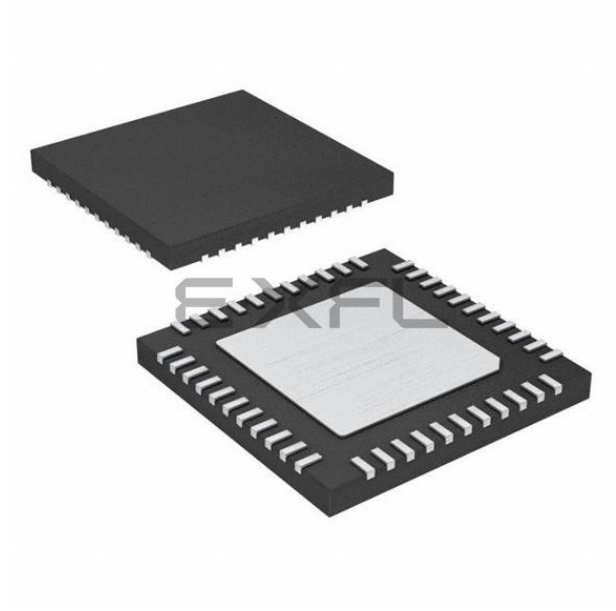


Welcome to E-XFL.COM

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	Active
Core Processor	AVR
Core Size	8/16-Bit
Speed	32MHz
Connectivity	I²C, IrDA, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	34
Program Memory Size	64KB (32K x 16)
Program Memory Type	FLASH
EEPROM Size	2K x 8
RAM Size	4K x 8
Voltage - Supply (Vcc/Vdd)	1.6V ~ 3.6V
Data Converters	A/D 12x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	44-VFQFN Exposed Pad
Supplier Device Package	44-VQFN (7x7)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atxmega64d4-mn

3.10 RAMP and Extended Indirect Registers

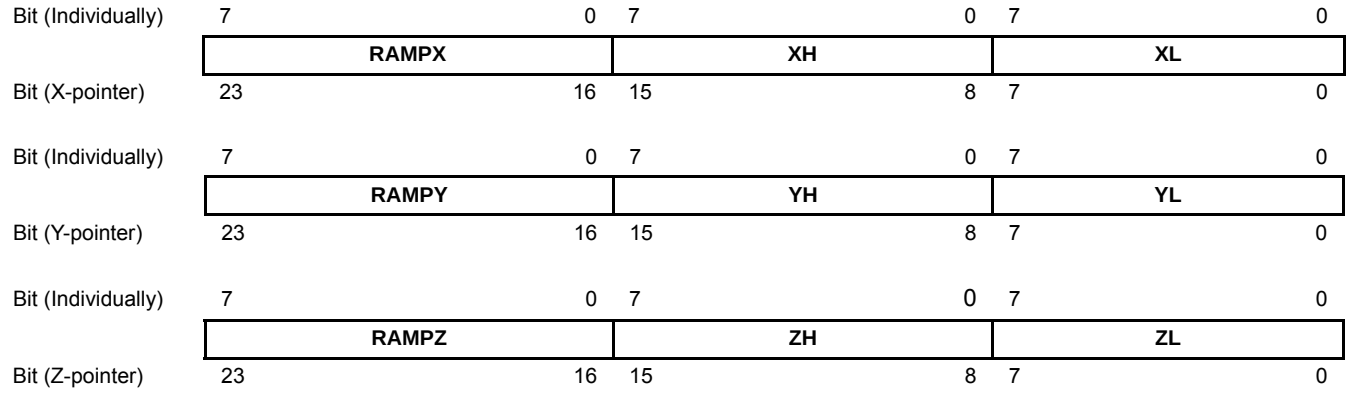
In order to access program memory or data memory above 64KB, the address pointer must be larger than 16 bits. This is done by concatenating one register to one of the X-, Y-, or Z-registers. This register then holds the most-significant byte (MSB) in a 24-bit address or address pointer.

These registers are available only on devices with external bus interface and/or more than 64KB of program or data memory space. For these devices, only the number of bits required to address the whole program and data memory space in the device is implemented in the registers.

3.10.1 RAMPX, RAMPY, and RAMPZ Registers

The RAMPX, RAMPY, and RAMPZ registers are concatenated with the X-, Y-, and Z-registers, respectively, to enable indirect addressing of the whole data memory space above 64KB and up to 16MB.

Figure 3-6. The Combined RAMPX + X, RAMPY + Y, and RAMPZ + Z Registers

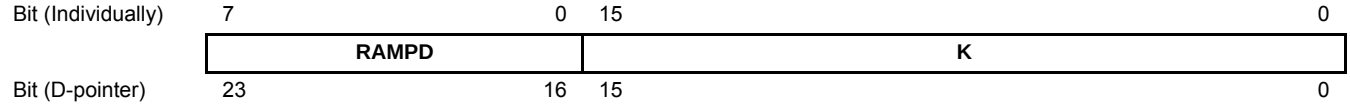


When reading (ELPM) and writing (SPM) program memory locations above the first 128KB of the program memory, RAMPZ is concatenated with the Z-register to form the 24-bit address. LPM is not affected by the RAMPZ setting.

3.10.2 RAMPD Register

This register is concatenated with the operand to enable direct addressing of the whole data memory space above 64KB. Together, RAMPD and the operand will form a 24-bit address.

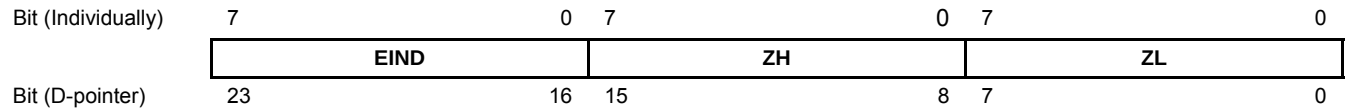
Figure 3-7. The Combined RAMPD + K Register



3.10.3 EIND - Extended Indirect Register

EIND is concatenated with the Z-register to enable indirect jump and call to locations above the first 128KB (64K words) of the program memory.

Figure 3-8. The combined EIND + Z Register



3.11 Accessing 16-bit Registers

The AVR data bus is 8 bits wide, and so accessing 16-bit registers requires atomic operations. These registers must be byte-accessed using two read or write operations. 16-bit registers are connected to the 8-bit bus and a temporary register using a 16-bit bus.

For a write operation, the low byte of the 16-bit register must be written before the high byte. The low byte is then written into the temporary register. When the high byte of the 16-bit register is written, the temporary register is copied into the low byte of the 16-bit register in the same clock cycle.

For a read operation, the low byte of the 16-bit register must be read before the high byte. When the low byte register is read by the CPU, the high byte of the 16-bit register is copied into the temporary register in the same clock cycle as the low byte is read. When the high byte is read, it is then read from the temporary register.

This ensures that the low and high bytes of 16-bit registers are always accessed simultaneously when reading or writing the register.

Interrupts can corrupt the timed sequence if an interrupt is triggered and accesses the same 16-bit register during an atomic 16-bit read/write operation. To prevent this, interrupts can be disabled when writing or reading 16-bit registers.

The temporary registers can also be read and written directly from user software.

3.11.1 Accessing 24- and 32-bit Registers

For 24- and 32-bit registers, the read and write access is done in the same way as described for 16-bit registers, except there are two temporary registers for 24-bit registers and three for 32-bit registers. The least-significant byte must be written first when doing a write, and read first when doing a read.

3.12 Configuration Change Protection

System critical I/O register settings are protected from accidental modification. The SPM instruction is protected from accidental execution, and the LPM instruction is protected when reading the fuses and signature row. This is handled globally by the configuration change protection (CCP) register. Changes to the protected I/O registers or bits, or execution of protected instructions, are only possible after the CPU writes a signature to the CCP register. The different signatures are described in the register description.

There are two modes of operation: one for protected I/O registers, and one for the protected instructions, SPM/LPM.

3.12.1 Sequence for Write Operation to Protected I/O Registers

1. The application code writes the signature that enable change of protected I/O registers to the CCP register.
2. Within four instruction cycles, the application code must write the appropriate data to the protected register. Most protected registers also contain a write enable/change enable bit. This bit must be written to one in the same operation as the data are written. The protected change is immediately disabled if the CPU performs write operations to the I/O register or data memory or if the SPM, LPM, or SLEEP instruction is executed.

3.12.2 Sequence for Execution of Protected SPM/LPM

1. The application code writes the signature for the execution of protected SPM/LPM to the CCP register.
2. Within four instruction cycles, the application code must execute the appropriate instruction. The protected change is immediately disabled if the CPU performs write operations to the data memory or if the SLEEP instruction is executed.

Once the correct signature is written by the CPU, interrupts will be ignored for the duration of the configuration change enable period. Any interrupt request (including non-maskable interrupts) during the CCP period will set the corresponding interrupt flag as normal, and the request is kept pending. After the CCP period is completed, any pending interrupts are executed according to their level and priority.

4.12 Register Description – NVM Controller

4.12.1 ADDR0 – Address Register 0

The ADDR0, ADDR1, and ADDR2 registers represent the 24-bit value, ADDR. This is used for addressing all NVM sections for read, write, and CRC operations.

Bit	7	6	5	4	3	2	1	0
+0x00	ADDR[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	1	1	1	1	1	1	1	1

- **Bit 7:0 – ADDR[7:0]: Address Byte 0**

This register gives the address low byte when accessing NVM locations.

4.12.2 ADDR1 – Address Register 1

Bit	7	6	5	4	3	2	1	0
+0x01	ADDR[15:8]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:0 – ADDR[15:8]: Address Byte 1**

This register gives the address high byte when accessing NVM locations.

4.12.3 ADDR2 – Address Register 2

Bit	7	6	5	4	3	2	1	0
+0x02	ADDR[23:16]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:0 – ADDR[23:16]: Address Byte 2**

This register gives the address extended byte when accessing NVM locations.

4.12.4 DATA0 – Data Register 0

The DATA0, DATA1, and DATA registers represent the 24-bit value, DATA. This holds data during NVM read, write, and CRC access.

Bit	7	6	5	4	3	2	1	0
+0x04	DATA[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:0 – DATA[7:0]: Data Byte 0**

This register gives the data value byte 0 when accessing NVM locations.

4.13 Register Descriptions – Fuses and Lock Bits

4.13.1 FUSEBYTE1 – Fuse Byte 1

Bit	7	6	5	4	3	2	1	0
+0x01	WDWPER[3:0]				WDPER[3:0]			
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:4 – WDWPER[3:0]: Watchdog Window Timeout Period**

These fuse bits are used to set initial value of the closed window for the Watchdog Timer in Window Mode. During reset these fuse bits are automatically written to the WPER bits Watchdog Window Mode Control Register. Refer to “WINCTRL – Window Mode Control Register” on page 92 in “WDT – Watchdog Timer” on page 89 for details.

- **Bit 3:0 – WDPER[3:0]: Watchdog Timeout Period**

These fuse bits are used to set the initial value of the watchdog timeout period. During reset these fuse bits are automatically written to the PER bits in the watchdog control register. Refer to “CTRL – Control Register” on page 91 in “WDT – Watchdog Timer” on page 89 for details.

4.13.2 FUSEBYTE2 – Fuse Byte 2

Bit	7	6	5	4	3	2	1	0
+0x02	–	BOOTRST	TOSCSEL	–	–	–	BODPD[1:0]	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	1	1	1	1	1	1	1	1

- **Bit 7 – Reserved**

This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to one when this register is written.

- **Bit 6 – BOOTRST: Boot Loader Section Reset Vector**

This fuse can be programmed so the reset vector is pointing to the first address in the boot loader flash section. The device will then start executing from the boot loader flash section after reset.

Table 4-1. Boot Reset Fuse

BOOSTRST	Reset address
0	Reset vector = Boot loader reset
1	Reset vector = Application reset (address 0x0000)

- **Bit 5 – TOSCSEL: 32.768kHz Timer Oscillator Pin Selection**

This fuse is used to select the pin location for the 32.768kHz timer oscillator (TOSC). This fuse is available only on devices where XTAL and TOSC pins by default are shared.

Table 4-2. TOSCSEL Fuse

TOSCSEL	Group configuration	Description
0	ALTERNATE(1)	TOSC1/2 on separate pins
1	XTAL	TOSC1/2 shared with XTAL

Note: 1. See the device datasheet for alternate TOSC position.

5.9 Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CH0MUX	CH0MUX[7:0]								51
+0x01	CH1MUX	CH1MUX[7:0]								51
+0x02	CH2MUX	CH2MUX[7:0]								51
+0x03	CH3MUX	CH3MUX[7:0]								51
+0x04	Reserved	–	–	–	–	–	–	–	–	
+0x05	Reserved	–	–	–	–	–	–	–	–	
+0x06	Reserved	–	–	–	–	–	–	–	–	
+0x07	Reserved	–	–	–	–	–	–	–	–	
+0x08	CH0CTRL	–	QDIRM[1:0]		QDIEN	QDEN	DIGFILT[2:0]			53
+0x09	CH1CTRL	–	–	–	–	–	DIGFILT[2:0]			53
+0x0A	CH2CTRL	–	QDIRM[1:0]		QDIEN	QDEN	DIGFILT[2:0]			53
+0x0B	CH3CTRL	–	–	–	–	–	DIGFILT[2:0]			53
+0x0C	Reserved	–	–	–	–	–	–	–	–	
+0x0D	Reserved	–	–	–	–	–	–	–	–	
+0x0E	Reserved	–	–	–	–	–	–	–	–	
+0x0F	Reserved	–	–	–	–	–	–	–	–	
+0x10	STROBE	STROBE[7:0]								54
+0x11	DATA	DATA[7:0]								54

6.9.3 LOCK – Lock Register

Bit	7	6	5	4	3	2	1	0
+0x02	–	–	–	–	–	–	–	LOCK
Read/Write	R	R	R	R	R	R	R	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:1 – Reserved**

These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.

- **Bit 0 – LOCK: Clock System Lock**

When this bit is written to one, the CTRL and PSCTRL registers cannot be changed, and the system clock selection and prescaler settings are protected against all further updates until after the next reset. This bit is protected by the configuration change protection mechanism. For details, refer to “Configuration Change Protection” on page 13.

The LOCK bit can be cleared only by a reset.

6.9.4 RTCCTRL – RTC Control Register

Bit	7	6	5	4	3	2	1	0
+0x03	–	–	–	–	RTCSRC[2:0]			RTCEN
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:4 – Reserved**

These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.

- **Bit 3:1 – RTCSRC[2:0]: RTC Clock Source**

These bits select the clock source for the real-time counter according to Table 6-4.

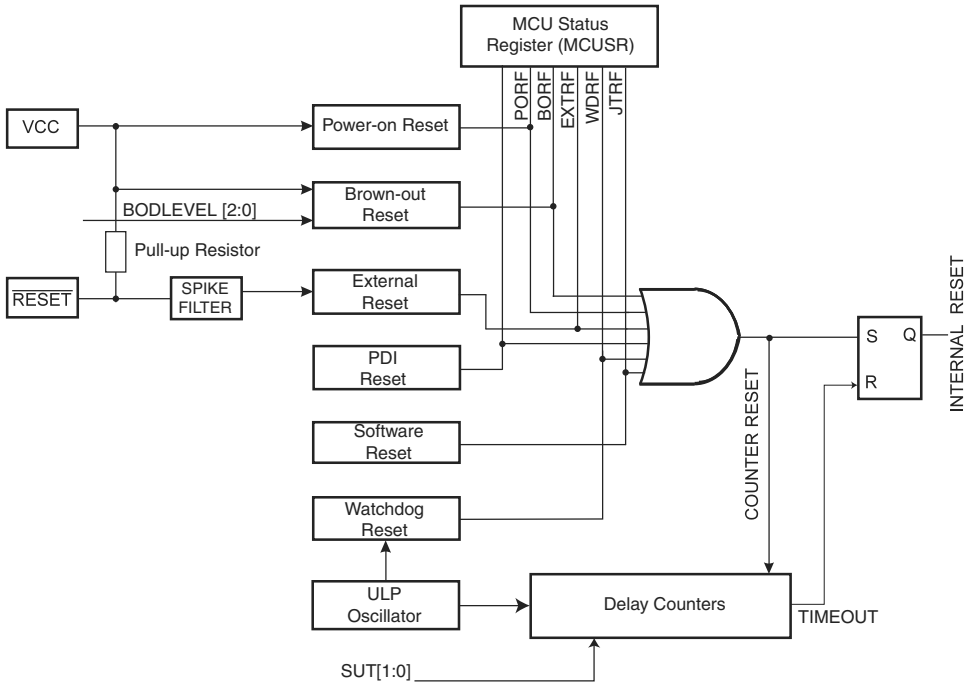
Table 6-4. RTC Clock Source Selection

RTCSRC[2:0]	Group configuration	Description
000	ULP	1kHz from 32kHz internal ULP oscillator
001	TOSC	1.024kHz from 32.768kHz crystal oscillator on TOSC
010	RCOSC	1.024kHz from 32.768kHz internal oscillator
011	—	Reserved
100	—	Reserved
101	TOSC32	32.768kHz from 32.768kHz crystal oscillator on TOSC
110	RCOSC32	32.768kHz from 32.768kHz internal oscillator
111	EXTCLK	External clock from TOSC1

- **Bit 0 – RTCEN: RTC Clock Source Enable**

Setting the RTCEN bit enables the selected RTC clock source for the real-time counter.

Figure 8-1. Reset System Overview



8.3 Reset Sequence

A reset request from any reset source will immediately reset the device and keep it in reset as long as the request is active. When all reset requests are released, the device will go through three stages before the device starts running again:

- Reset counter delay
- Oscillator startup
- Oscillator calibration

If another reset requests occurs during this process, the reset sequence will start over again.

8.3.1 Reset Counter

The reset counter can delay reset release with a programmable period from when all reset requests are released. The reset delay is timed from the 1kHz output of the ultra low power (ULP) internal oscillator, and in addition 24 System clock (clk_{SYS}) cycles are counted before reset is released. The reset delay is set by the STARTUPTIME fuse bits. The selectable delays are shown in Table 8-1.

Table 8-1. Reset Delay

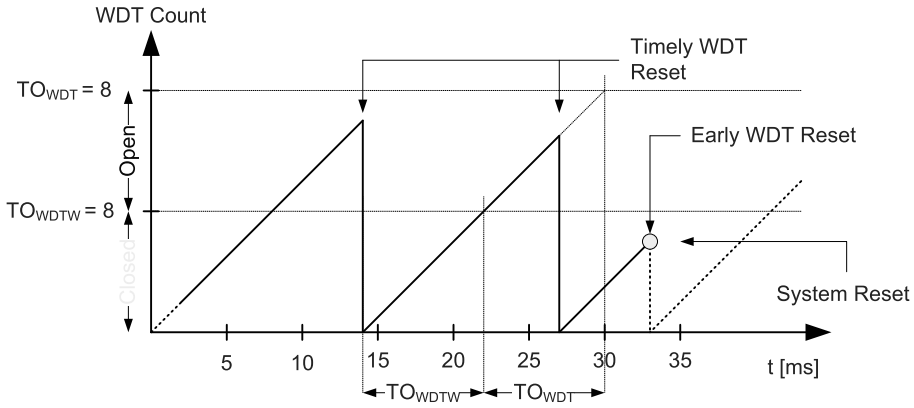
SUT[1:0]	Number of 1kHz ULP oscillator clock cycles	Recommended usage
00	64K Clk_{ULP}+ 24 Clk_{SYS}	Stable frequency at startup
01	4K Clk_{ULP} + 24 Clk_{SYS}	Slowly rising power
10	Reserved	-
11	24 Clk_{SYS}	Fast rising power or BOD enabled

Whenever a reset occurs, the clock system is reset and the internal 2MHz internal oscillator is chosen as the source for Clk_{SYS}.

9.4 Window Mode Operation

In window mode operation, the WDT uses two different timeout periods, a "closed" window timeout period (TO_{WDTW}) and the normal timeout period (TO_{WDT}). The closed window timeout period defines a duration of from 8ms to 8s where the WDT cannot be reset. If the WDT is reset during this period, the WDT will issue a system reset. The normal WDT timeout period, which is also 8ms to 8s, defines the duration of the "open" period during which the WDT can (and should) be reset. The open period will always follow the closed period, and so the total duration of the timeout period is the sum of the closed window and the open window timeout periods. The default closed window timeout period is controlled by fuses (*both* open and closed periods are controlled by fuses). The window mode operation is illustrated in Figure 9-2.

Figure 9-2. Window Mode Operation



9.5 Watchdog Timer Clock

The WDT is clocked from the 1kHz output from the 32kHz ultra low power (ULP) internal oscillator. Due to the ultra low power design, the oscillator is not very accurate, and so the exact timeout period may vary from device to device. When designing software which uses the WDT, this device-to-device variation must be kept in mind to ensure that the timeout periods used are valid for all devices. For more information on ULP oscillator accuracy, consult the device datasheet.

9.6 Configuration Protection and Lock

The WDT is designed with two security mechanisms to avoid unintentional changes to the WDT settings.

The first mechanism is the configuration change protection mechanism, employing a timed write procedure for changing the WDT control registers. In addition, for the new configuration to be written to the control registers, the register's change enable bit must be written at the same time.

The second mechanism locks the configuration by setting the WDT lock fuse. When this fuse is set, the watchdog time control register cannot be changed; hence, the WDT cannot be disabled from software. After system reset, the WDT will resume at the configured operation. When the WDT lock fuse is programmed, the window mode timeout period cannot be changed, but the window mode itself can still be enabled or disabled.

10. Interrupts and Programmable Multilevel Interrupt Controller

10.1 Features

- Short and predictable interrupt response time
- Separate interrupt configuration and vector address for each interrupt
- Programmable multilevel interrupt controller
  - Interrupt prioritizing according to level and vector address
  - Three selectable interrupt levels for all interrupts: low, medium and high
  - Selectable, round-robin priority scheme within low-level interrupts
  - Non-maskable interrupts for critical functions
- Interrupt vectors optionally placed in the application section or the boot loader section

10.2 Overview

Interrupts signal a change of state in peripherals, and this can be used to alter program execution. Peripherals can have one or more interrupts, and all are individually enabled and configured. When an interrupt is enabled and configured, it will generate an interrupt request when the interrupt condition is present. The programmable multilevel interrupt controller (PMIC) controls the handling and prioritizing of interrupt requests. When an interrupt request is acknowledged by the PMIC, the program counter is set to point to the interrupt vector, and the interrupt handler can be executed.

All peripherals can select between three different priority levels for their interrupts: low, medium, and high. Interrupts are prioritized according to their level and their interrupt vector address. Medium-level interrupts will interrupt low-level interrupt handlers. High-level interrupts will interrupt both medium- and low-level interrupt handlers. Within each level, the interrupt priority is decided from the interrupt vector address, where the lowest interrupt vector address has the highest interrupt priority. Low-level interrupts have an optional round-robin scheduling scheme to ensure that all interrupts are serviced within a certain amount of time.

Non-maskable interrupts (NMI) are also supported, and can be used for system critical functions.

10.3 Operation

Interrupts must be globally enabled for any interrupts to be generated. This is done by setting the global interrupt enable (I) bit in the CPU status register. The I bit will not be cleared when an interrupt is acknowledged. Each interrupt level must also be enabled before interrupts with the corresponding level can be generated.

When an interrupt is enabled and the interrupt condition is present, the PMIC will receive the interrupt request. Based on the interrupt level and interrupt priority of any ongoing interrupts, the interrupt is either acknowledged or kept pending until it has priority. When the interrupt request is acknowledged, the program counter is updated to point to the interrupt vector. The interrupt vector is normally a jump to the interrupt handler; the software routine that handles the interrupt. After returning from the interrupt handler, program execution continues from where it was before the interrupt occurred. One instruction is always executed before any pending interrupt is served.

The PMIC status register contains state information that ensures that the PMIC returns to the correct interrupt level when the RETI (interrupt return) instruction is executed at the end of an interrupt handler. Returning from an interrupt will return the PMIC to the state it had before entering the interrupt. The status register (SREG) is not saved automatically upon an interrupt request. The RET (subroutine return) instruction cannot be used when returning from the interrupt handler routine, as this will not return the PMIC to its correct state.

12.12 Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRLA	–	–	–	–	CLKSEL[3:0]				135
+0x01	CTRLB	CCDEN	CCCEN	CCBEN	CCAEN	–	WGMODE[2:0]			135
+0x02	CTRLC	–	–	–	–	CMPD	CMPC	CMPB	CMPA	136
+0x03	CTRLD	EVACT[2:0]			EVDLY	EVSEL[3:0]				136
+0x04	CTRLE	–	–	–	–	–	–	BYTEM		138
+0x05	Reserved	–	–	–	–	–	–	–	–	
+0x06	INTCTRLA	–	–	–	–	ERRINTLVL[1:0]		OVINTLVL[1:0]		138
+0x07	INTCTRLB	CCCINTLVL[1:0]		CCCINTLVL[1:0]		CCBINTLVL[1:0]		CCAINTLVL[1:0]		138
+0x08	CTRLFCLR	–	–	–	–	CMD[1:0]		LUPD	DIR	139
+0x09	CTRLFSET	–	–	–	–	CMD[1:0]		LUPD	DIR	140
+0x0A	CTRLGCLR	–	–	–	CCDBV	CCCBV	CCBBV	CCABV	PERBV	140
+0x0B	CTRLGSET	–	–	–	CCDBV	CCCBV	CCBBV	CCABV	PERBV	140
+0x0C	INTFLAGS	CCDIF	CCCIF	CCBIF	CCAIF	–	–	ERRIF	OVFIF	140
+0x0D	Reserved	–	–	–	–	–	–	–	–	
+0x0E	Reserved	–	–	–	–	–	–	–	–	
+0x0F	TEMP	TEMP[7:0]								141
+0x10 to +0x1F	Reserved	–	–	–	–	–	–	–	–	
+0x20	CNTL	CNT[7:0]								141
+0x21	CNTH	CNT[15:8]								141
+0x22 to +0x25	Reserved	–	–	–	–	–	–	–	–	
+0x26	PERL	PER[7:0]								141
+0x27	PERH	PER[8:15]								142
+0x28	CCAL	CCA[7:0]								142
+0x29	CCAH	CCA[15:8]								142
+0x2A	CCBL	CCB[7:0]								142
+0x2B	CCBH	CCB[15:8]								142
+0x2C	CCCL	CCC[7:0]								142
+0x02D	CCCH	CCC[15:8]								142
+0x2E	CCDL	CCD[7:0]								142
+0x2F	CCDH	CCD[15:8]								142
+0x30 to +0x35	Reserved	–	–	–	–	–	–	–	–	
+0x36	PERBUFL	PERBUF[7:0]								142
+0x37	PERBUFH	PERBUF[15:8]								143
+0x38	CCABUFL	CCABUF[7:0]								143
+0x39	CCABUFH	CCABUF[15:8]								143
+0x3A	CCBBUFL	CCBBUF[7:0]								143
+0x3B	CCBBUFH	CCBBUF[15:8]								143
+0x3C	CCCBUFL	CCCBUF[7:0]								143
+0x3D	CCCBUFH	CCCBUF[15:8]								143
+0x3E	CCDBUFL	CCDBUF[7:0]								143
+0x3F	CCDBUFH	CCDBUF[15:8]								143

12.13 Interrupt Vector Summary

Offset	Source	Interrupt description
0x00	OVF_vect	Timer/counter overflow/underflow interrupt vector offset
0x02	ERR_vect	Timer/counter error interrupt vector offset
0x04	CCA_vect	Timer/counter compare or capture channel A interrupt vector offset
0x06	CCB_vect	Timer/counter compare or capture channel B interrupt vector offset
0x08	CCC_vect(1)	Timer/counter compare or capture channel C interrupt vector offset
0x0A	CCD_vect(1)	Timer/counter compare or capture channel D interrupt vector offset

Note: 1. Available only on timer/counters with four compare or capture channels.

13.10 Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	CTRLA	–	–	–	–	CLKSEL[3:0]				150
+0x01	CTRLB	HCMPDEN	HCMPCEN	HCMPBEN	HCMPAEN	LCMPDEN	LCMPCEN	LCMPBEN	LCMPAEN	150
+0x02	CTRLC	HCMPD	HCMPC	HCMPB	HCMPA	LCMPD	LCMPC	LCMPB	LCMPA	151
+0x03	Reserved	–	–	–	–	–	–	–	–	
+0x04	CTRLE	–	–	–	–	–	–	BYTEM[1:0]		151
+0x05	Reserved	–	–	–	–	–	–	–	–	
+0x06	INTCTRLA	–	–	–	–	HUNFINTLVL[1:0]		LUNFINTLVL[1:0]		151
+0x07	INTCTRLB	LCMPDINTLVL[1:0]		LCMPCINTLVL[1:0]		LCMPBINTLVL[1:0]		LCMPAINTLVL[1:0]		152
+0x08	Reserved	–	–	–	–	–	–	–	–	
+0x09	CTRLF	–	–	–	–	CMD[1:0]		CMDEN[1:0]		152
+0x0A	Reserved	–	–	–	–	–	–	–	–	
+0x0B	Reserved	–	–	–	–	–	–	–	–	
+0x0C	INTFLAGS	LCMPDIF	LCMPCIF	LCMPBIF	LCMPAIF	–	–	HUNFIF	LUNFIF	153
+0x0D	Reserved	–	–	–	–	–	–	–	–	
+0x0E	Reserved	–	–	–	–	–	–	–	–	
+0x0F	Reserved	–	–	–	–	–	–	–	–	
+0x10 to	Reserved	–	–	–	–	–	–	–	–	
+0x20	LCNT	Low-byte Timer/Counter Count Register								153
+0x21	HCNT	High-byte Timer/Counter Count Register								153
+0x22 to	Reserved	–	–	–	–	–	–	–	–	
+0x26	LPER	Low-byte Timer/Counter Period Register								154
+0x27	HPER	High-byte Timer/Counter Period Register								154
+0x28	LCMPA	Low-byte Compare Register A								154
+0x29	HCMPA	High-byte Compare Register A								154
+0x2A	LCMPB	Low-byte Compare Register B								154
+0x2B	HCMPB	High-byte Compare Register B								154
+0x2C	LCMPC	Low-byte Compare Register C								154
+0x02D	HCMPC	High-byte Compare Register C								154
+0x2E	LCMPD	Low-byte Compare Register D								154
+0x2F	HCMPD	High-byte Compare Register D								154
+0x30 to	Reserved	–	–	–	–	–	–	–	–	

13.11 Interrupt Vector Summary

Offset	Source	Interrupt Description
0x00	LUNF_vect	Low-byte Timer/counter underflow interrupt vector offset
0x02	HUNF_vect	High-byte Timer/counter underflow interrupt vector offset
0x4	LCMPA_vect	Low-byte Timer/counter compare channel A interrupt vector offset
0x6	LCMPB_vect	Low-byte Timer/counter compare channel B interrupt vector offset
0x8	LCMPC_vect	Low-byte Timer/counter compare channel C interrupt vector offset
0x0A	LCMPD_vect	Low-byte Timer/counter compare channel D interrupt vector offset

17.9.4 STATUS – Status Register

Bit	7	6	5	4	3	2	1	0
+0x03	**RIF**	**WIF**	**CLKHOLD**	**RXACK**	**ARBLOST**	**BUSERR**	**BUSSTATE[1:0]**	
Read/Write	R/W	R/W	R	R	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7 – RIF: Read Interrupt Flag**

This flag is set when a byte is successfully received in master read mode; i.e., no arbitration was lost or bus error occurred during the operation. Writing a one to this bit location will clear RIF. When this flag is set, the master forces the SCL line low, stretching the TWI clock period. Clearing the interrupt flags will release the SCL line.

This flag is also cleared automatically when:

- Writing to the ADDR register
- Writing to the DATA register
- Reading the DATA register
- Writing a valid command to the CMD bits in the CTRLC register

- **Bit 6 – WIF: Write Interrupt Flag**

This flag is set when a byte is transmitted in master write mode. The flag is set regardless of the occurrence of a bus error or an arbitration lost condition. WIF is also set if arbitration is lost during sending of a NACK in master read mode, and if issuing a START condition when the bus state is unknown. Writing a one to this bit location will clear WIF. When this flag is set, the master forces the SCL line low, stretching the TWI clock period. Clearing the interrupt flags will release the SCL line.

The flag is also cleared automatically for the same conditions as RIF.

- **Bit 5 – CLKHOLD: Clock Hold**

This flag is set when the master is holding the SCL line low. This is a status flag and a read-only flag that is set when RIF or WIF is set. Clearing the interrupt flags and releasing the SCL line will indirectly clear this flag.

The flag is also cleared automatically for the same conditions as RIF.

- **Bit 4 – RXACK: Received Acknowledge**

This flag contains the most recently received acknowledge bit from the slave. This is a read-only flag. When read as zero, the most recent acknowledge bit from the slave was ACK, and when read as one the most recent acknowledge bit was NACK.

- **Bit 3 – ARBLOST: Arbitration Lost**

This flag is set if arbitration is lost while transmitting a high data bit or a NACK bit, or while issuing a START or repeated START condition on the bus. Writing a one to this bit location will clear ARBLOST.

Writing the ADDR register will automatically clear ARBLOST.

- **Bit 2 – BUSERR: Bus Error**

This flag is set if an illegal bus condition has occurred. An illegal bus condition occurs if a repeated START or a STOP condition is detected, and the number of received or transmitted bits from the previous START condition is not a multiple of nine. Writing a one to this bit location will clear BUSERR.

Writing the ADDR register will automatically clear BUSERR.

- **Bit 1:0 – BUSSTATE[1:0]: Bus State**

These bits indicate the current TWI bus state as defined in Table 17-6 on page 189. The change of bus state is dependent on bus activity. Refer to the “TWI Bus State Logic” on page 180.

19.7.1 Receiving Frames

The receiver starts data reception when it detects a valid start bit. Each bit that follows the start bit will be sampled at the baud rate or XCK clock and shifted into the receive shift register until the first stop bit of a frame is received. A second stop bit will be ignored by the receiver. When the first stop bit is received and a complete serial frame is present in the receive shift register, the contents of the shift register will be moved into the receive buffer. The receive complete interrupt flag (RXCIF) is set, and the optional interrupt is generated.

The receiver buffer can be read by reading the data register (DATA) location. DATA should not be read unless the receive complete interrupt flag is set. When using frames with fewer than eight bits, the unused most-significant bits are read as zero. If 9-bit characters are used, the ninth bit must be read from the RXB8 bit before the low byte of the character is read from DATA.

19.7.2 Receiver Error Flags

The USART receiver has three error flags. The frame error (FERR), buffer overflow (BUFOVF) and parity error (PERR) flags are accessible from the status register. The error flags are located in the receive FIFO buffer together with their corresponding frame. Due to the buffering of the error flags, the status register must be read before the receive buffer (DATA), since reading the DATA location changes the FIFO buffer.

19.7.3 Parity Checker

When enabled, the parity checker calculates the parity of the data bits in incoming frames and compares the result with the parity bit of the corresponding frame. If a parity error is detected, the parity error flag is set.

19.7.4 Disabling the Receiver

A disabling of the receiver will be immediate. The receiver buffer will be flushed, and data from ongoing receptions will be lost.

19.7.5 Flushing the Receive Buffer

If the receive buffer has to be flushed during normal operation, read the DATA location until the receive complete interrupt flag is cleared.

19.8 Asynchronous Data Reception

The USART includes a clock recovery and a data recovery unit for handling asynchronous data reception. The clock recovery unit is used for synchronizing the incoming asynchronous serial frames at the RxD pin to the internally generated baud rate clock. It samples and low-pass filters each incoming bit, thereby improving the noise immunity of the receiver. The asynchronous reception operational range depends on the accuracy of the internal baud rate clock, the rate of the incoming frames, and the frame size in number of bits.

19.8.1 Asynchronous Clock Recovery

The clock recovery unit synchronizes the internal clock to the incoming serial frames. Figure 19-6 on page 210 illustrates the sampling process for the start bit of an incoming frame. The sample rate is 16 times the baud rate for normal mode, and eight times the baud rate for double speed mode. The horizontal arrows illustrate the synchronization variation due to the sampling process. Note the larger time variation when using the double speed mode of operation. Samples denoted as zero are samples done when the RxD line is idle; i.e., when there is no communication activity.

21.6.2 STATUS – Status Register

Bit	7	6	5	4	3	2	1	0
+0x02	–	–	–	–	–	–	ZERO	BUSY
Read/Write	R	R	R	R	R	R	R	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:2 – Reserved**

These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.

- **Bit 1 – ZERO: Checksum Zero**

This flag is set if the CHECKSUM is zero when the CRC generation is complete. It is automatically cleared when a new CRC source is selected.

When running CRC-32 and appending the checksum at the end of the packet (as little endian), the final checksum shold be 0x2144df1c, and not zero. However, if the checksum is complemented before it is appended (as little endian) to the data, the final result in the checksum register will be zero.

See the description of CHECKSUM to read out different versions of the CHECKSUM.

- **Bit 0 – BUSY: Busy**

This flag is read as one when a source configuration is selected and as long as the source is using the CRC module. If the I/O interface is selected as the source, the flag can be cleared by writing a one this location. If flash memory is selected as the source, the flag is cleared when the CRC generation is completed.

21.6.3 DATAIN – Data Input Register

Bit	7	6	5	4	3	2	1	0
+0x03	DATAIN[7:0]							
Read/Write	W	W	W	W	W	W	W	W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:0 – DATAIN[7:0]: Data Input**

This register is used to store the data for which the CRC checksum is computed. A new CHECKSUM is ready one clock cycle after the DATAIN register is written.

21.6.4 CHECKSUM0 – Checksum Register 0

CHECKSUM0, CHECKSUM1, CHECKSUM2, and CHECKSUM3 represent the 16- or 32-bit CHECKSUM value and the generated CRC. The registers are reset to zero by default, but it is possible to write RESET to reset all bits to one. It is possible to write these registers only when the CRC module is disabled. If NVM is selected as the source, reading CHECKSUM will return a zero value until the BUSY flag is cleared. If CRC-32 is selected and the BUSY flag is cleared (i.e., CRC generation is completed or aborted), the bit reversed (bit 31 is swapped with bit 0, bit 30 with bit 1, etc.) and complemented result will be read from CHECKSUM. If CRC-16 is selected or the BUSY flag is set (i.e., CRC generation is ongoing), CHECKSUM will contain the actual content.

Bit	7	6	5	4	3	2	1	0
+0x04	CHECKSUM[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:0 – CHECKSUM[7:0]: Checksum byte 0**

These bits hold byte 0 of the generated CRC.

22. ADC – Analog-to-Digital Converter

22.1 Features

- 12-bit resolution
- Up to 300 thousand samples per second
  - Down to 2.3µs conversion time with 8-bit resolution
  - Down to 3.35µs conversion time with 12-bit resolution
- Differential and single-ended input
  - Up to 16 single-ended inputs
  - Up to 16x4 differential inputs without gain
  - 8x4 differential input with gain
- Built-in differential gain stage
  - 1/2x, 1x, 2x, 4x, 8x, 16x, 32x, and 64x gain options
- Single, continuous and scan conversion options
- Three internal inputs
  - Internal temperature sensor
  - AV_{CC} voltage divided by 10
  - 1.1V bandgap voltage
- Internal and external reference options
- Compare function for accurate monitoring of user defined thresholds
- Optional event triggered conversion for accurate timing
- Optional interrupt/event on compare result

22.2 Overview

The ADC converts analog signals to digital values. The ADC has 12-bit resolution and is capable of converting up to 300 thousand samples per second (ksps). The input selection is flexible, and both single-ended and differential measurements can be done. For differential measurements, an optional gain stage is available to increase the dynamic range. In addition, several internal signal inputs are available. The ADC can provide both signed and unsigned results.

The ADC measurements can either be started by application software or an incoming event from another peripheral in the device. The ADC measurements can be started with predictable timing, and without software intervention.

Both internal and external reference voltages can be used. An integrated temperature sensor is available for use with the ADC. The AV_{CC}/10 and the bandgap voltage can also be measured by the ADC.

The ADC has a compare function for accurate monitoring of user defined thresholds with minimum software intervention required.

Table 22-11. ADC MUXPOS Configuration when INPUTMODE[1:0] = 01 (Single-ended), INPUTMODE[1:0] = 10 (Differential without Gain), or INPUTMODE[1:0]=11 (Differential with Gain) is used

MUXPOS[3:0]	Group configuration	Description
0000	PIN0	ADC0 pin
0001	PIN1	ADC1 pin
0010	PIN2	ADC2 pin
0011	PIN3	ADC3 pin
0100	PIN4	ADC4 pin
0101	PIN5	ADC5 pin
0110	PIN6	ADC6 pin
0111	PIN7	ADC7 pin
1000	PIN8	ADC8 pin
1001	PIN9	ADC9 pin
1010	PIN10	ADC10 pin
1011	PIN11	ADC11 pin
1100	PIN12	ADC12 pin
1101	PIN13	ADC13 pin
1110	PIN14	ADC14 pin
1111	PIN15	ADC15 pin

Depending on the device pin count and feature configuration, the actual number of analog input pins may be less than 16. Refer to the device datasheet and pin-out description for details.

- **Bit 2:0 – MUXNEG[2:0]: MUX Selection on Negative ADC Input**

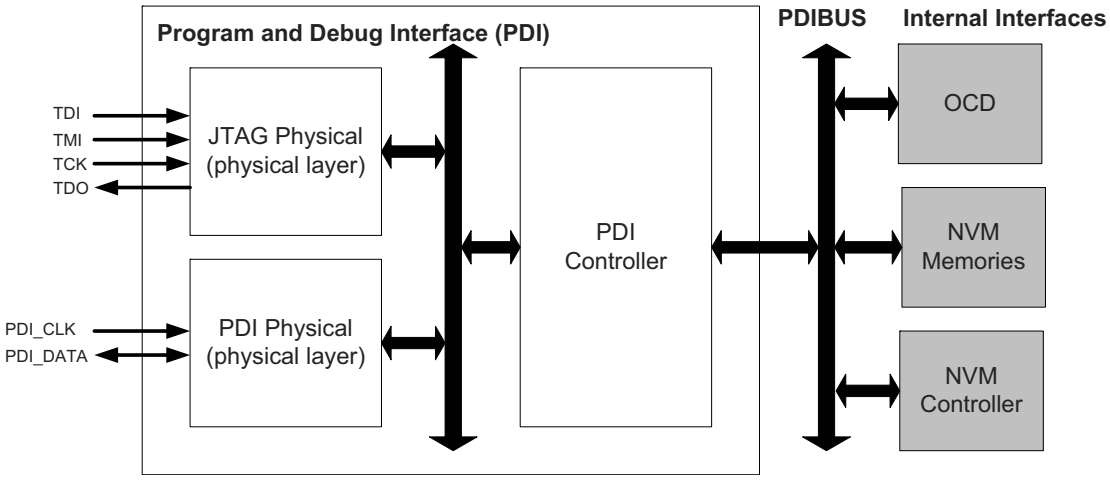
These bits define the MUX selection for the negative ADC input when differential measurements are done. For internal or single-ended measurements, these bits are not used.

Table 22-12 and Table 22-13 on page 250 show the possible input sections.

Table 22-12. ADC MUXNEG Configuration, INPUTMODE[1:0] = 10, Differential without Gain

MUXNEG[2:0]	Group configuration	Analog input
000	PIN0	ADC0 pin
001	PIN1	ADC1 pin
010	PIN2	ADC2 pin
011	PIN3	ADC3 pin
100	–	Reserved
101	GND	PAD ground
110	–	Reserved
111	INTGND	Internal ground

Figure 24-1. The PDI and PDI Physical Layers and Closely Related Modules (grey)

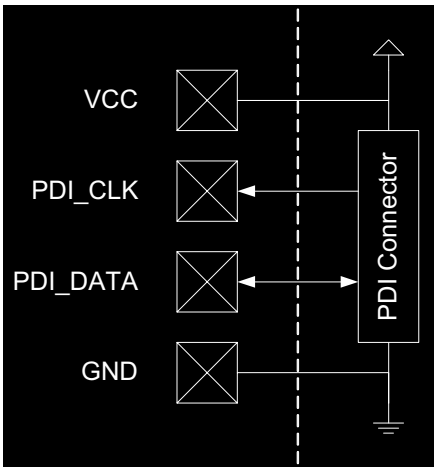


24.3 PDI Physical

The PDI physical layer handles the low-level serial communication. It uses a bidirectional, half-duplex, synchronous serial receiver and transmitter (just as a USART in USRT mode). The physical layer includes start-of-frame detection, frame error detection, parity generation, parity error detection, and collision detection.

In addition to PDI_CLK and PDI_DATA, the PDI_DATA pin has an internal pull resistor, V_{CC} and GND must be connected between the External Programmer/debugger and the device. Figure 24-2 shows a typical connection.

Figure 24-2. PDI Connection



The remainder of this section is intended for use only by third parties developing programmers or programming support for Atmel AVR XMEGA devices.

24.3.1 Enabling

The PDI physical layer must be enabled before use. This is done by first forcing the PDI_DATA line high for a period longer than the equivalent external reset minimum pulse width (refer to device datasheet for external reset pulse width data). This will disable the $\overline{\text{RESET}}$ functionality of the Reset pin, if not already disabled by the fuse settings.

Next, continue to keep the PDI_DATA line high for 16 PDI_CLK cycles. The first PDI_CLK cycle must start no later than 100µs after the $\overline{\text{RESET}}$ functionality of the Reset pin is disabled. If this does not occur in time, the enabling procedure must start over again. The enable sequence is shown in Figure 24-3 on page 265.

Table 25-2. Flash Self-programming Commands

CMD[6:0]	Group configuration	Description	Trigger	CPU halted	NVM busy	Change protected	Address pointer	Data register
0x00	NO_OPERATION	No operation / read flash	-/(E)LPM	-/N	N	-/N	-/ Z-pointer	-/Rd
Flash Page Buffer								
0x23	LOAD_FLASH_BUFFER	Load flash page buffer	SPM	N	N	N	Z-pointer	R1:R0
0x26	ERASE_FLASH_BUFFER	Erase flash page buffer	CMDEX	N	Y	Y	Z-pointer	-
Flash								
0x2B	ERASE_FLASH_PAGE	Erase flash page	SPM	N/Y(2)	Y	Y	Z-pointer	-
0x02E	WRITE_FLASH_PAGE	Write flash page	SPM	N/Y(2)	Y	Y	Z-pointer	-
0x2F	ERASE_WRITE_FLASH_PAGE	Erase and write flash page	SPM	N/Y(2)	Y	Y	Z-pointer	-
0x3A	FLASH_RANGE_CRC(3)	Flash range CRC	CMDEX	Y	Y	Y	DATA/ADDR(1)	DATA
Application Section								
0x20	ERASE_APP	Erase application section	SPM	Y	Y	Y	Z-pointer	-
0x22	ERASE_APP_PAGE	Erase application section page	SPM	N	Y	Y	Z-pointer	-
0x24	WRITE_APP_PAGE	Write application section page	SPM	N	Y	Y	Z-pointer	-
0x25	ERASE_WRITE_APP_PAGE	Erase and write application section page	SPM	N	Y	Y	Z-pointer	-
0x38	APP_CRC	Application section CRC	CMDEX	Y	Y	Y	-	DATA
Boot Loader Section								
0x2A	ERASE_BOOT_PAGE	Erase boot loader section page	SPM	Y	Y	Y	Z-pointer	-
0x2C	WRITE_BOOT_PAGE	Write boot loader section page	SPM	Y	Y	Y	Z-pointer	-
0x2D	ERASE_WRITE_BOOT_PAGE	Erase and write boot loader section page	SPM	Y	Y	Y	Z-pointer	-
0x39	BOOT_CRC	Boot loader section CRC	CMDEX	Y	Y	Y	-	DATA
User Signature Row								
0x01(4)	READ_USER_SIG_ROW	Read user signature row	LPM	N	N	N	Z-pointer	Rd
0x18	ERASE_USER_SIG_ROW	Erase user signature row	SPM	Y	Y	Y	-	-
0x1A	WRITE_USER_SIG_ROW	Write user signature row	SPM	Y	Y	Y	-	-
Production Signature (Calibration) Row(5)								
0x02(4)	READ_CALIB_ROW	Read calibration row	LPM	N	N	N	Z-pointer	Rd

Notes:
1. The flash range CRC command used byte addressing of the flash.
2. Will depend on the flash section (application or boot loader) that is actually addressed.
3. This command is qualified with the lock bits, and requires that the boot lock bits are unprogrammed.
4. When using a command that changes the normal behavior of the LPM command; READ_USER_SIG_ROW and READ_CALIB_ROW; it is recommended to disable interrupts to ensure correct execution of the LPM instruction.
5. For consistency the name Calibration Row has been renamed to Production Signature Row throughout the document.

25.11.2.1 Read Flash

The (E)LPM instruction is used to read one byte from the flash memory.

1. Load the Z-pointer with the byte address to read.
2. Load the NVM command register (NVM CMD) with the no operation command.
3. Execute the LPM instruction.

The destination register will be loaded during the execution of the LPM instruction.

25.11.5.6 Erase EEPROM

The erase EEPROM command is used to erase all locations in all EEPROM pages that are loaded and tagged in the EEPROM page buffer.

1. Set up the NVM CMD register to the erase EPPROM command.
2. Set the CMDEX bit in the NVM CTRLA register. This requires the timed CCP sequence during self-programming.

The BUSY flag in the NVM STATUS register will be set until the operation is finished.

25.11.5.7 Read EEPROM

The read EEPROM command is used to read one byte from the EEPROM.

1. Load the NVM CMD register with the read EEPROM command.
2. Load the NVM ADDR register with the address to read.
3. Set the CMDEX bit in the NVM CTRLA register. This requires the timed CCP sequence during self-programming.

The data byte read will be available in the NVM DATA0 register.

25.12 External Programming

External programming is the method for programming code and nonvolatile data into the device from an external programmer or debugger. This can be done by both in-system or in mass production programming.

For external programming, the device is accessed through the PDI and PDI controller, and using either the JTAG or PDI physical connection. For details on PDI and JTAG and how to enable and use the physical interface, refer to “Program and Debug Interface” on page 263. The remainder of this section assumes that the correct physical connection to the PDI is enabled. Doing this all data and program memory spaces are mapped into the linear PDI memory space. Figure 25-3 on page 288 shows the PDI memory space and the base address for each memory space in the device.