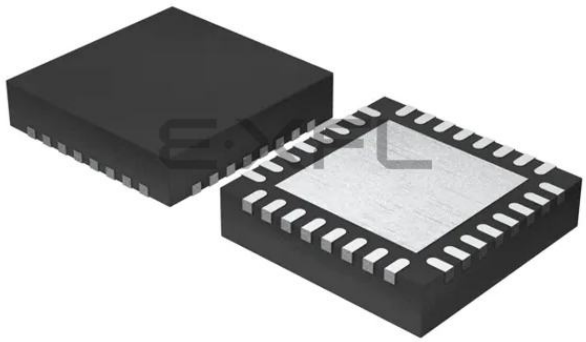


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Discontinued at Digi-Key
Core Processor	ARM® Cortex®-M0+
Core Size	32-Bit Single-Core
Speed	24MHz
Connectivity	EBI/EMI, I²C, IrDA, SmartCard, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, DMA, I²S, POR, PWM, WDT
Number of I/O	24
Program Memory Size	4KB (4K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	1.98V ~ 3.8V
Data Converters	A/D 4x12b; D/A 1x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	32-VQFN Exposed Pad
Supplier Device Package	32-QFN (6x6)
Purchase URL	https://www.e-xfl.com/product-detail/silicon-labs/efm32zg210f4-qfn32t

Table 3.3. Minor Revision Number Interpretation

Minor Rev[7:0]	Revision
0x00	A

the period, i.e. for the 1 us PERIOD, the number of cycles should at least span 1.1 us, and for the 5 us period they should span at least 5.5 us. For the 1 MHz band, PERIOD in MSC_TIMEBASE should be set to 5US, while it should be set to 1US for all other AUXHFRCO bands.

Both page erase and write operations require that the address is written into the MSC_ADDRB register. For erase operations, the address may be any within the page to be erased. Load the address by writing 1 to the LADDRIM bit in the MSC_WRITECMD register. The LADDRIM bit only has to be written once when loading the first address. After each word is written the internal address register ADDR will be incremented automatically by 4. The INVADDR bit of the MSC_STATUS register is set if the loaded address is outside the flash and the LOCKED bit of the MSC_STATUS register is set if the page addressed is locked. Any attempts to command erase of or write to the page are ignored if INVADDR or the LOCKED bits of the MSC_STATUS register are set. To abort an ongoing erase, set the ERASEABORT bit in the MSC_WRITECMD register.

When a word is written to the MSC_WDATA register, the WDATAREADY bit of the MSC_STATUS register is cleared. When this status bit is set, software or DMA may write the next word.

A single word write is commanded by setting the WRITEONCE bit of the MSC_WRITECMD register. The operation is complete when the BUSY bit of the MSC_STATUS register is cleared and control of the flash is handed back to the AHB interface, allowing application code to resume execution.

For a DMA write the software must write the first word to the MSC_WDATA register and then set the WRITETRIG bit of the MSC_WRITECMD register. DMA triggers when the WDATAREADY bit of the MSC_STATUS register is set.

It is possible to write words twice between each erase by keeping at 1 the bits that are not to be changed. Let us take as an example writing two 16 bit values, 0xAAAA and 0x5555. To safely write them in the same flash word this method can be used:

- Write 0xFFFFAAAA (word in flash becomes 0xFFFFAAAA)
- Write 0x5555FFFF (word in flash becomes 0x5555AAAA)

Note that there is a maximum of two writes to the same word between each erase due to a physical limitation of the flash.

Note

During a write or erase, flash read accesses will be stalled, effectively halting code execution from flash. Code execution continues upon write/erase completion. Code residing in RAM may be executed during a write/erase operation.

Note

The MSC_WDATA and MSC_ADDRB registers are not retained when entering EM2 or lower energy modes.

7.3.5.1 Mass erase

A mass erase can be initiated from software using ERASEMAIN0 in MSC_WRITECMD. This command will start a mass erase of the entire flash. Prior to initiating a mass erase, MSC_MASSLOCK must be unlocked by writing 0x631A to it. After a mass erase has been started, this register can be locked again to prevent runaway code from accidentally triggering a mass erase.

The regular flash page lock bits will not prevent a mass erase. To prevent software from initiating mass erases, use the mass erase lock bits in the mass erase lock word (MLW).

7.5.2 MSC_READCTRL - Read Control Register

Offset	Bit Position																															
0x004	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0			0	0		0x1	
Access																									RW			RW	RW		RW	
Name																									RAMCEN			AIDIS	IFCDIS		MODE	

Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*		
7	RAMCEN	0	RW	**RAM Cache Enable**
	Enable instruction caching for RAM in code-space.			
6:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*		
4	AIDIS	0	RW	**Automatic Invalidate Disable**
	When this bit is set the cache is not automatically invalidated when a write or page erase is performed.			
3	IFCDIS	0	RW	**Internal Flash Cache Disable**
	Disable instruction cache for internal flash memory.			
2:0	MODE	0x1	RW	**Read Mode**

If software wants to set a core clock frequency above 16 MHz, this register must be set to WS1 before the core clock is switched to the higher frequency. When changing to a lower frequency, this register can be set to WS0 after the frequency transition has been completed. After reset, the core clock is 14 MHz from the HFRCO but the MODE field of MSC_READCTRL register is set to WS1. This is because the HFRCO may produce a frequency above 16 MHz before it is calibrated. If the HFRCO is used as clock source, wait until the oscillator is stable on the new frequency to avoid unpredictable behavior.

Value	Mode	Description
0	WS0	Zero wait-states inserted in fetch or read transfers.
1	WS1	One wait-state inserted for each fetch or read transfer. This mode is required for a core frequency above 16 MHz.

7.5.3 MSC_WRITECTRL - Write Control Register

Offset	Bit Position																															
0x008	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																															0	0
Access																															RW	RW
Name																															IRQERASEABORT	WREN

Bit	Name	Reset	Access	Description
31:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*		
1	IRQERASEABORT	0	RW	**Abort Page Erase on Interrupt**
	When this bit is set to 1, any Cortex interrupt aborts any current page erase operation.			
0	WREN	0	RW	**Enable Write/Erase Controller**

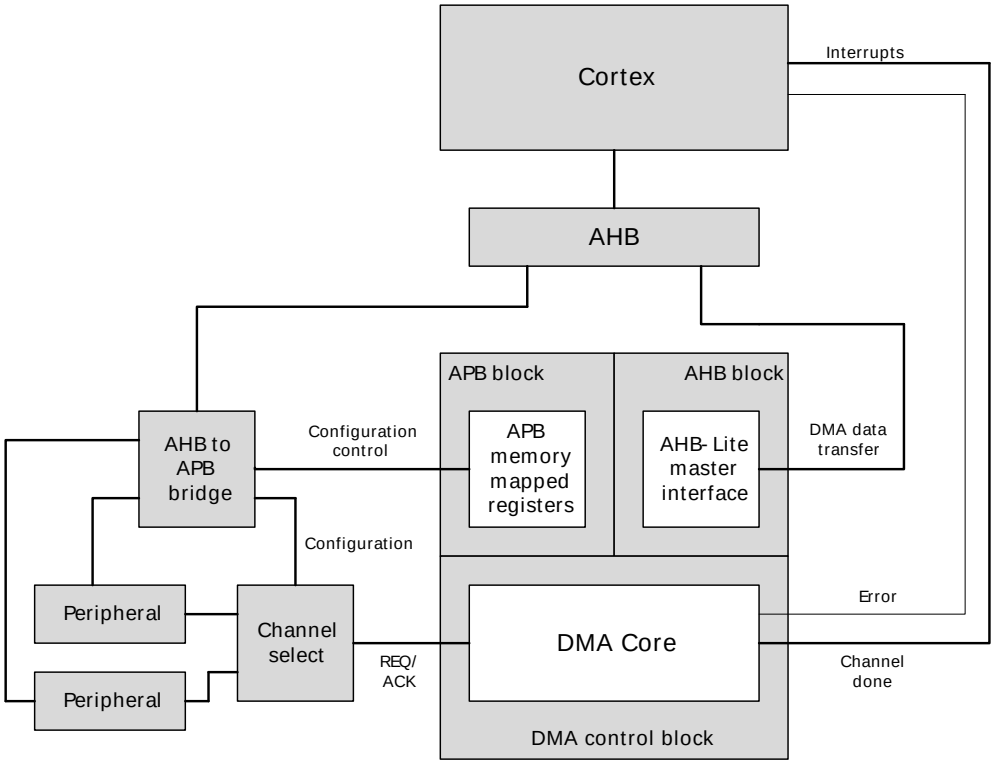
- Ping-pong (switching between the primary or alternate DMA descriptors, for continuous data flow to/from peripherals)
- Scatter-gather (using the primary descriptor to configure the alternate descriptor)

- Each channel has a programmable transfer length
- Channels 0 and 1 support looped transfers
- Channel 0 supports 2D copy
- A DMA channel can be triggered by any of several sources:
  - Communication modules (USART, LEUART)
  - Timers (TIMER)
  - Analog modules (ACMP, ADC)
  - Software
- Programmable mapping between channel number and peripherals - any DMA channel can be triggered by any of the available sources
- Interrupts upon transfer completion
- Data transfer to/from LEUART in EM2 is supported by the DMA, providing extremely low energy consumption while performing UART communications

8.3 Block Diagram

An overview of the DMA and the modules it interacts with is shown in Figure 8.1 (p. 44) .

Figure 8.1. DMA Block Diagram



The DMA Controller consists of four main parts:

- An APB block allowing software to configure the DMA controller
- An AHB block allowing the DMA to read and write the DMA descriptors and the source and destination data for the DMA transfers
- A DMA control block controlling the operation of the DMA, including request/acknowledge signals for the connected peripherals
- A channel select block routing the right peripheral request to each DMA channel

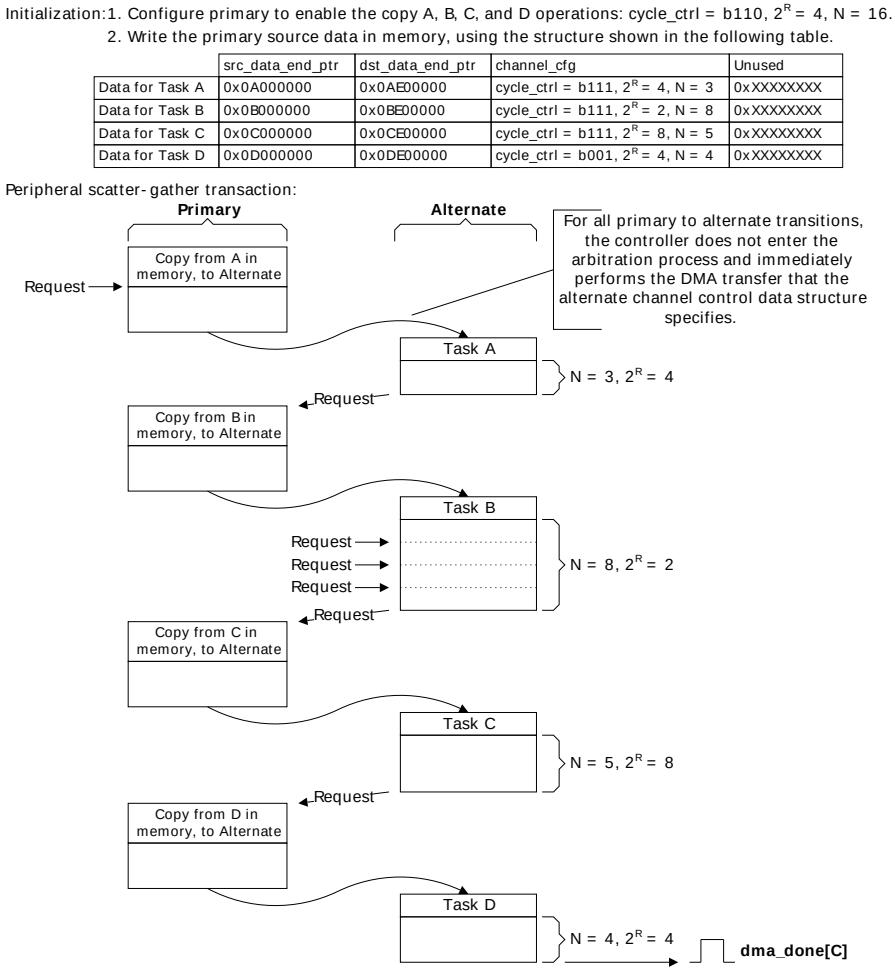
Bit	Field	Value	Description
[20:18]	src_prot_ctrl	-	Configures the state of HPROT when the controller reads the source data
[13:4]	n_minus_1	N [1]	Configures the controller to perform N DMA transfers, where N is a multiple of four
[3]	next_useburst	-	When set to 1, the controller sets the chnl_useburst_set [C] bit to 1 after the alternate transfer completes

[1]Because the R_power field is set to four, you must set N to be a multiple of four. The value given by N/4 is the number of times that you must configure the alternate data structure.

See Section 8.4.3.3 (p. 58) for more information.

Figure 8.5 (p. 54) shows a peripheral scatter-gather example.

Figure 8.5. Peripheral scatter-gather example



In Figure 8.5 (p. 54) :

Initialization

1. The host processor configures the primary data structure to operate in peripheral scatter-gather mode by setting cycle_ctrl to b110. Because a data structure for a single channel consists of four words then you must set 2^R to 4. In this example, there are four tasks and therefore N is set to 16.
2. The host processor writes the data structure for tasks A, B, C, and D to the memory locations that the primary src_data_end_ptr specifies.
3. The host processor enables the channel.

The peripheral scatter-gather transaction commences when the controller receives a request on `dma_req[]`. The transaction continues as follows:

Bit	Name	Reset	Access	Description
	Set when the DMA channel has completed its transfer. If the channel is disabled, the flag is set when there is a request for the channel.			

8.7.21 DMA_IFS - Interrupt Flag Set Register

Offset	Bit Position																															
0x1004	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0																												0	0	0	0
Access	W1																												W1	W1	W1	W1
Name	ERR																												CH3DONE	CH2DONE	CH1DONE	CH0DONE

Bit	Name	Reset	Access	Description
31	ERR	0	W1	**DMA Error Interrupt Flag Set**
	Set to 1 to set DMA error interrupt flag.			
30:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*		
3	CH3DONE	0	W1	**DMA Channel 3 Complete Interrupt Flag Set**
	Write to 1 to set the corresponding DMA channel complete interrupt flag.			
2	CH2DONE	0	W1	**DMA Channel 2 Complete Interrupt Flag Set**
	Write to 1 to set the corresponding DMA channel complete interrupt flag.			
1	CH1DONE	0	W1	**DMA Channel 1 Complete Interrupt Flag Set**
	Write to 1 to set the corresponding DMA channel complete interrupt flag.			
0	CH0DONE	0	W1	**DMA Channel 0 Complete Interrupt Flag Set**
	Write to 1 to set the corresponding DMA channel complete interrupt flag.			

8.7.22 DMA_IFC - Interrupt Flag Clear Register

Offset	Bit Position																															
0x1008	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0																												0	0	0	0
Access	W1																												W1	W1	W1	W1
Name	ERR																												CH3DONE	CH2DONE	CH1DONE	CH0DONE

Bit	Name	Reset	Access	Description
31	ERR	0	W1	**DMA Error Interrupt Flag Clear**
	Set to 1 to clear DMA error interrupt flag. Note that if an error happened, the Bus Error Clear Register must be used to clear the DMA.			
30:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*		
3	CH3DONE	0	W1	**DMA Channel 3 Complete Interrupt Flag Clear**
	Write to 1 to clear the corresponding DMA channel complete interrupt flag.			
2	CH2DONE	0	W1	**DMA Channel 2 Complete Interrupt Flag Clear**
	Write to 1 to clear the corresponding DMA channel complete interrupt flag.			
1	CH1DONE	0	W1	**DMA Channel 1 Complete Interrupt Flag Clear**

9.4 Register Map

The offset register address is relative to the registers base address.

Offset	Name	Type	Description
0x000	RMU_CTRL	RW	Control Register
0x004	RMU_RSTCAUSE	R	Reset Cause Register
0x008	RMU_CMD	W1	Command Register

9.5 Register Description

9.5.1 RMU_CTRL - Control Register

Offset	Bit Position																													
0x000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																																0
Access																																RW
Name																																LOCKUPRDIS

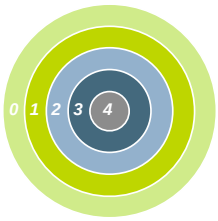
Bit	Name	Reset	Access	Description
31:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*		
0	LOCKUPRDIS	0	RW	**Lockup Reset Disable**
	Set this bit to disable the LOCKUP signal (from the Cortex) from resetting the device.			

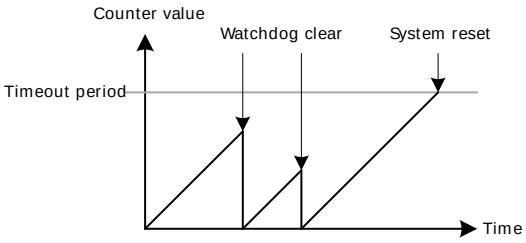
9.5.2 RMU_RSTCAUSE - Reset Cause Register

Offset	Bit Position																													
0x004	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																						0	0	0	0	0	0	0	0	0	0	0
Access																						R	R	R	R	R	R	R	R	R	R	R
Name																						BODAVDD1	BODAVDD0	EM4WURST	EM4RST	SYSREQRST	LOCKUPRST	WDOGRST	EXTRST	BODREGRST	BODUNREGRST	PORST

Bit	Name	Reset	Access	Description
31:11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*		
10	BODAVDD1	0	R	**AVDD1 Bod Reset**
	Set if analog power domain 1 brown out detector reset has been performed. Must be cleared by software. Please see Table 9.1 (p. 80) for details on how to interpret this bit.			
9	BODAVDD0	0	R	**AVDD0 Bod Reset**
	Set if analog power domain 0 brown out detector reset has been performed. Must be cleared by software. Please see Table 9.1 (p. 80) for details on how to interpret this bit.			
8	EM4WURST	0	R	**EM4 Wake-up Reset**
	Set if the system has been woken up from EM4 from a reset request from pin. Must be cleared by software. Please see Table 9.1 (p. 80) for details on how to interpret this bit.			

12 WDOG - Watchdog Timer





Quick Facts

What?

The WDOG (Watchdog Timer) resets the system in case of a fault condition, and can be enabled in all energy modes as long as the low frequency clock source is available.

Why?

If a software failure or external event renders the MCU unresponsive, a Watchdog timeout will reset the system to a known, safe state.

How?

An enabled Watchdog Timer implements a configurable timeout period. If the CPU fails to re-start the Watchdog Timer before it times out, a full system reset will be triggered. The Watchdog consumes insignificant power, and allows the device to remain safely in low energy modes for up to 256 seconds at a time.

12.1 Introduction

The purpose of the watchdog timer is to generate a reset in case of a system failure, to increase application reliability. The failure may e.g. be caused by an external event, such as an ESD pulse, or by a software failure.

12.2 Features

- Clock input from selectable oscillators
  - Internal 32.768 Hz RC oscillator
  - Internal 1 kHz RC oscillator
  - External 32.768 Hz XTAL oscillator
- Configurable timeout period from 9 to 256k watchdog clock cycles
- Individual selection to keep running or freeze when entering EM2 or EM3
- Selection to keep running or freeze when entering debug mode
- Selection to block the CPU from entering Energy Mode 4
- Selection to block the CMU from disabling the selected watchdog clock

12.3 Functional Description

The watchdog is enabled by setting the EN bit in WDOG_CTRL. When enabled, the watchdog counts up to the period value configured through the PERSEL field in WDOG_CTRL. If the watchdog timer is not cleared to 0 (by writing a 1 to the CLEAR bit in WDOG_CMD) before the period is reached, the chip is reset. If a timely clear command is issued, the timer starts counting up from 0 again. The watchdog can optionally be locked by writing the LOCK bit in WDOG_CTRL. Once locked, it cannot be disabled or reconfigured by software.

The watchdog counter is reset when EN is reset.

14.5.3 I2Cn_STATE - State Register

Offset	Bit Position																															
0x008	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0x0			0	0	0	0	1
Access																									R			R	R	R	R	R
Name																									STATE			BUSHOLD	NACKED	TRANSMITTER	MASTER	BUSY

Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*		
7:5	STATE	0x0	R	**Transmission State**

The state of any current transmission. Cleared if the I^{2}C module is idle.

Value	Mode	Description
0	IDLE	No transmission is being performed.
1	WAIT	Waiting for idle. Will send a start condition as soon as the bus is idle.
2	START	Start transmitted or received
3	ADDR	Address transmitted or received
4	ADDRACK	Address ack/nack transmitted or received
5	DATA	Data transmitted or received
6	DATAACK	Data ack/nack transmitted or received

Bit	Name	Reset	Access	Description
4	BUSHOLD	0	R	**Bus Held**

Set if the bus is currently being held by this I^{2}C module.

Bit	Name	Reset	Access	Description
3	NACKED	0	R	**Nack Received**

Set if a NACK was received and STATE is ADDRACK or DATAACK.

Bit	Name	Reset	Access	Description
2	TRANSMITTER	0	R	**Transmitter**

Set when operating as a master transmitter or a slave transmitter. When cleared, the system may be operating as a master receiver, a slave receiver or the current mode is not known.

Bit	Name	Reset	Access	Description
1	MASTER	0	R	**Master**

Set when operating as an I^{2}C master. When cleared, the system may be operating as an I^{2}C slave.

Bit	Name	Reset	Access	Description
0	BUSY	1	R	**Bus Busy**

Set when the bus is busy. Whether the I^{2}C module is in control of the bus or not has no effect on the value of this bit. When the MCU comes out of reset, the state of the bus is not known, and thus BUSY is set. Use the ABORT command or a bus idle timeout to force the I^{2}C module out of the BUSY state.

14.5.4 I2Cn_STATUS - Status Register

Offset	Bit Position																															
0x00C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																								0	1	0	0	0	0	0	0	0
Access																								R	R	R	R	R	R	R	R	R
Name																								RXDATAV	TXBL	TXC	PABORT	PCONT	PNACK	PACK	PSTOP	PSTART

Bit	Name	Reset	Access	Description
31:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*		

Bit	Name	Reset	Access	Description
	Write to 1 to clear the SSTOP interrupt flag.			
15	CLTO	0	W1	**Clear Clock Low Interrupt Flag**
	Write to 1 to clear the CLTO interrupt flag.			
14	BITO	0	W1	**Clear Bus Idle Timeout Interrupt Flag**
	Write to 1 to clear the BITO interrupt flag.			
13	RXUF	0	W1	**Clear Receive Buffer Underflow Interrupt Flag**
	Write to 1 to clear the RXUF interrupt flag.			
12	TXOF	0	W1	**Clear Transmit Buffer Overflow Interrupt Flag**
	Write to 1 to clear the TXOF interrupt flag.			
11	BUSHOLD	0	W1	**Clear Bus Held Interrupt Flag**
	Write to 1 to clear the BUSHOLD interrupt flag.			
10	BUSERR	0	W1	**Clear Bus Error Interrupt Flag**
	Write to 1 to clear the BUSERR interrupt flag.			
9	ARBLOST	0	W1	**Clear Arbitration Lost Interrupt Flag**
	Write to 1 to clear the ARBLOST interrupt flag.			
8	MSTOP	0	W1	**Clear MSTOP Interrupt Flag**
	Write to 1 to clear the MSTOP interrupt flag.			
7	NACK	0	W1	**Clear Not Acknowledge Received Interrupt Flag**
	Write to 1 to clear the NACK interrupt flag.			
6	ACK	0	W1	**Clear Acknowledge Received Interrupt Flag**
	Write to 1 to clear the ACK interrupt flag.			
5:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*		
3	TXC	0	W1	**Clear Transfer Completed Interrupt Flag**
	Write to 1 to clear the TXC interrupt flag.			
2	ADDR	0	W1	**Clear Address Interrupt Flag**
	Write to 1 to clear the ADDR interrupt flag.			
1	RSTART	0	W1	**Clear Repeated START Interrupt Flag**
	Write to 1 to clear the RSTART interrupt flag.			
0	START	0	W1	**Clear START Interrupt Flag**
	Write to 1 to clear the START interrupt flag.			

14.5.14 I2Cn_IEN - Interrupt Enable Register

Offset	Bit Position																															
0x034	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Access																RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Name																SSTOP	CLTO	BITO	RXUF	TXOF	BUSHOLD	BUSERR	ARBLOST	MSTOP	NACK	ACK	RXDATAV	TXBL	TXC	ADDR	RSTART	START

Bit	Name	Reset	Access	Description
31:17	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*		
16	SSTOP	0	RW	**SSTOP Interrupt Enable**
	Enable interrupt on SSTOP.			

Bit	Name	Reset	Access	Description
10:8	LOCATION	0x0	RW	**I/O Location**

Decides the location of the I^2C I/O pins.

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 3
4	LOC4	Location 4
5	LOC5	Location 5
6	LOC6	Location 6

7:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*		
1	SCLPEN	0	RW	**SCL Pin Enable**

When set, the SCL pin of the I^2C is enabled.

0	SDAPEN	0	RW	**SDA Pin Enable**

When set, the SDA pin of the I^2C is enabled.

15.5.10 USARTn_RXDOUBLE - RX FIFO Double Data Register

Offset	Bit Position																													
0x024	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0x00								0x00							
Access																	R								R							
Name																	RXDATA1								RXDATA0							

Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*		
15:8	RXDATA1	0x00	R	**RX Data 1**
	Second frame read from buffer.			
7:0	RXDATA0	0x00	R	**RX Data 0**
	First frame read from buffer.			

15.5.11 USARTn_RXDATAXP - RX Buffer Data Extended Peek Register

Offset	Bit Position																													
0x028	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0	0						0x000								
Access																	R	R						R								
Name																	FERRP	PERRP						RXDATAP								

Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*		
15	FERRP	0	R	**Data Framing Error Peek**
	Set if data in buffer has a framing error. Can be the result of a break condition.			
14	PERRP	0	R	**Data Parity Error Peek**
	Set if data in buffer has a parity error (asynchronous mode only).			
13:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*		
8:0	RXDATAP	0x000	R	**RX Data Peek**
	Use this register to access data read from the USART.			

16.3.4.3 Jitter in Transmitted Data

Internally the LEUART module uses only the positive edges of the 32.768 kHz clock (LFBCLK) for transmission and reception. Transmitted data will thus have jitter equal to the difference between the optimal data set-up location and the closest positive edge on the 32.768 kHz clock. The jitter in on the location data is set up by the transmitter will thus be no more than half a clock period according to the optimal set-up location. The jitter in the period of a single baud output by the transmitter will never be more than one clock period.

16.3.5 Data Reception

Data reception is enabled by setting RXEN in LEUARTn_CMD. When the receiver is enabled, it actively samples the input looking for a transition from high to low indicating the start baud of a new frame. When a start baud is found, reception of the new frame begins if the receive shift register is empty and ready for new data. When the frame has been received, it is pushed into the receive buffer, making the shift register ready for another frame of data, and the receiver starts looking for another start baud. If the receive buffer is full, the received frame remains in the shift register until more space in the receive buffer is available.

If an incoming frame is detected while both the receive buffer and the receive shift register are full, the data in the receive shift register is overwritten, and the RXOF interrupt flag in LEUARTn_IF is set to indicate the buffer overflow.

The receiver can be disabled by setting the command bit RXDIS in LEUARTn_CMD. Any frame currently being received when the receiver is disabled is discarded. Whether or not the receiver is enabled at a given time can be read out from RXENS in LEUARTn_STATUS.

16.3.5.1 Receive Buffer Operation

When data becomes available in the receive buffer, the RXDATAV flag in LEUARTn_STATUS and the RXDATAV interrupt flag in LEUARTn_IF are set. Both the RXDATAV status flag and the RXDATAV interrupt flag are cleared by hardware when data is no longer available, i.e. when data has been read out of the buffer.

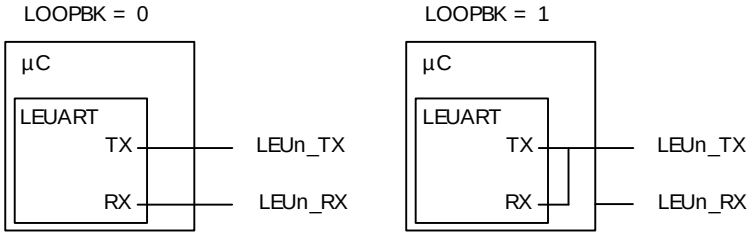
Data can be read from receive buffer using either LEUARTn_RXDATA or LEUARTn_RXDATAX. LEUARTn_RXDATA gives access to the 8 least significant bits of the received frame, while LEUARTn_RXDATAX must be used to get access to the 9th, most significant bit. The latter register also contains status information regarding the frame.

When a frame is read from the receive buffer using LEUARTn_RXDATA or LEUARTn_RXDATAX, the frame is removed from the buffer, making room for a new one. If an attempt is done to read more frames from the buffer than what is available, the RXUF interrupt flag in LEUARTn_IF is set to signal the underflow, and the data read from the buffer is undefined.

Frames can also be read from the receive buffer without removing the data by using LEUARTn_RXDATAXP, which gives access to the frame in the buffer including control bits. Data read from this register when the receive buffer is empty is undefined. No underflow interrupt is generated by a read using LEUARTn_RXDATAXP, i.e. the RXUF interrupt flag is never set as a result of reading from LEUARTn_RXDATAXP.

An overview of the operation of the receiver is shown in Figure 16.4 (p. 221) .

Figure 16.5. LEUART Local Loopback



16.3.7 Half Duplex Communication

When doing full duplex communication, two data links are provided, making it possible for data to be sent and received at the same time. In half duplex mode, data is only sent in one direction at a time. There are several possible half duplex setups, as described in the following sections.

16.3.7.1 Single Data-link

In this setup, the LEUART both receives and transmits data on the same pin. This is enabled by setting LOOPBK in LEUARTn_CTRL, which connects the receiver to the transmitter output. Because they are both connected to the same line, it is important that the LEUART transmitter does not drive the line when receiving data, as this would corrupt the data on the line.

When communicating over a single data-link, the transmitter must thus be tristated whenever not transmitting data. If AUTOTRI in LEUARTn_CTRL is set, the LEUART automatically tristates LEUn_TX whenever the transmitter is inactive. It is then the responsibility of the software protocol to make sure the transmitter is not transmitting data whenever incoming data is expected.

The transmitter can also be tristated from software by configuring the GPIO pin as an input and disabling the LEUART output on LEUn_TX.

Note

Another way to tristate the transmitter is to enable wired-and or wired-or mode in GPIO. For wired-and mode, outputting a 1 will be the same as tristating the output, and for wired-or mode, outputting a 0 will be the same as tristating the output. This can only be done on buses with a pull-up or pull-down resistor respectively.

16.3.7.2 Single Data-link with External Driver

Some communication schemes, such as RS-485 rely on an external driver. Here, the driver has an extra input which enables it, and instead of Tristating the transmitter when receiving data, the external driver must be disabled. The USART has hardware support for automatically turning the driver on and off. When using the LEUART in such a setup, the driver must be controlled by a GPIO. Figure 16.6 (p. 224) shows an example configuration using an external driver.

Figure 16.6. LEUART Half Duplex Communication with External Driver

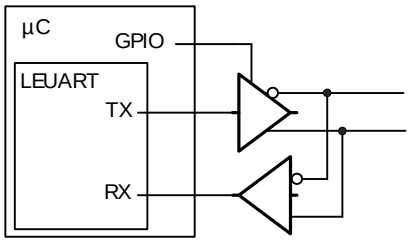
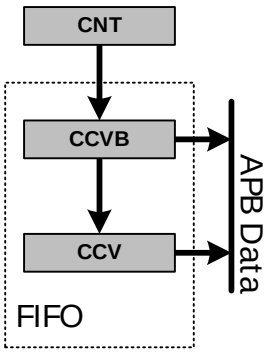


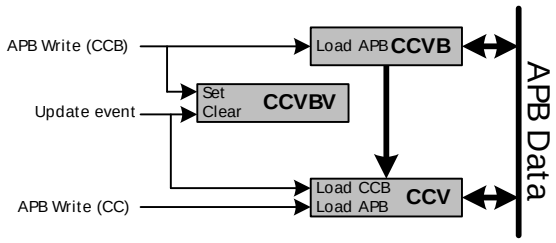
Figure 17.11. TIMER Input Capture Buffer Functionality



17.3.2.2.2 Compare and PWM Mode

When running in Output Compare or PWM mode, the value in TIMERn_CCx_CCV will be compared against the count value. In Compare mode the output can be configured to toggle, clear or set on compare match, overflow and underflow through the CMOA, COFOA and CUFOA fields in TIMERn_CCx_CTRL. TIMERn_CCx_CCV can be accessed directly or through the buffer register TIMERn_CCx_CCVB, see Figure 17.12 (p. 249) . When writing to the buffer register, the value in TIMERn_CCx_CCVB will be written to TIMERn_CCx_CCV on the next update event. This functionality ensures glitch free PWM outputs. The CCVBV flag in TIMERn_STATUS indicates whether the TIMERn_CCx_CCVB register contains data that have not yet been written to the TIMERn_CCx_CCV register. Note that when writing 0 to TIMERn_CCx_CCVB the CCV value is updated when the timer counts from 0 to 1. Thus, the compare match for the next period will not happen until the timer reaches 0 again on the way down.

Figure 17.12. TIMER Output Compare/PWM Buffer Functionality



17.3.2.3 Input Capture

In Input Capture Mode, the counter value (TIMERn_CNT) can be captured in the Compare/Capture Register (TIMERn_CCx_CCV), see Figure 17.13 (p. 250) . In this mode, TIMERn_CCx_CCV is read-only. Together with the Compare/Capture Buffer Register (TIMERn_CCx_CCVB) the TIMERn_CCx_CCV form a double-buffered capture registers allowing two subsequent capture events to take place before a read-out is required. The CCPOL bits in TIMERn_STATUS indicate the polarity the edge that triggered the capture in TIMERn_CCx_CCV.

20.5.8 ACMPn_ROUTE - I/O Routing Register

Offset	Bit Position																													
0x01C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																							0x0									0
Access																							RW									RW
Name																							LOCATION									ACMPPEN

Bit	Name	Reset	Access	Description
31:11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*		
10:8	LOCATION	0x0	RW	**I/O Location**
	Decides the location of the ACMP I/O pin.			

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2

7:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*		
0	ACMPPEN	0	RW	**ACMP Output Pin Enable**
	Enable/disable analog comparator output to pin.			

Bit	Name	Reset	Access	Description
23:18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*		
17	SCANDV	0	R	**Scan Data Valid**
	Scan conversion data is valid.			
16	SINGLEDV	0	R	**Single Sample Data Valid**
	Single conversion data is valid.			
15:13	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*		
12	WARM	0	R	**ADC Warmed Up**
	ADC is warmed up.			
11:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*		
9	SCANREFWARM	0	R	**Scan Reference Warmed Up**
	Reference selected for scan mode is warmed up.			
8	SINGLEREFWARM	0	R	**Single Reference Warmed Up**
	Reference selected for single mode is warmed up.			
7:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*		
1	SCANACT	0	R	**Scan Conversion Active**
	Scan sequence is active or has pending conversions.			
0	SINGLEACT	0	R	**Single Conversion Active**
	Single conversion is active or has pending conversions.			

22.5.4 ADCn_SINGLECTRL - Single Sample Control Register

Offset	Bit Position																													
0x00C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset			0x0					0	0x0					0x0							0x0						0x0			0	0	0
Access			RW					RW	RW					RW							RW						RW			RW	RW	RW
Name			PRSSEL					PRSEN	AT					REF							INPUTSEL						RES			ADJ	DIFF	REP

Bit	Name	Reset	Access	Description
31:30	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*		
29:28	PRSSEL	0x0	RW	**Single Sample PRS Trigger Select**
	Select PRS trigger for single sample.			

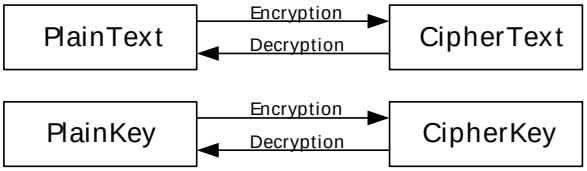
Value	Mode	Description
0	PRSCH0	PRS ch 0 triggers single sample
1	PRSCH1	PRS ch 1 triggers single sample
2	PRSCH2	PRS ch 2 triggers single sample
3	PRSCH3	PRS ch 3 triggers single sample

27:25	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*		
24	PRSEN	0	RW	**Single Sample PRS Trigger Enable**
	Enabled/disable PRS trigger of single sample.			

Value	Description
0	Single sample is not triggered by PRS input
1	Single sample is triggered by PRS input selected by PRSSEL

23:20	AT	0x0	RW	**Single Sample Acquisition Time**

Figure 24.1. AES Key and Data Definitions



24.3.1 Encryption/Decryption

The AES module can be set to encrypt or decrypt by clearing/setting the DECRYPT bit in AES_CTRL. The AES_CTRL register should not be altered while AES is running, as this may lead to unpredictable behaviour.

An AES encryption/decryption can be started in the following ways:

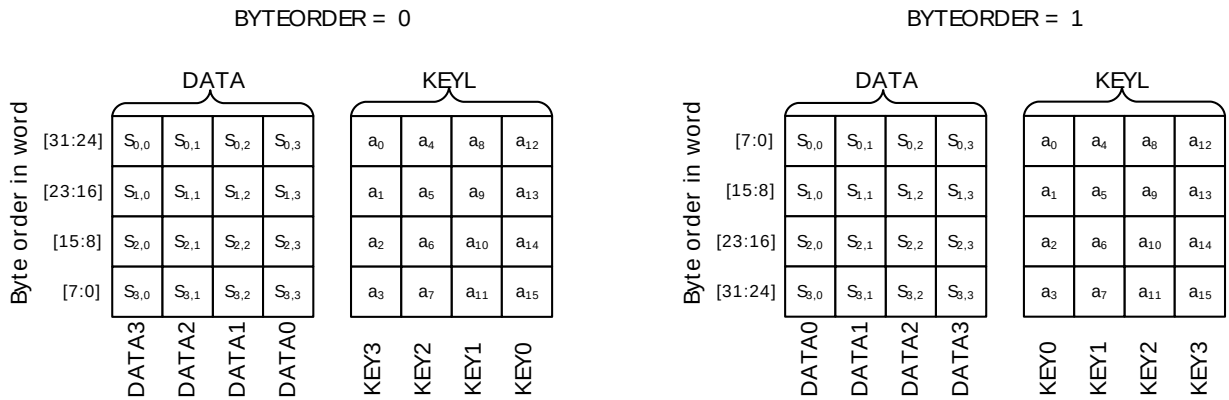
- Writing a 1 to the START bit in AES_CMD
- Writing 4 times 32 bits to AES_DATA when the DATASTART control bit is set
- Writing 4 times 32 bits to AES_XORDATA when the XORSTART control bit is set

An AES encryption/decryption can be stopped by writing a 1 to the STOP bit in AES_CMD. The RUNNING bit in AES_STATUS indicates that an AES encryption/decryption is ongoing.

24.3.2 Data and Key Access

The AES module contains a 128-bit DATA (State) register and a 128-bit KEY register defined as DATA3-DATA0 and KEY3-KEY0 (KEYL). The AES module has configurable byte ordering which is configured in BYTEORDER in AES_CTRL. Figure 24.2 (p. 346) illustrates how data written to the AES registers is mapped to the key and state defined in the Advanced Encryption Standard (FIPS-197). AES encryption/decryption takes two extra cycles when BYTEORDER is set. BYTEORDER has to be set prior to loading the data and key registers.

Figure 24.2. AES Data and Key Orientation as Defined in the Advanced Encryption Standard



The registers DATA3-DATA0, are not memory mapped directly, but can be written/read by accessing AES_DATA or AES_XORDATA. The same applies for the key registers, KEY3-KEY0 which are accessed through AES_KEYLn (n=A, B, C or D). Writing DATA3-DATA0 is then done through 4

25.5.9 GPIO_Px_PINLOCKN - Port Unlocked Pins Register

Offset	Bit Position																													
0x020	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0xFFFF														
Access																	RW														
Name																	PINLOCKN														

Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*		
15:0	PINLOCKN	0xFFFF	RW	**Unlocked Pins**
	Shows unlocked pins in the port. To lock pin n, clear bit n. The pin is then locked until reset.			

25.5.10 GPIO_EXTIPSELL - External Interrupt Port Select Low Register

Offset	Bit Position																													
0x100	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset		0x0				0x0				0x0				0x0				0x0				0x0				0x0				0x0		
Access		RW				RW				RW				RW				RW				RW				RW				RW		
Name		EXTIPSEL7				EXTIPSEL6				EXTIPSEL5				EXTIPSEL4				EXTIPSEL3				EXTIPSEL2				EXTIPSEL1				EXTIPSEL0		

Bit	Name	Reset	Access	Description
31	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*		
30:28	EXTIPSEL7	0x0	RW	**External Interrupt 7 Port Select**
	Select input port for external interrupt 7.			

Value	Mode	Description
0	PORTA	Port A pin 7 selected for external interrupt 7
1	PORTB	Port B pin 7 selected for external interrupt 7
2	PORTC	Port C pin 7 selected for external interrupt 7
3	PORTD	Port D pin 7 selected for external interrupt 7
4	PORTE	Port E pin 7 selected for external interrupt 7
5	PORTF	Port F pin 7 selected for external interrupt 7

27	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*		
26:24	EXTIPSEL6	0x0	RW	**External Interrupt 6 Port Select**
	Select input port for external interrupt 6.			

Value	Mode	Description
0	PORTA	Port A pin 6 selected for external interrupt 6
1	PORTB	Port B pin 6 selected for external interrupt 6
2	PORTC	Port C pin 6 selected for external interrupt 6
3	PORTD	Port D pin 6 selected for external interrupt 6
4	PORTE	Port E pin 6 selected for external interrupt 6
5	PORTF	Port F pin 6 selected for external interrupt 6

Bit	Name	Reset	Access	Description
23	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*		
22:20	EXTIPSEL5	0x0	RW	**External Interrupt 5 Port Select**

Select input port for external interrupt 5.

Value	Mode	Description
0	PORTA	Port A pin 5 selected for external interrupt 5
1	PORTB	Port B pin 5 selected for external interrupt 5
2	PORTC	Port C pin 5 selected for external interrupt 5
3	PORTD	Port D pin 5 selected for external interrupt 5
4	PORTE	Port E pin 5 selected for external interrupt 5
5	PORTF	Port F pin 5 selected for external interrupt 5

19	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*		
18:16	EXTIPSEL4	0x0	RW	**External Interrupt 4 Port Select**

Select input port for external interrupt 4.

Value	Mode	Description
0	PORTA	Port A pin 4 selected for external interrupt 4
1	PORTB	Port B pin 4 selected for external interrupt 4
2	PORTC	Port C pin 4 selected for external interrupt 4
3	PORTD	Port D pin 4 selected for external interrupt 4
4	PORTE	Port E pin 4 selected for external interrupt 4
5	PORTF	Port F pin 4 selected for external interrupt 4

15	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*		
14:12	EXTIPSEL3	0x0	RW	**External Interrupt 3 Port Select**

Select input port for external interrupt 3.

Value	Mode	Description
0	PORTA	Port A pin 3 selected for external interrupt 3
1	PORTB	Port B pin 3 selected for external interrupt 3
2	PORTC	Port C pin 3 selected for external interrupt 3
3	PORTD	Port D pin 3 selected for external interrupt 3
4	PORTE	Port E pin 3 selected for external interrupt 3
5	PORTF	Port F pin 3 selected for external interrupt 3

11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*		
10:8	EXTIPSEL2	0x0	RW	**External Interrupt 2 Port Select**

Select input port for external interrupt 2.

Value	Mode	Description
0	PORTA	Port A pin 2 selected for external interrupt 2
1	PORTB	Port B pin 2 selected for external interrupt 2
2	PORTC	Port C pin 2 selected for external interrupt 2
3	PORTD	Port D pin 2 selected for external interrupt 2
4	PORTE	Port E pin 2 selected for external interrupt 2
5	PORTF	Port F pin 2 selected for external interrupt 2

7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in Section 2.1 (p. 3)*		
6:4	EXTIPSEL1	0x0	RW	**External Interrupt 1 Port Select**

Select input port for external interrupt 1.

Value	Mode	Description
0	PORTA	Port A pin 1 selected for external interrupt 1
1	PORTB	Port B pin 1 selected for external interrupt 1
2	PORTC	Port C pin 1 selected for external interrupt 1
3	PORTD	Port D pin 1 selected for external interrupt 1
4	PORTE	Port E pin 1 selected for external interrupt 1