within the AES block.
4 INTEXT_BSWAP	Input Text Byte Swap If set then the input text has the bytes swapped before it is used within the AES block.
3 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
2 GF128_SEL	GF128 Select Mode 0b - GF128 Hash Input Text. 1b - GF128 Hash Output Text.

Table continues on the next page...

Table continued from the previous page...

Field	Function
1-0 PROC_EN	Processing Mode Enable 00b - Reserved. 01b - Encrypt/Decrypt Only. 10b - GF128 Hash Only. 11b - Encrypt/Decrypt and Hash.

24.1.3 Command Register (CMD)

Offset

Register	Offset
CMD	4h

Function

This register is used to send commands to the engine. Writing to this register can abort encryption operations and clear data.

Diagram

Fields

Field	Function
31-10 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
9 WIPE	Wipe Performs Abort, clear KEY, disable cipher, and clear GF128_Y

Table continues on the next page...

Table continued from the previous page...

Field	Function
8 ABORT	Abort Abort Encrypt/Decrypt and GF128 Hash, clear INTEXT, clear OUTTEXT, and clear HOLDING
7-5 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
4 SWITCH_MODE	Switch Mode Switch mode from Forward to Reverse or from Reverse to Forward. Must wait for Idle after command. Typically used for non-counter modes (ECB, CBC, CFB, OFB) to switch from forward to reverse mode for decryption.
3-2 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
1 COPY_TO_Y	Copies Output Text to GF128 Y Typically used for GCM where the Hash requires a Y input which is the result of an ECB encryption of 0s. Should be performed after encryption of 0s. Writes to this bit performs an abort of the encrypt/decrypt logic and clears data and an abort of the gf128 logic.
0 COPY_SKEY	Copy Secret Key and Enable Cipher Secret key is held in OTP and is copied to the AES block by setting this bit. It is necessary to do this after initial powerup, powerdown cycles and internal resets.

24.1.4 Status Register (STAT)

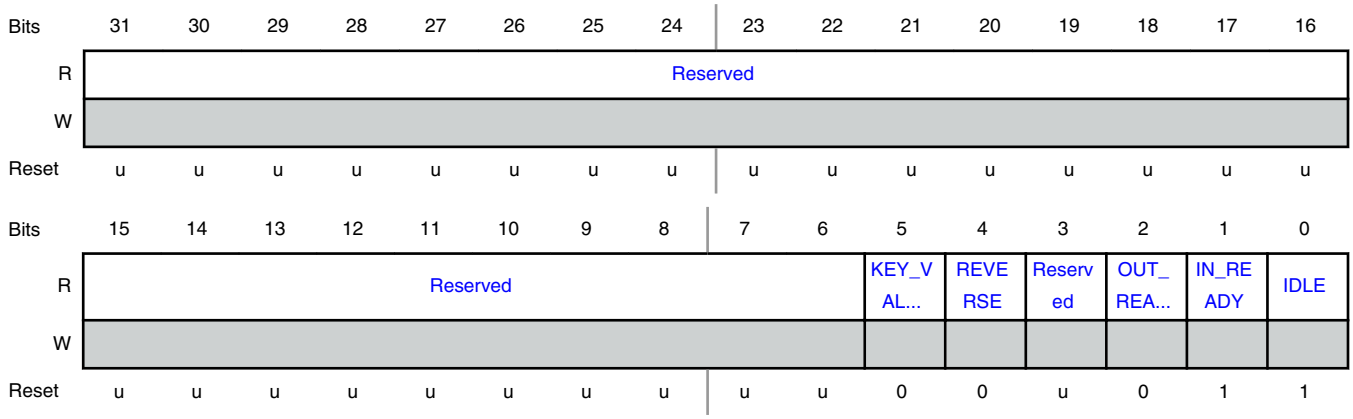
Offset

Register	Offset
STAT	8h

Function

This read only register indicates the status of the AES operations.

Diagram



Fields

Field	Function
31-6 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
5 KEY_VALID	Key Valid When set, Key is valid
4 REVERSE	Reverse When set, Cipher in reverse mode
3 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
2 OUT_READY	Output Ready When set, output Text can be read
1 IN_READY	Input Ready When set, input Text can be written
0 IDLE	Idle When set, all state machines are idle

24.1.5 Counter Increment Register (CTR_INCR)

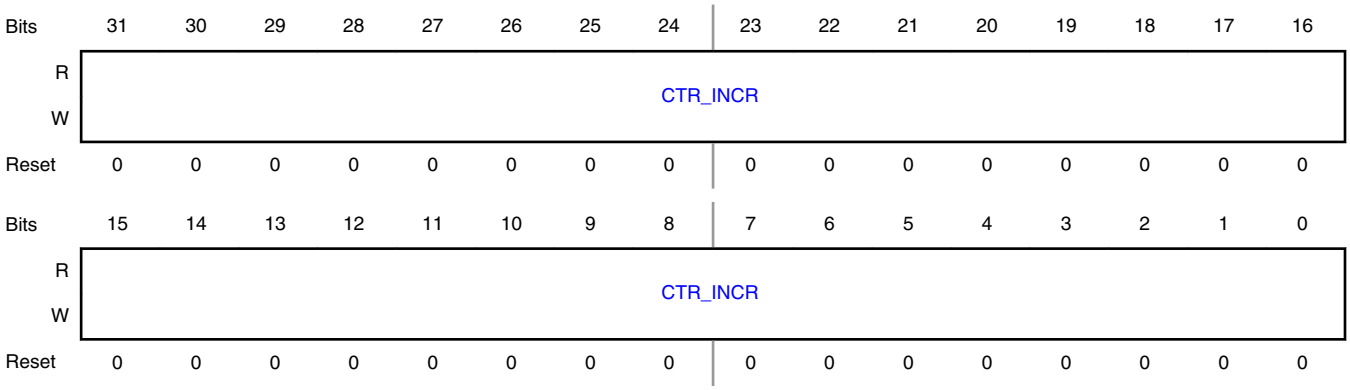
Offset

Register	Offset
CTR_INCR	Ch

Function

Increment value for HOLDING when in Counter modes

Diagram



Fields

Field	Function
31-0 CTR_INCR	Control Increment

24.1.6 Key a Register (KEY0 - KEY7)

Offset

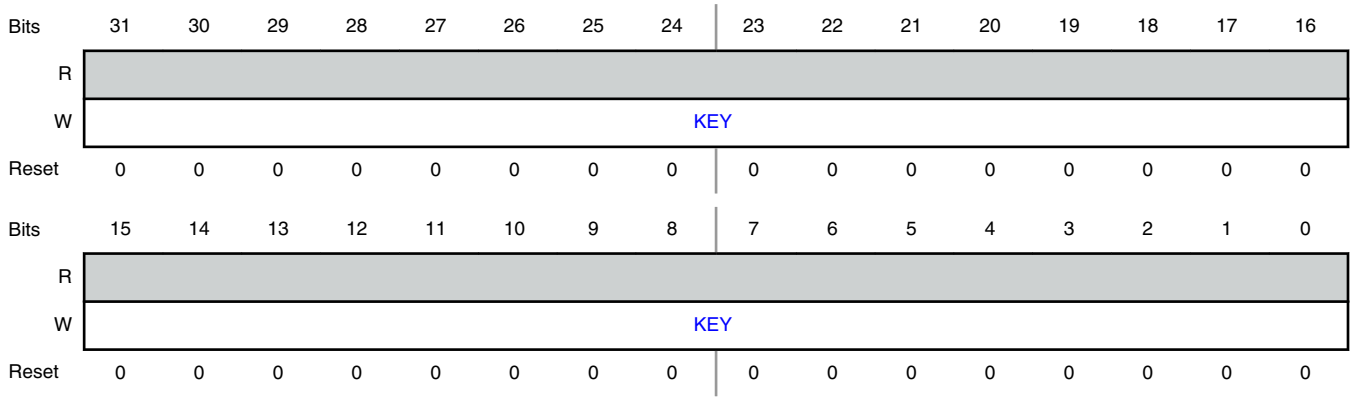
For a = 0 to 7:

Register	Offset
KEYa	20h + (a × 4h)

Function

Key [32xa+31:32xa]. The key will be enabled by writing sequentially KEY0, KEY1, KEY2,

Diagram



Fields

Field	Function
31-0 KEY	KEY

24.1.7 Input Text a Register (INTEXT0 - INTEXT3)

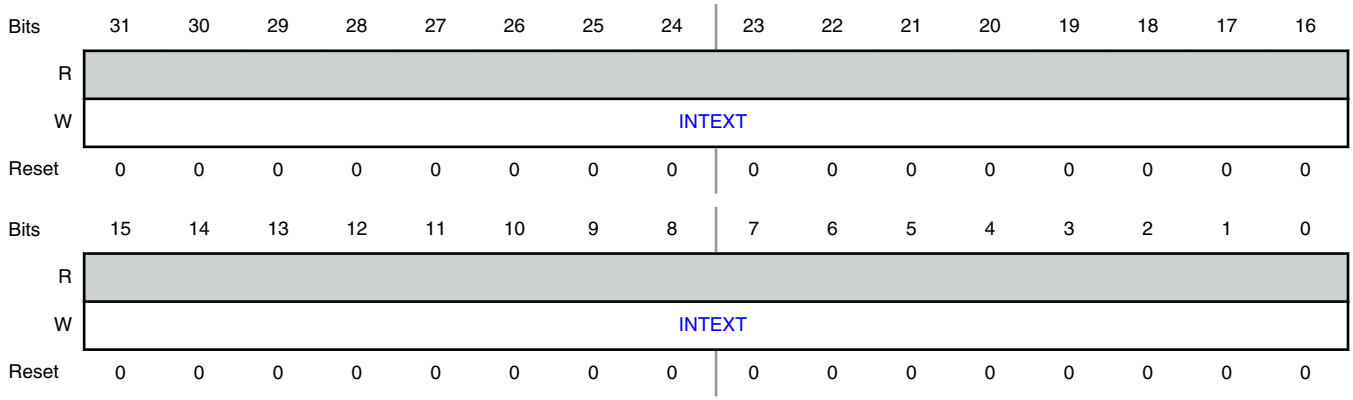
Offset

Register	Offset
INTEXT0	40h
INTEXT1	44h
INTEXT2	48h
INTEXT3	4Ch

Function

Input Text [32xa+31:32xa]. Contains the input data for processing. Typically holds plaintext when encrypting and ciphertext when decrypting

Diagram



Fields

Field	Function
31-0 INTEXT	IN

24.1.8 Holding a Register (HOLDING0 - HOLDING3)

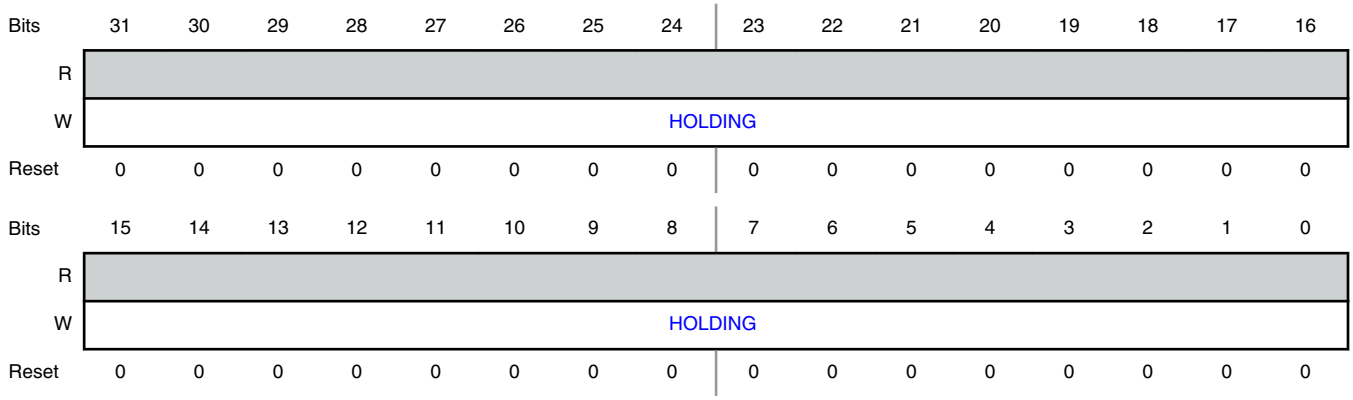
Offset

Register	Offset
HOLDING0	50h
HOLDING1	54h
HOLDING2	58h
HOLDING3	5Ch

Function

Holding [32xa+31:32xa]. Temporary storage used for processing. Begins with Initialization Vector (IV).

Diagram



Fields

Field	Function
31-0 HOLDING	HOLDING

24.1.9 Output Text a Register (OUTTEXT0 - OUTTEXT3)

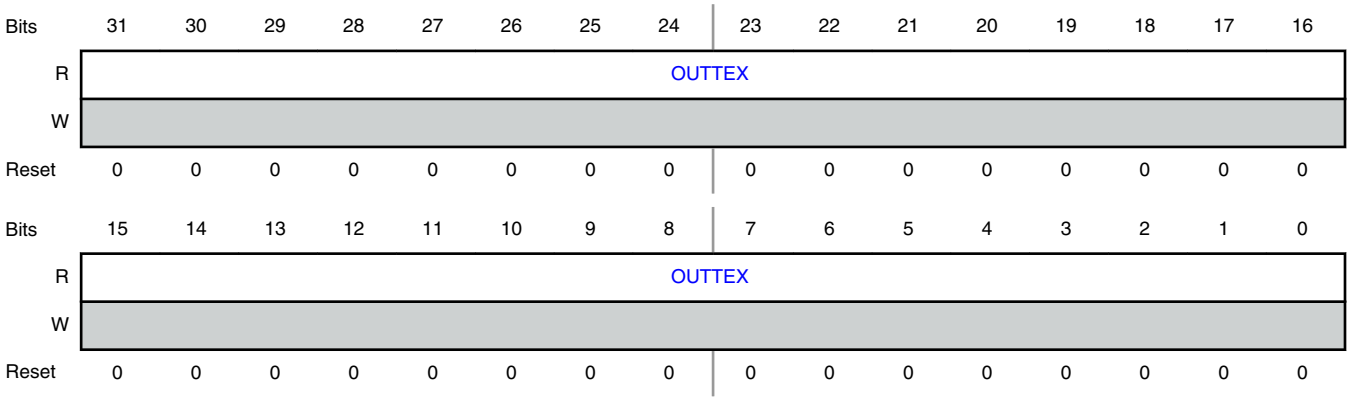
Offset

Register	Offset
OUTTEXT0	60h
OUTTEXT1	64h
OUTTEXT2	68h
OUTTEXT3	6Ch

Function

Output Text [32xa+31:32xa]. Contains the output data from processing. Typically holds ciphertext when encrypting and plaintext when decrypting.

Diagram



Fields

Field	Function
31-0 OUTTEX	OUT

24.1.10 GF128 Ya Register (GF128_Y0 - GF128_Y3)

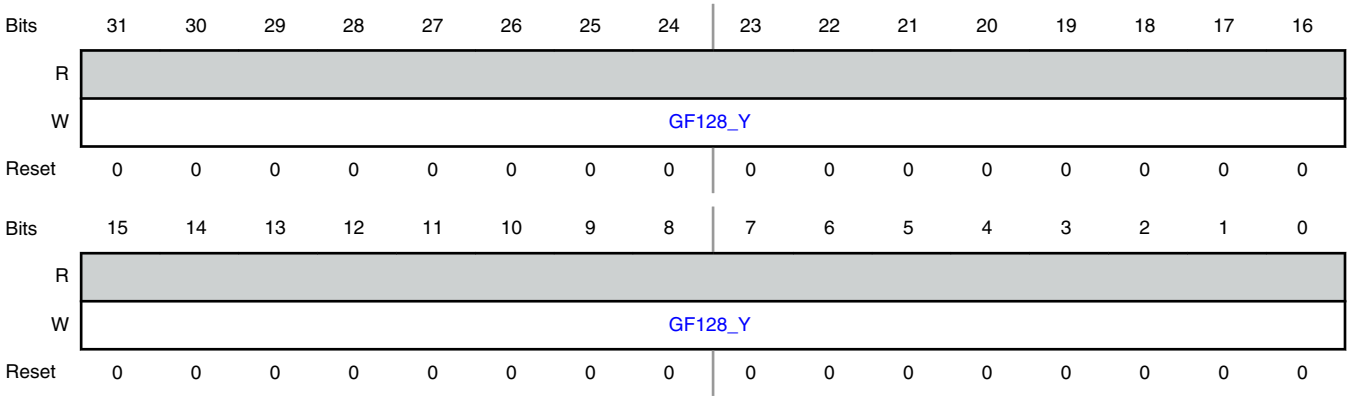
Offset

Register	Offset
GF128_Y0	70h
GF128_Y1	74h
GF128_Y2	78h
GF128_Y3	7Ch

Function

GF128 Y [32xa+31:32xa]. Contains Y input of GF128 hash.

Diagram



Fields

Field	Function
31-0 GF128_Y	GF128_Y

24.1.11 GF128 Za Register (GF128_Z0 - GF128_Z3)

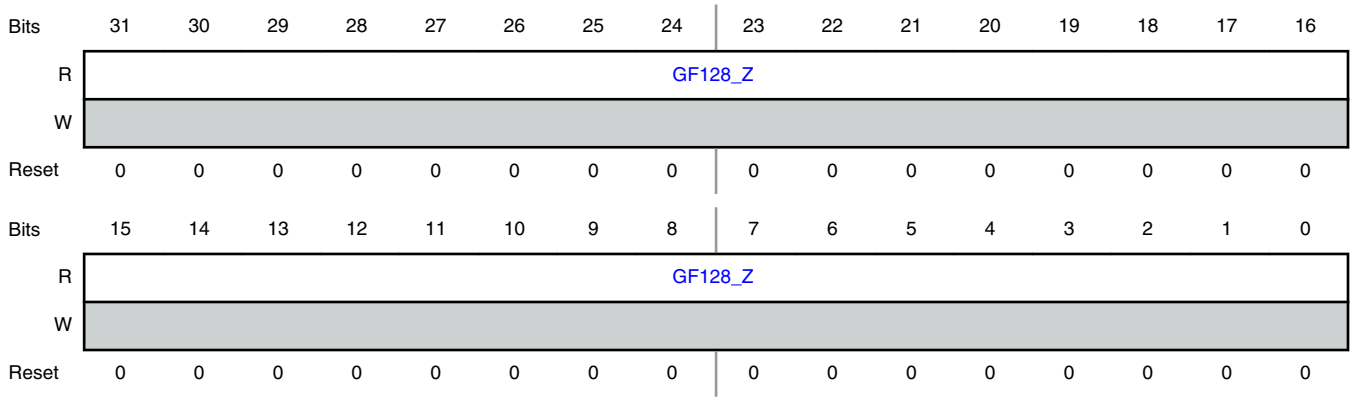
Offset

Register	Offset
GF128_Z0	80h
GF128_Z1	84h
GF128_Z2	88h
GF128_Z3	8Ch

Function

GF128 Z [32xa+31:32xa]. Holds the results of the GF-128 hash. Used in GCM modes for authentication.

Diagram



Fields

Field	Function
31-0 GF128_Z	GF128_Z

24.1.12 GCM Tag a Register (GCM_TAG0 - GCM_TAG3)

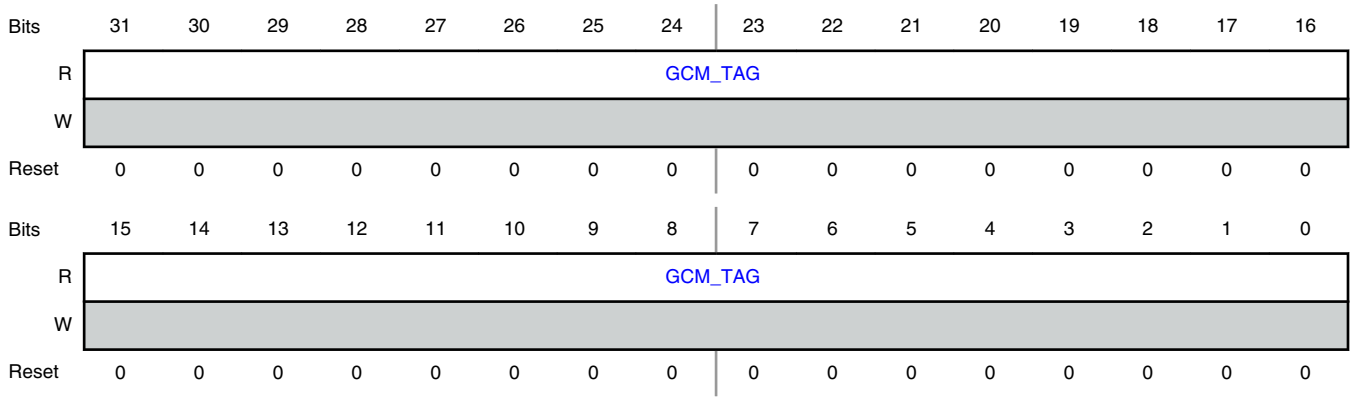
Offset

Register	Offset
GCM_TAG0	90h
GCM_TAG1	94h
GCM_TAG2	98h
GCM_TAG3	9Ch

Function

GCM Tag [32xa+31:32xa]. Calculated by XORing Output Text and GF128 Z.

Diagram



Fields

Field	Function
31-0 GCM_TAG	GCM_TAG0

Chapter 25
Infra-Red Modulator (CIC_IRB)

25.1 CIC_IRB register descriptions

25.1.1 CIC_IRB memory map

CIC_IRB base address: 4000_7000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Infra-Red Modulator Configuration Register (CONF)	32	RW	See description
4h	Carrier Configuration Register (CARRIER)	32	RW	See description
8h	Infra-Red Modulator Envelope FIFO Input Register (FIFO_IN)	32	RW	See description
Ch	Infra-Red Modulator Status Register (STATUS)	32	RO	See description
10h	Infra-Red Modulator Commands Register (CMD)	32	WO	See description
FE0h	Interrupt Status Register (INT_STATUS)	32	RO	See description
FE4h	Interrupt Enable Register (INT_ENA)	32	RW	See description
FE8h	Interrupt Clear Register (INT_CLR)	32	WO	See description
FECh	Interrupt Set Register (INT_SET)	32	WO	See description
FFCh	IR Blaster Module Identifier Register (MODULE_ID)	32	RO	0131_3000h

25.1.2 Infra-Red Modulator Configuration Register (CONF)

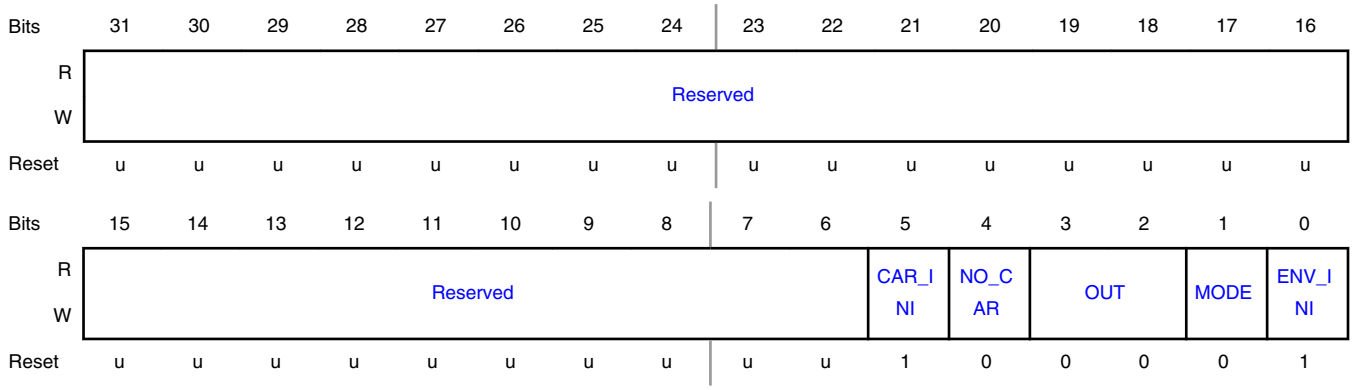
Offset

Register	Offset
CONF	0h

Function

This register configures the Infra-Red Modulator.

Diagram



Fields

Field	Function
31-6 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
5 CAR_INI	Initial Carrier Value 0b - Carrier starts with '0' (the carrier is low during CARRIER[CHIGH] and high during CARRIER[CLOW]) 1b - Carrier starts with '1' (the carrier is high during CARRIER[CHIGH] and low during CARRIER[CLOW])
4 NO_CAR	No Carrier 0b - Normal. IR_Blaster output = envelope + carrier 1b - Carrier is inhibited. IR_Blaster output = envelope only
3-2 OUT	Output Logic Function 00b - envelope AND carrier 01b - envelope OR carrier 10b - envelope NAND carrier 11b - envelope NOR carrier
1 MODE	Blaster Mode 0b - IR Blaster will stop when it encouters an envelope with FIFO_IN[ENV_LAST] = 1. 1b - Automatic restart. IR Blaster will transmit all envelopes and stop only when FIFO becomes empty. FIFO_IN[ENV_LAST] bit only generates an interrupt but doesn't stop transmission.
0 ENV_INI	Initial Envelope Value This is the level of the first envelope after IR Blaster starts or restarts. 0b - First envelope is in low level. 1b - First envelope is in high level.

25.1.3 Carrier Configuration Register (CARRIER)

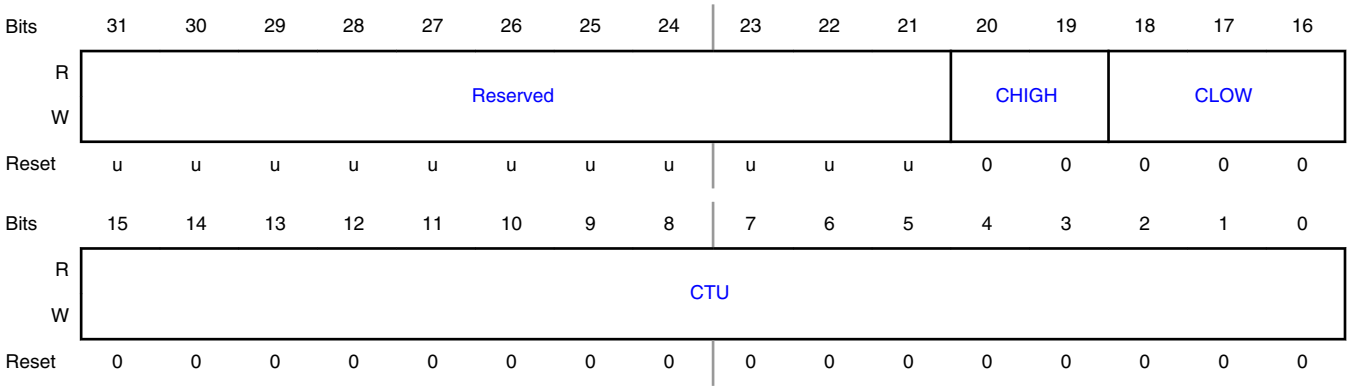
Offset

Register	Offset
CARRIER	4h

Function

This register configures the high and low times and the carrier time unit.

Diagram



Fields

Field	Function
31-21 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
20-19 CHIGH	Carrier High Period Carrier high level duration = (CHIGH + 1) * CTU. It is recommended to modify this field when the blaster unit is disabled (i.e when STATUS[ENA_ST] = 0)
18-16 CLOW	Carrier Low Period Carrier low level duration = (CLOW + 1) * CTU. It is recommended to modify this field when the blaster unit is disabled (i.e when STATUS[ENA_ST] = 0)
15-0 CTU	Carrier Time Unit Length Carrier Time Unit = CTU x TIRCP, TIRCP = IR module clock period, e.g. 1/48 MHz. Value 0x0 is equivalent to 0x1. It is recommended to modify this field when the blaster unit is disabled (i.e when STATUS[ENA_ST] = 0)

25.1.4 Infra-Red Modulator Envelope FIFO Input Register (FIFO_IN)

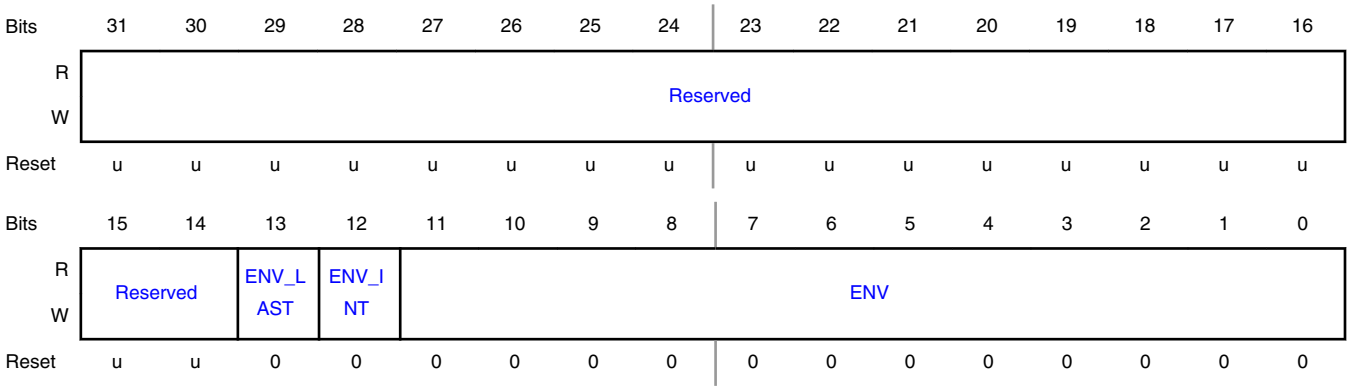
Offset

Register	Offset
FIFO_IN	8h

Function

This register writes data to the FIFO. This data will determine the sequence on the output.

Diagram



Fields

Field	Function
31-14 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
13 ENV_LAST	Last Envelope 0b - IR Blaster loads the next envelope when this envelope finishes. 1b - IR Blaster stops and generates an interrupt when this envelope is completly transmitted. If CONF[MODE] = 1, IR Blaster only generates an interrupt.
12 ENV_INT	Envelope Interrupt Generate an interrupt when starting emission of the envelope. 0b - Not generate interrupt 1b - Generate interrupt
11-0 ENV	Envelope Duration Envelope duration expressed in carrier period number. $T_{envelope}$ = ENV x (CHIGH + CLOW + 2) x CTU. Value 0x000 has the same behaviour as the value 0x001.

25.1.5 Infra-Red Modulator Status Register (STATUS)

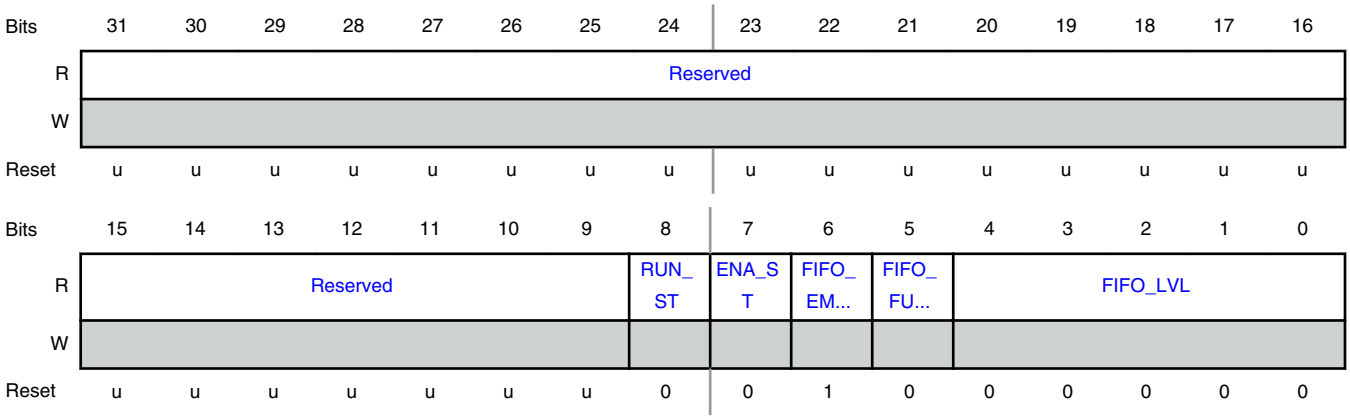
Offset

Register	Offset
STATUS	Ch

Function

This register indicates the status of the Infra-Red Modulator.

Diagram



Fields

Field	Function
31-9 —	RESERVED Reserved. The value read from a reserved bit is not defined.
8 RUN_ST	IR Blaster Run Status 0b - IR Blaster is not running. The transmission has not been started or has been finished. 1b - IR Blaster is running.
7 ENA_ST	IR Blaster Status 0b - IR Blaster is disabled 1b - IR Blaster is enabled
6 FIFO_EMPTY	IR Blaster FIFO Empty Flag 0b - FIFO is not empty 1b - FIFO is empty (FIFO level = 000000)

Table continues on the next page...

Table continued from the previous page...

Field	Function
5 FIFO_FULL	IR Blaster FIFO Full Flag 0b - FIFO is not full 1b - FIFO is full (FIFO level = 100000)
4-0 FIFO_LVL	Current IR Blaster FIFO Level 00000b - Empty 00001b - 1 entry ..... 10000b - 16 entries/FIFO is full 10001b-11111b - Not valid

25.1.6 Infra-Red Modulator Commands Register (CMD)

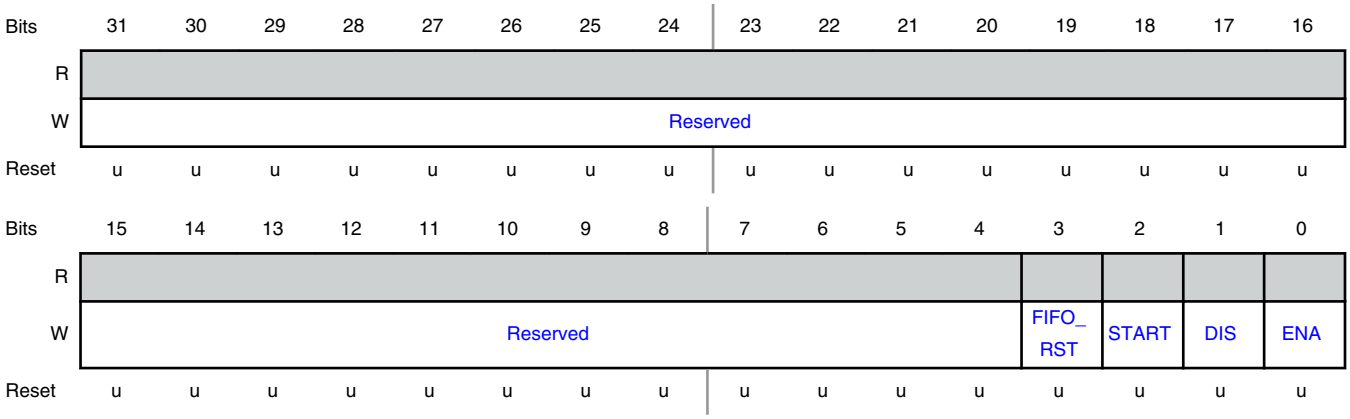
Offset

Register	Offset
CMD	10h

Function

This register issues commands to the Infra-Red Modulator, such as Start and Disable.

Diagram



Fields

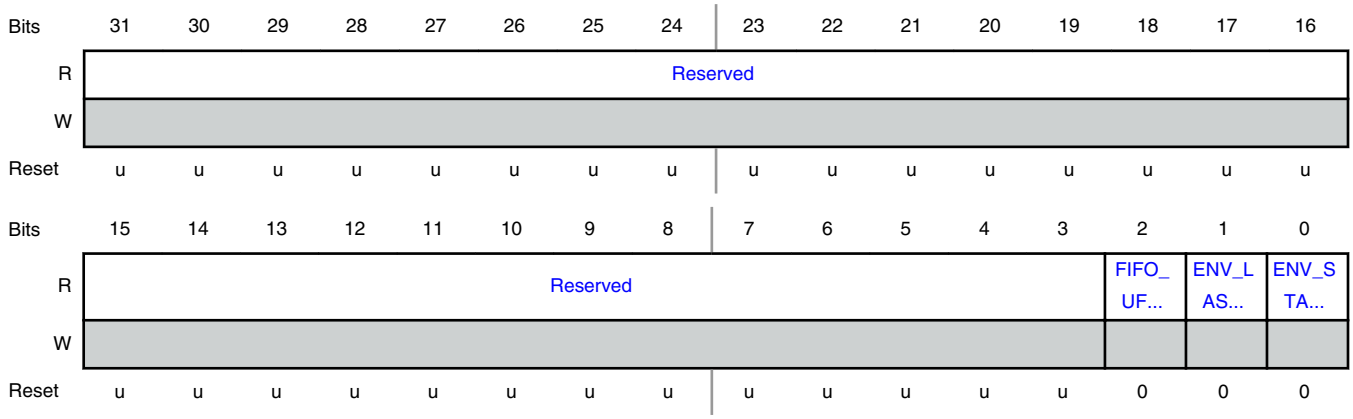
Field	Function
31-4 —	RESERVED Reserved. User software should write zeroes to reserved bits.
3 FIFO_RST	Reset IR Blaster FIFO This bit is self clearing. 0b - No effect 1b - Reset FIFO. IR Blaster FIFO is completly re-initialized, all data in the FIFO are erased.
2 START	Start IR Blaster This bit is self clearing. 0b - No effect 1b - Start transmission. Before setting this field, the blaster must be enabled (the bit field ENA must be set first).
1 DIS	Disable IR Blaster This bit is self clearing. 0b - No effect 1b - Disable IR Blaster. The transmission of envelopes is immediatly stopped. The FIFO is not reinitialized (the content of the FIFO is conserved).
0 ENA	Enable IR Blaster This bit is self clearing. 0b - No effect 1b - Enable IR Blaster

25.1.7 Interrupt Status Register (INT_STATUS)

Offset

Register	Offset
INT_STATUS	FE0h

Diagram



Fields

Field	Function
31-3 —	RESERVED Reserved. The value read from a reserved bit is not defined.
2 FIFO_UFL_INT	IR Blaster FIFO Underflow (FIFO_UFL) Interrupt This interrupt occurs when IR Blaster tries to transmit data but the FIFO is empty. 0b - Interrupt is not pending 1b - Interrupt is pending
1 ENV_LAST_INT	Envelop Last (ENV_LAST) Interrupt The interrupt occurs when IR Blaster has finished transmitting an envelope with FIFO_IN[ENV_LAST] = 1. 0b - Interrupt is not pending 1b - Interrupt is pending
0 ENV_START_INT	Envelop Start (ENV_START) Interrupt This interrupt occurs when IR Blaster has started to transmit an envelope with FIFO_IN[ENV_INT] = 1 0b - Interrupt is not pending 1b - Interrupt is pending

25.1.8 Interrupt Enable Register (INT_ENA)

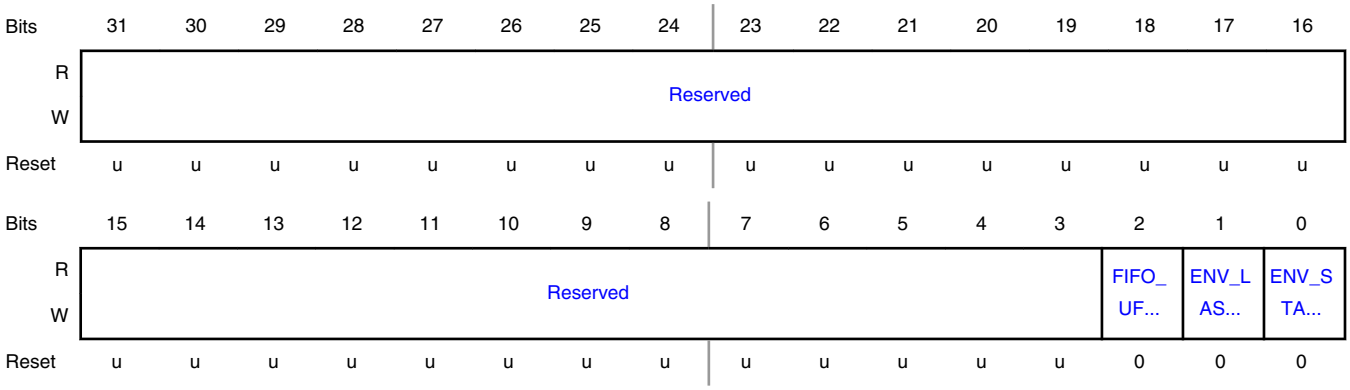
Offset

Register	Offset
INT_ENA	FE4h

Function

This register enables or disables the interrupts described in the INT_STATUS register.

Diagram



Fields

Field	Function
31-3 —	RESERVED Reserved. User software should write zeroes to reserved bits. The value read from a reserved bit is not defined.
2 FIFO_UFL_ENA	Enable/Disable FIFO_UFL Interrupt 0b - Disable FIFO_UFL interrupt 1b - Enable FIFO_UFL interrupt
1 ENV_LAST_ENA	Enable/Disable ENV_LAST Interrupt 0b - Disable ENV_LAST interrupt 1b - Enable ENV_LAST interrupt
0 ENV_START_ENA	Enable/Disable ENV_START Interrupt 0b - Disable ENV_START interrupt 1b - Enable ENV_START interrupt

25.1.9 Interrupt Clear Register (INT_CLR)

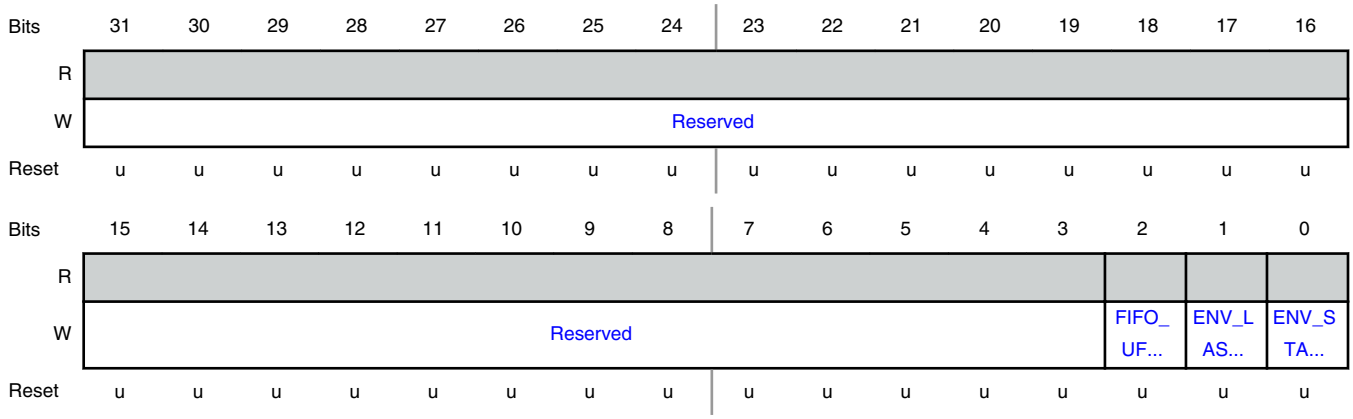
Offset

Register	Offset
INT_CLR	FE8h

Function

This register clears any interrupts that are set in the INT_STATUS register. This is a write only register; write a 1 to a bit location to clear the associated interrupt status flag.

Diagram



Fields

Field	Function
31-3 —	RESERVED Reserved. User software should write zeroes to reserved bits.
2 FIFO_UFL_CLR	Clear FIFO_UFL Interrupt This bit is self clearing. 0b - No effect 1b - Clear FIFO_UFL interrupt
1 ENV_LAST_CLR	Clear ENV_LAST Interrupt This bit is self clearing. 0b - No effect 1b - Clear ENV_LAST interrupt
0 ENV_START_CLR	Clear ENV_START interrupt This bit is self clearing. 0b - No effect 1b - Clear ENV_START interrupt

25.1.10 Interrupt Set Register (INT_SET)

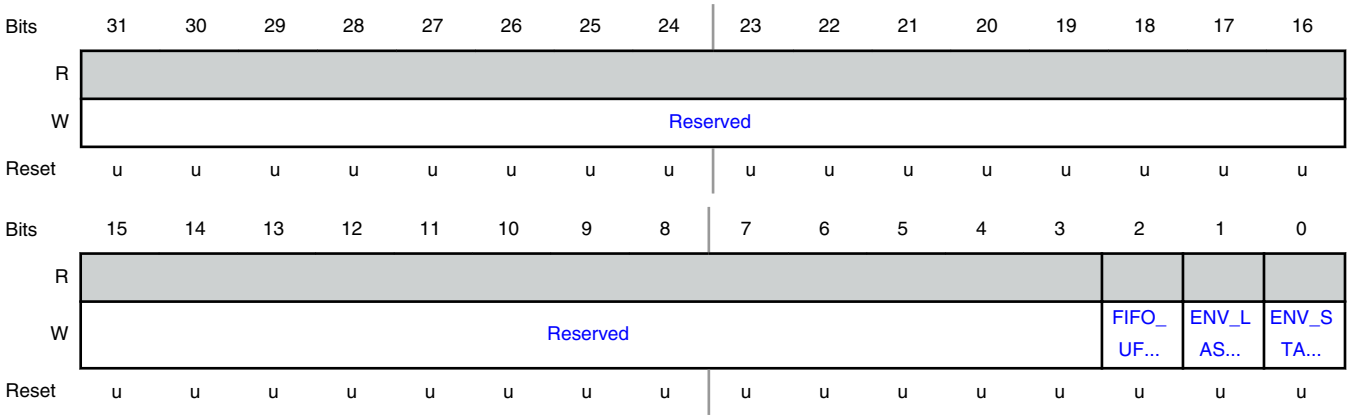
Offset

Register	Offset
INT_SET	FECh

Function

This register sets any interrupts in the INT_STATUS register. This is a write only register; write a 1 to a bit location to set the associated interrupt status flag. This could be used during software development to generate specific interrupts.

Diagram



Fields

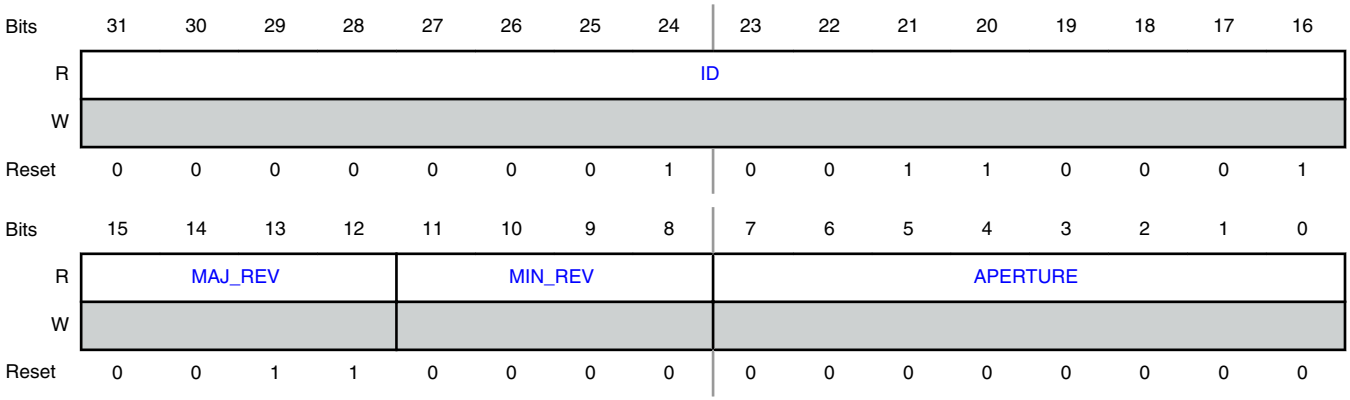
Field	Function
31-3 —	RESERVED Reserved. User software should write zeroes to reserved bits.
2 FIFO_UFL_SET	Set FIFO_UFL Interrupt This bit is self clearing. 0b - No effect 1b - Set FIFO_UFL interrupt
1 ENV_LAST_SET	Set ENV_LAST Interrupt This bit is self clearing. 0b - No effect 1b - Set ENV_LAST interrupt
0 ENV_START_SET	Set ENV_START Interrupt This bit is self clearing. 0b - No effect 1b - Set ENV_START interrupt

25.1.11 IR Blaster Module Identifier Register (MODULE_ID)

Offset

Register	Offset
MODULE_ID	FFCh

Diagram



Fields

Field	Function
31-16 ID	Identifier This is the unique identifier of the module.
15-12 MAJ_REV	Major revision Major revision implies software modifications.
11-8 MIN_REV	Minor Revision Minor revision without software consequences
7-0 APERTURE	Aperture Aperture number minus 1 of consecutive packets 4 KB reserved for this IP.

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 06/2020
Document identifier: UM11138

arm