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-Bit
Speed	16MHz
Connectivity	CANbus, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, Temp Sensor, WDT
Number of I/O	-
Program Memory Size	32KB (16K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 11x10b; D/A 1x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 150°C (TA)
Mounting Type	Surface Mount
Package / Case	32-TQFP
Supplier Device Package	32-TQFP (7x7)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atmega32m1-15ad

- **Bit 6 – T: Bit Copy Storage**

The bit copy instructions BLD (Bit LoaD) and BST (Bit STore) use the T-bit as source or destination for the operated bit. A bit from a register in the register file can be copied into T by the BST instruction, and a bit in T can be copied into a bit in a register in the register file by the BLD instruction.

- **Bit 5 – H: Half Carry Flag**

The half carry flag H indicates a half carry in some arithmetic operations. Half carry Is useful in BCD arithmetic. See the “Instruction Set Description” for detailed information.

- **Bit 4 – S: Sign Bit, S = N ⊕ V**

The S-bit is always an exclusive or between the negative flag N and the two’s complement overflow flag V. See the “Instruction Set Description” for detailed information.

- **Bit 3 – V: Two’s Complement Overflow Flag**

The two’s complement overflow flag V supports two’s complement arithmetics. See the “Instruction Set Description” for detailed information.

- **Bit 2 – N: Negative Flag**

The negative flag N indicates a negative result in an arithmetic or logic operation. See the “Instruction Set Description” for detailed information.

- **Bit 1 – Z: Zero Flag**

The zero flag Z indicates a zero result in an arithmetic or logic operation. See the “Instruction Set Description” for detailed information.

- **Bit 0 – C: Carry Flag**

The carry flag C indicates a carry in an arithmetic or logic operation. See the “Instruction Set Description” for detailed information.

3.5 General Purpose Register File

The register file is optimized for the AVR enhanced RISC instruction set. In order to achieve the required performance and flexibility, the following input/output schemes are supported by the register file:

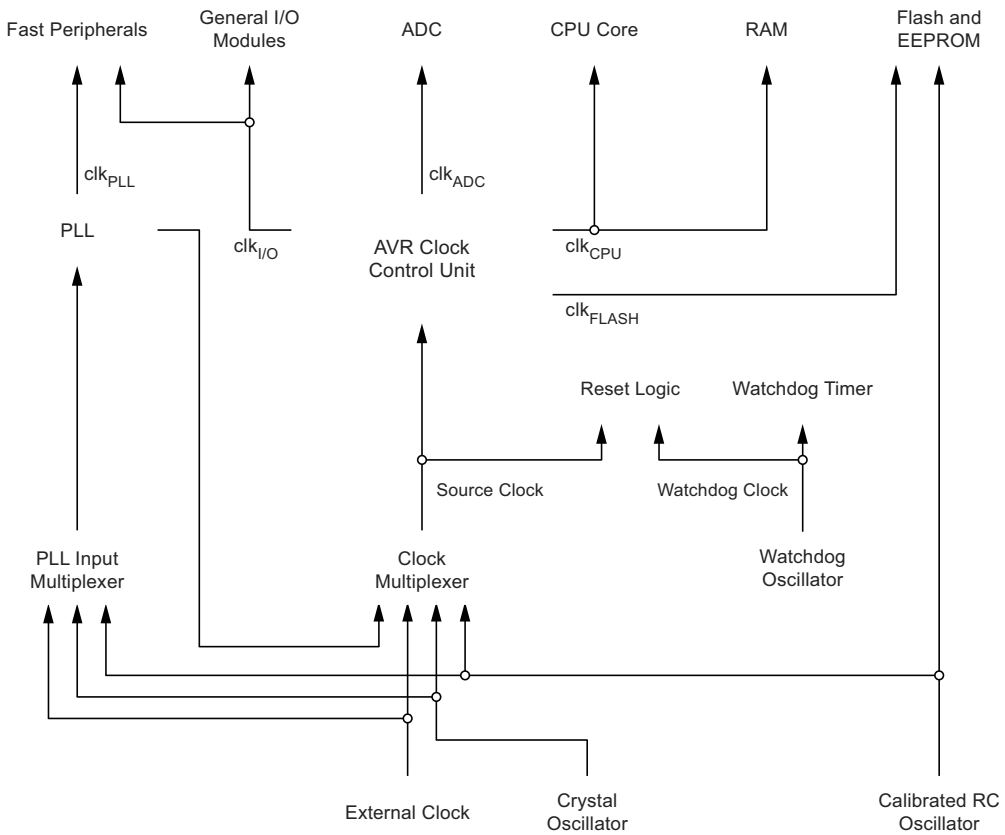
- One 8-bit output operand and one 8-bit result input
- Two 8-bit output operands and one 8-bit result input
- Two 8-bit output operands and one 16-bit result input
- One 16-bit output operand and one 16-bit result input

5. System Clock

5.1 Clock Systems and their Distribution

Figure 5-1 presents the principal clock systems in the AVR® and their distribution. All of the clocks need not be active at a given time. In order to reduce power consumption, the clocks to unused modules can be halted by using different sleep modes, as described in Section 6. “Power Management and Sleep Modes” on page 34. The clock systems are detailed below.

Figure 5-1. Clock Distribution



5.1.1 CPU Clock – clk_{CPU}

The CPU clock is routed to parts of the system concerned with operation of the AVR core. Examples of such modules are the General Purpose Register File, the Status Register and the data memory holding the Stack Pointer. Halting the CPU clock inhibits the core from performing general operations and calculations.

5.1.2 I/O Clock – $clk_{I/O}$

The I/O clock is used by the majority of the I/O modules, like Timer/Counters, SPI, UART. The I/O clock is also used by the External Interrupt module, but note that some external interrupts are detected by asynchronous logic, allowing such interrupts to be detected even if the I/O clock is halted.

5.1.3 Flash Clock – clk_{FLASH}

The Flash clock controls operation of the Flash interface. The flash clock is usually active simultaneously with the CPU clock.

5.3 Default Clock Source

The device is shipped with CKSEL = “0010”, SUT = “10”, and CKDIV8 programmed. The default clock source setting is the Internal RC Oscillator with longest start-up time and an initial system clock prescaling of 8. This default setting ensures that all users can make their desired clock source setting using an in-system or parallel programmer.

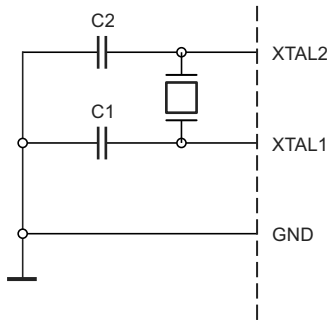
5.4 Low Power Crystal Oscillator

XTAL1 and XTAL2 are input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 5-2. Either a quartz crystal or a ceramic resonator may be used.

This crystal oscillator is a low power oscillator, with reduced voltage swing on the XTAL2 output. It gives the lowest power consumption, but is not capable of driving other clock inputs.

C1 and C2 should always be equal for both crystals and resonators. The optimal value of the capacitors depends on the crystal or resonator in use, the amount of stray capacitance, and the electromagnetic noise of the environment. Some initial guidelines for choosing capacitors for use with crystals are given in Table 5-3. For ceramic resonators, the capacitor values given by the manufacturer should be used. For more information on how to choose capacitors and other details on Oscillator operation, refer to the multi-purpose oscillator application note.

Figure 5-2. Crystal Oscillator Connections



The Oscillator can operate in three different modes, each optimized for a specific frequency range. The operating mode is selected by the fuses CKSEL3..1 as shown in Table 5-3.

Table 5-3. Crystal Oscillator Operating Modes

CKSEL3..1	Frequency Range (MHz)	Recommended Range for Capacitors C1 and C2 for Use with Crystals (pF)
100(1)	0.4 - 0.9	–
101	0.9 - 3.0	12 - 22
110	3.0 - 8.0	12 - 22
111	8.0 -16.0	12 - 22

Note: 1. This option should not be used with crystals, only with ceramic resonators.

When this oscillator is selected, start-up times are determined by the SUT fuses as shown in Table 5-6 on page 29.

Table 5-6. Start-up times for the internal calibrated RC Oscillator clock selection

Power Conditions	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V_{CC} = 5.0V)	SUT1..0
BOD enabled	6 CK	14CK[1]	00
Fast rising power	6 CK	14CK + 4.1ms	01
Slowly rising power	6 CK	14CK + 65ms[2]	10
Reserved			11

Notes: 1. If the RSTDISBL fuse is programmed, this start-up time will be increased to 14CK + 4.1 ms to ensure programming mode can be entered.

2. The device is shipped with this option selected.

5.5.1 Oscillator Calibration Register – OSCCAL

Bit	7	6	5	4	3	2	1	0	
	CAL7	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0	OSCCAL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	Device Specific Calibration Value								

- **Bits 7..0 – CAL7..0: Oscillator Calibration Value**

The oscillator calibration register is used to trim the calibrated internal RC oscillator to remove process variations from the oscillator frequency. The factory-calibrated value is automatically written to this register during chip reset, giving an oscillator frequency of 8.0MHz at 25°C. The application software can write this register to change the oscillator frequency. The oscillator can be calibrated to any frequency in the range 7.3 - 8.1MHz within ±1% accuracy. Calibration outside that range is not guaranteed.

Note that this oscillator is used to time EEPROM and flash write accesses, and these write times will be affected accordingly. If the EEPROM or flash are written, do not calibrate to more than 8.8MHz. Otherwise, the EEPROM or flash write may fail.

The CAL7 bit determines the range of operation for the oscillator. Setting this bit to 0 gives the lowest frequency range, setting this bit to 1 gives the highest frequency range. The two frequency ranges are overlapping, in other words a setting of OSCCAL = 0x7F gives a higher frequency than OSCCAL = 0x80.

The CAL6..0 bits are used to tune the frequency within the selected range. A setting of 0x00 gives the lowest frequency in that range, and a setting of 0x7F gives the highest frequency in the range. Incrementing CAL6..0 by 1 will give a frequency increment of less than 2% in the frequency range 7.3 - 8.1MHz.

5.6 PLL

5.6.1 Internal PLL

The internal PLL in the Atmel® ATmega16/32/64/M1/C1 generates a clock frequency that is 64x multiplied from its nominal 1MHz input. The source of the 1MHz PLL input clock can be:

- the output of the internal RC oscillator divided by 8
- the output of the crystal oscillator divided by 8
- the external clock divided by 8

See Figure 5-3 on page 30.

When the PLL is locked on the RC Oscillator, adjusting the RC Oscillator via OSCCAL Register, will also modify the PLL clock output. However, even if the possibly divided RC Oscillator is taken to a higher frequency than 8MHz, the PLL output clock frequency saturates at 70MHz (worst case) and remains oscillating at the maximum frequency. It should be noted that the PLL in this case is not locked any more with its 1MHz source clock.

8.1.1 Moving Interrupts Between Application and Boot Space

The MCU control register controls the placement of the interrupt vector table.

8.1.2 MCU Control Register – MCUCR

Bit	7	6	5	4	3	2	1	0	
	SPIPS	–	–	PUD	–	–	IVSEL	IVCE	MCUCR
Read/Write	R/W	R	R	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 1 – IVSEL: Interrupt Vector Select**

When the IVSEL bit is cleared (zero), the Interrupt Vectors are placed at the start of the flash memory. When this bit is set (one), the interrupt vectors are moved to the beginning of the boot loader section of the flash. The actual address of the start of the boot flash section is determined by the BOOTSZ fuses. Refer to Section 24. “Boot Loader Support – Read-while-write Self-Programming ATmega16/32/64/M1/C1” on page 241 for details. To avoid unintentional changes of Interrupt vector tables, a special write procedure must be followed to change the IVSEL bit:

1. Write the interrupt vector change enable (IVCE) bit to one.
2. Within four cycles, write the desired value to IVSEL while writing a zero to IVCE.

Interrupts will automatically be disabled while this sequence is executed. Interrupts are disabled in the cycle IVCE is set, and they remain disabled until after the instruction following the write to IVSEL. If IVSEL is not written, interrupts remain disabled for four cycles. The I-bit in the status register is unaffected by the automatic disabling.

Note: If interrupt vectors are placed in the boot loader section and boot lock bit BLB02 is programmed, interrupts are disabled while executing from the application section. If interrupt vectors are placed in the application section and boot lock bit BLB12 is programed, interrupts are disabled while executing from the boot loader section. Refer to Section 24. “Boot Loader Support – Read-while-write Self-Programming ATmega16/32/64/M1/C1” on page 241 for details on boot lock bits.

- **Bit 0 – IVCE: Interrupt Vector Change Enable**

The IVCE bit must be written to logic one to enable change of the IVSEL bit. IVCE is cleared by hardware four cycles after it is written or when IVSEL is written. Setting the IVCE bit will disable interrupts, as explained in the IVSEL description above. See code example below.

Assembly Code Example

```
Move_interrupts:
       ; Enable change of Interrupt Vectors
       ldi           r16, (1<<IVCE)
       out           MCUCR, r16
       ; Move interrupts to Boot Flash section
       ldi           r16, (1<<IVSEL)
       out           MCUCR, r16
       ret
```

C Code Example

```
void Move_interrupts(void)
{
       /* Enable change of Interrupt Vectors */
       MCUCR = (1<<IVCE);
       /* Move interrupts to Boot Flash section */
       MCUCR = (1<<IVSEL);
}
```

12. 8-bit Timer/Counter0 with PWM

Timer/Counter0 is a general purpose 8-bit Timer/Counter module, with two independent Output Compare Units, and with PWM support. It allows accurate program execution timing (event management) and wave generation. The main features are:

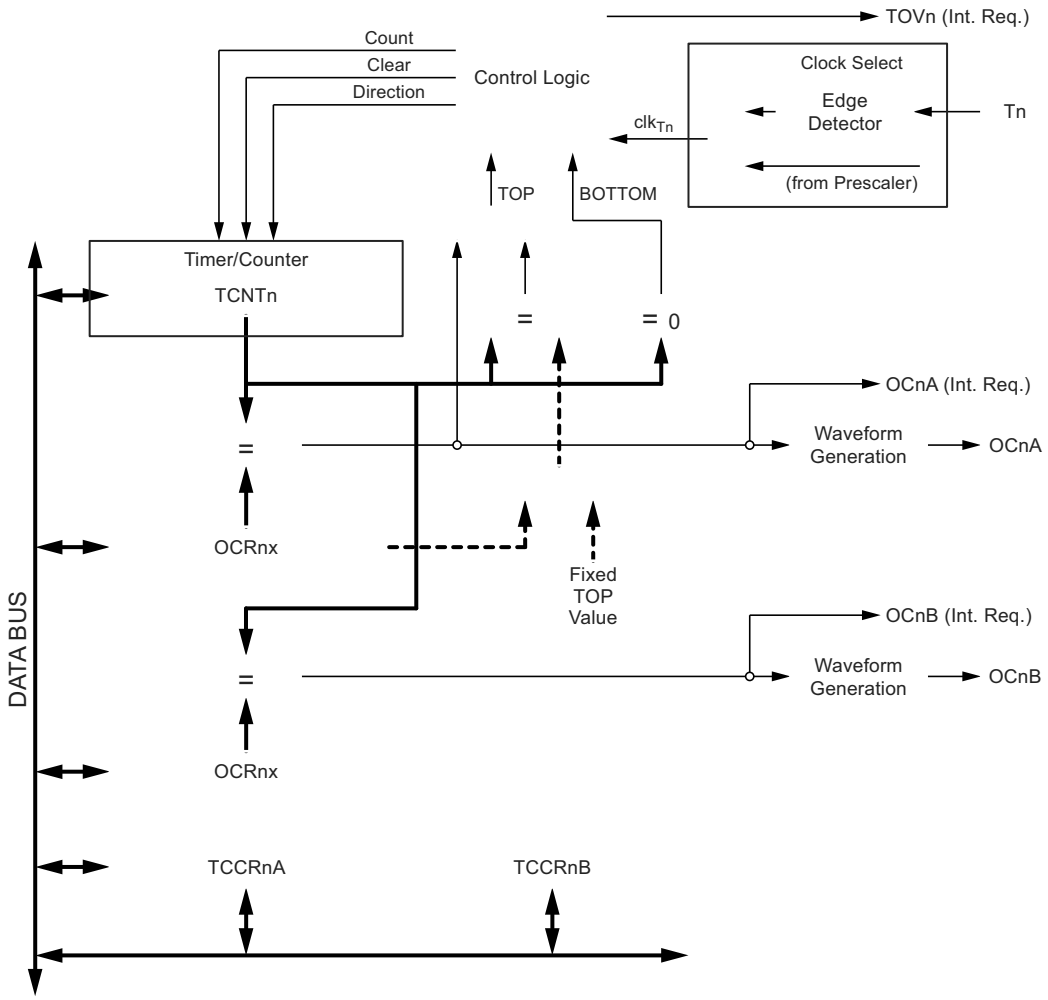
- Two independent output compare units
- Double buffered output compare registers
- Clear timer on compare match (auto reload)
- Glitch free, phase correct pulse width modulator (PWM)
- Variable PWM period
- Frequency generator
- Three independent interrupt sources (TOV0, OCF0A, and OCF0B)

12.1 Overview

A simplified block diagram of the 8-bit Timer/Counter is shown in Figure 12-1. For the actual placement of I/O pins, refer to Section 2.3 “Pin Descriptions” on page 9. CPU accessible I/O registers, including I/O bits and I/O pins, are shown in bold. The device-specific I/O Register and bit locations are listed in Section 12.8 “8-bit Timer/Counter Register Description” on page 86.

The PRTIM0 bit in Section 6.6 “Power Reduction Register” on page 36 must be written to zero to enable Timer/Counter0 module.

Figure 12-1. 8-bit Timer/Counter Block Diagram



The assembly code example returns the TCNTn value in the r17:r16 register pair.

It is important to notice that accessing 16-bit registers are atomic operations. If an interrupt occurs between the two instructions accessing the 16-bit register, and the interrupt code updates the temporary register by accessing the same or any other of the 16-bit timer registers, then the result of the access outside the interrupt will be corrupted. Therefore, when both the main code and the interrupt code update the temporary register, the main code must disable the interrupts during the 16-bit access.

The following code examples show how to do an atomic read of the TCNTn Register contents. Reading any of the OCRnx or ICRn registers can be done by using the same principle.

Assembly Code Example(1)

```
TIM16_ReadTCNTn:
        ; Save global interrupt flag
        in          r18,SREG
        ; Disable interrupts
        cli
        ; Read TCNTn into r17:r16
        in          r16,TCNTnL
        in          r17,TCNTnH
        ; Restore global interrupt flag
        out         SREG,r18
        ret
```

C Code Example(1)

```
unsigned int TIM16_ReadTCNTn( void )
{
        unsigned char sreg;
        unsigned int i;
        /* Save global interrupt flag */
        sreg = SREG;
        /* Disable interrupts */
        _CLI();
        /* Read TCNTn into i */
        i = TCNTn;
        /* Restore global interrupt flag */
        SREG = sreg;
        return i;
}
```

Note: 1. The example code assumes that the part specific header file is included.
For I/O registers located in extended I/O map, “IN”, “OUT”, “SBIS”, “SBIC”, “CBI”, and “SBI” instructions must be replaced with instructions that allow access to extended I/O. Typically “LDS” and “STS” combined with “SBRS”, “SBRC”, “SBR”, and “CBR”.

The assembly code example returns the TCNTn value in the r17:r16 register pair.

Table 13-3 shows the COMnx1:0 bit functionality when the WGMn3:0 bits are set to the phase correct or the phase and frequency correct, PWM mode.

Table 13-3. Compare Output Mode, Phase Correct and Phase and Frequency Correct PWM(1)

COMnA1/COMnB1	COMnA0/COMnB0	Description
0	0	Normal port operation, OCnA/OCnB disconnected.
0	1	WGMn3:0 = 8, 9 10 or 11: Toggle OCnA on compare match, OCnB disconnected (normal port operation). For all other WGM1 settings, normal port operation, OC1A/OC1B disconnected.
1	0	Clear OCnA/OCnB on compare match when up-counting. Set OCnA/OCnB on compare match when downcounting.
1	1	Set OCnA/OCnB on compare match when up-counting. Clear OCnA/OCnB on compare match when downcounting.

Note: 1. A special case occurs when OCRnA/OCRnB equals TOP and COMnA1/COMnB1 is set. See Section 13.8.4 “Phase Correct PWM Mode” on page 105 for more details.

- **Bit 1:0 – WGMn1:0: Waveform Generation Mode**

Combined with the WGMn3:2 bits found in the TCCRnB register, these bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be used, see Table 13-4. Modes of operation supported by the Timer/Counter unit are: normal mode (counter), clear timer on compare match (CTC) mode, and three types of Pulse Width Modulation (PWM) modes (see Section 13. “16-bit Timer/Counter1 with PWM” on page 92).

Table 13-4. Waveform Generation Mode Bit Description(1)

Mode	WGMn3	WGMn2 (CTCn)	WGMn1 (PWMn1)	WGMn0 (PWMn0)	Timer/Counter Mode of Operation	TOP	Update of OCRnx at	TOVn Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, phase correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, phase correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, phase correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCRnA	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	TOP	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	TOP	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	TOP	TOP
8	1	0	0	0	PWM, phase and frequency correct	ICRn	BOTTOM	BOTTOM
9	1	0	0	1	PWM, phase and frequency correct	OCRnA	BOTTOM	BOTTOM
10	1	0	1	0	PWM, phase correct	ICRn	TOP	BOTTOM
11	1	0	1	1	PWM, phase correct	OCRnA	TOP	BOTTOM
12	1	1	0	0	CTC	ICRn	Immediate	MAX
13	1	1	0	1	(Reserved)	–	–	–
14	1	1	1	0	Fast PWM	ICRn	TOP	TOP
15	1	1	1	1	Fast PWM	OCRnA	TOP	TOP

Note: 1. The CTCn and PWMn1:0 bit definition names are obsolete. Use the WGMn2:0 definitions. However, the functionality and location of these bits are compatible with previous versions of the timer.

On-time A = (POCRnRAH/L - POCRnSAH/L) × 1/Fclkpsc

On-time B = (POCRnRBH/L - POCRnSBH/L) × 1/Fclkpsc

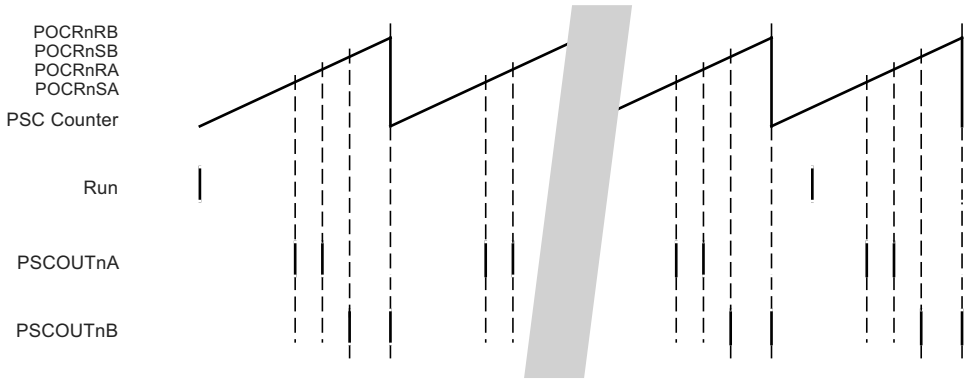
Dead-time A = (POCRnSAH/L + 1) × 1/Fclkpsc

Dead-time B = (POCRnSBH/L – POCRnRAH/L) × 1/Fclkpsc

Minimal value for dead-time A = 1/Fclkpsc

If the overlap protection is disabled, in one-ramp mode, PSCOUTnA and PSCOUTnB outputs can be configured to overlap each other, though in normal use this is not desirable.

Figure 14-5. Controlled Start and Stop Mechanism in One-Ramp Mode

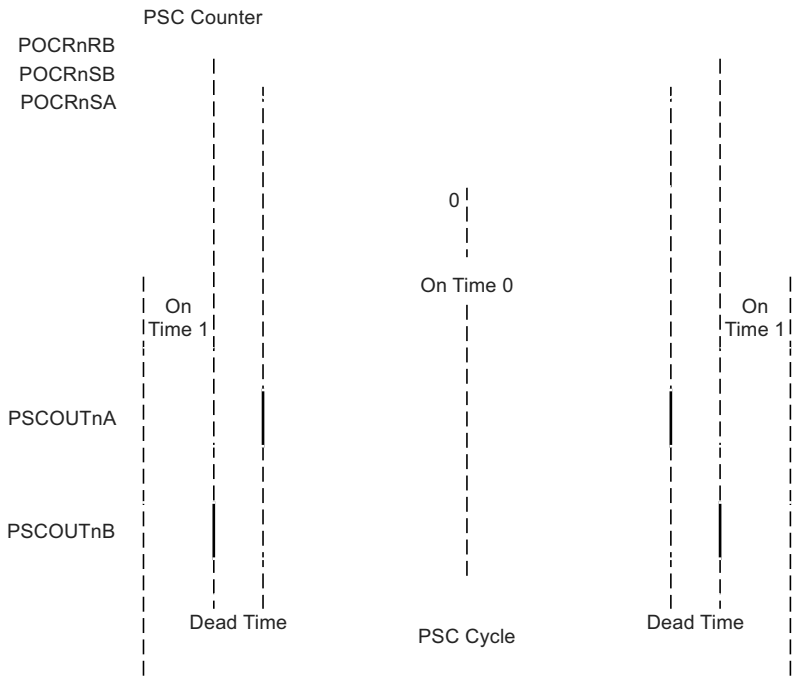


Note: See Section 14.16.8 “PSC Control Register – PCTL” on page 130 (PCCYC = 1)

14.5.3.2 Center Aligned Mode

In center aligned mode, the center of PSCOUTnA and PSCOUTnB signals are centered.

Figure 14-6. PSCOUTnA and PSCOUTnB Basic Waveforms in Center Aligned Mode



14.6.1 Value Update Synchronization

New timing values or PSC output configuration can be written during the PSC cycle. Thanks to LOCK configuration bit, the new whole set of values can be taken into account after the end of the PSC cycle.

When LOCK configuration bit is set, there is no update. The update of the PSC internal registers will be done at the end of the PSC cycle if the LOCK bit is released to zero.

The registers which update is synchronized thanks to LOCK are POC, POM2, POCRnSAH/L, POCRnRAH/L, POCRnSBH/L and POCRnRBH/L.

See these register's description starting on in Section 14.16.7 "PSC Configuration Register – PCNF" on page 130

14.7 Overlap Protection

Thanks to overlap protection two outputs on a same module cannot be active at the same time. So it cannot generate cross conduction. This feature can be disactivated thanks to POVEn (PSC overlap enable).

For ATmega16/64M1, and ATmega32M1 since rev C, the overlap protection is activated with only one condition:

1. POVENn=0 (PSC module n overlap enable)

Up to rev B of ATmega32M1, the overlap protection was activated with the 2 following conditions:

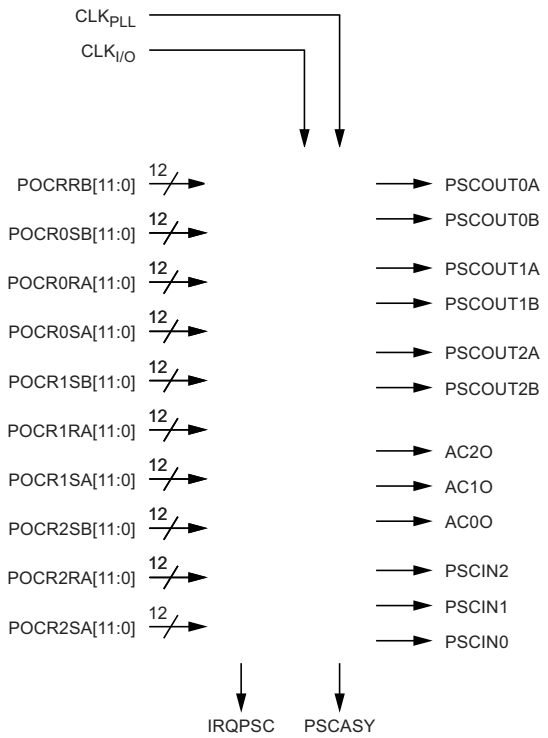
2. POVENn=0 (PSC module n overlap enable)
3. The two channels A and B of a pwm pair n must be activated (POENnA = POENnB = 1)

This difference can induce some behavior change between rev B and rev C of ATmega32M1, when only one channel of a PWM pair output is active.

To avoid such behavior, it is recommended in case of using only one channel of a pwm pair, to disable overlap protection bit (POVENn = 1).

14.8 Signal Description

Figure 14-9. PSC External Block View



16.11 MOb Registers

The MOb registers has **no** initial (default) value after RESET.

16.11.1 CAN MOb Status Register - CANSTMOB

Bit	7	6	5	4	3	2	1	0	
	DLCW	TXOK	RXOK	BERR	SERR	CERR	FERR	AERR	CANSTMOB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	-	-	-	-	-	-	-	-	

- **Bit 7 – DLCW: Data Length Code Warning**

The incoming message does not have the DLC expected. Whatever the frame type, the DLC field of the CANCDMOB register is updated by the received DLC.

- **Bit 6 – TXOK: Transmit OK**

This flag can generate an interrupt. It must be cleared using a read-modify-write software routine on the whole CANSTMOB register.

The communication enabled by transmission is completed. TxOK rises at the end of EOF field. When the controller is ready to send a frame, if two or more message objects are enabled as producers, the lower MOb index (0 to 14) is supplied first.

- **Bit 5 – RXOK: Receive OK**

This flag can generate an interrupt. It must be cleared using a read-modify-write software routine on the whole CANSTMOB register.

The communication enabled by reception is completed. RxOK rises at the end of the 6^{th} bit of EOF field. In case of two or more message object reception hits, the lower MOb index (0 to 14) is updated first.

- **Bit 4 – BERR: Bit Error (Only in Transmission)**

This flag can generate an interrupt. It must be cleared using a read-modify-write software routine on the whole CANSTMOB register.

The bit value monitored is different from the bit value sent.

Exceptions: the monitored recessive bit sent as a dominant bit during the arbitration field and the acknowledge slot detecting a dominant bit during the sending of an error frame.

- **Bit 3 – SERR: Stuff Error**

This flag can generate an interrupt. It must be cleared using a read-modify-write software routine on the whole CANSTMOB register.

Detection of more than five consecutive bits with the same polarity. This flag can generate an interrupt.

- **Bit 2 – CERR: CRC Error**

This flag can generate an interrupt. It must be cleared using a read-modify-write software routine on the whole CANSTMOB register.

The receiver performs a CRC check on every de-stuffed received message from the start of frame up to the data field. If this checking does not match with the de-stuffed CRC field, a CRC error is set.

- **Bit 1 – FERR: Form Error**

This flag can generate an interrupt. It must be cleared using a read-modify-write software routine on the whole CANSTMOB register.

The form error results from one or more violations of the fixed form in the following bit fields:

- CRC delimiter.
- Acknowledgment delimiter.
- EOF

- **Bit 0 – AERR: Acknowledgment Error**

This flag can generate an interrupt. It must be cleared using a read-modify-write software routine on the whole CANSTMOB register.

No detection of the dominant bit in the acknowledge slot.

16.11.2 CAN MOb Control and DLC Register - CANCDMOB

Bit	7	6	5	4	3	2	1	0	
	CONMOB1	**CONMOB0**	**RPLV**	**IDE**	**DLC3**	**DLC2**	**DLC1**	**DLC0**	**CANCDMOB**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	-	-	-	-	-	-	-	-	

- **Bit 7:6 – CONMOB1:0: Configuration of Message Object**

These bits set the communication to be performed (**no** initial value after RESET).

- 00 - disable.
- 01 - enable transmission.
- 10 - enable reception.
- 11 - enable frame buffer reception

These bits are **not** cleared once the communication is performed. The user must re-write the configuration to enable a new communication.

- This operation is necessary to be able to reset the BXOK flag.
- This operation also set the corresponding bit in the CANEN registers.

- **Bit 5 – RPLV: Reply Valid**

Used in the automatic reply mode after receiving a remote frame.

- 0 - reply not ready.
- 1 - reply ready and valid.

- **Bit 4 – IDE: Identifier Extension**

IDE bit of the remote or data frame to send.
This bit is updated with the corresponding value of the remote or data frame received.

- 0 - CAN standard rev 2.0 A (identifiers length = 11 bits).
- 1 - CAN standard rev 2.0 B (identifiers length = 29 bits).

- **Bit 3:0 – DLC3:0: Data Length Code**

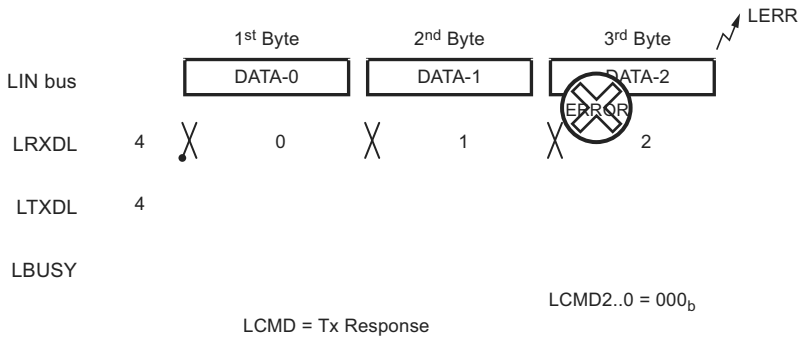
Number of Bytes in the data field of the message.

DLC field of the remote or data frame to send. The range of DLC is from 0 up to 8. If DLC field >8 then effective DLC=8.

This field is updated with the corresponding value of the remote or data frame received. If the expected DLC differs from the incoming DLC, a DLC warning appears in the CANSTMOB register.

17.5.7.5 Data Length after Error

Figure 17-11. Tx Response - Error



Note: Information on response (ex: error on byte) is only available at the end of the serialization/de-serialization of the byte.

17.5.7.6 Data Length in UART Mode

- The UART mode forces LRXDL and LTXDL to 0 and disables the writing in LINDLR register,
- Note that after reset, LRXDL and LTXDL are also forced to 0.

17.5.8 xxOK Flags

There are three xxOK flags in LINSIR register:

- LIDOK: LIN IDentifier OK
  It is set at the end of the header, either by the Tx header function or by the Rx header. In LIN 1.3, before generating LIDOK, the controller updates the LRXDL and LTXDL fields in LINDLR register.
  It is not driven in UART mode.
- LRXOK: LIN RX response complete
  It is set at the end of the response by the Rx response function in LIN mode and once a character is received in UART mode.
- LTXOK: LIN TX response complete
  It is set at the end of the response by the Tx Response function in LIN mode and once a character has been sent in UART mode.

These flags can generate interrupts if the corresponding enable interrupt bit is set in the LINENIR register (see Section 17.5.13 “Interrupts” on page 188).

17.5.9 xxERR Flags

LERR bit of the LINSIR register is an logical ‘OR’ of all the bits of LINERR register (see Section 17.5.13 “Interrupts” on page 188). There are eight flags:

- LBERR = LIN Bit ERRor.
  A unit that is sending a bit on the bus also monitors the bus. A LIN bit error will be flagged when the bit value that is monitored is different from the bit value that is sent. After detection of a LIN bit error the transmission is aborted.
- LCERR = LIN Checksum ERRor.
  A LIN checksum error will be flagged if the inverted modulo-256 sum of all received data bytes (and the protected identifier in LIN 2.1) added to the checksum does not result in 0xFF.

- **Bit 7 – SPMIE: SPM Interrupt Enable**

When the SPMIE bit is written to one, and the I-bit in the status register is set (one), the SPM ready interrupt will be enabled. The SPM ready interrupt will be executed as long as the SPMEN bit in the SPMCSR register is cleared.

- **Bit 6 – RWWSB: Read-while-write Section Busy**

When a self-programming (page erase or page write) operation to the RWW section is initiated, the RWWSB will be set (one) by hardware. When the RWWSB bit is set, the RWW section cannot be accessed. The RWWSB bit will be cleared if the RWWSRE bit is written to one after a self-programming operation is completed. Alternatively the RWWSB bit will automatically be cleared if a page load operation is initiated.

- **Bit 5 – SIGRD: Signature Row Read**

If this bit is written to one at the same time as SPMEN, the next LPM instruction within three clock cycles will read a byte from the signature row into the destination register. see Section 24.7.10 “Reading the Signature Row from Software” on page 249 for details. An SPM instruction within four cycles after SIGRD and SPMEN are set will have no effect. This operation is reserved for future use and should not be used.

- **Bit 4 – RWWSRE: Read-while-write Section Read Enable**

When programming (page erase or page write) to the RWW section, the RWW section is blocked for reading (the RWWSB will be set by hardware). To re-enable the RWW section, the user software must wait until the programming is completed (SPMEN will be cleared). Then, if the RWWSRE bit is written to one at the same time as SPMEN, the next SPM instruction within four clock cycles re-enables the RWW section. The RWW section cannot be re-enabled while the flash is busy with a page erase or a page write (SPMEN is set). If the RWWSRE bit is written while the flash is being loaded, the flash load operation will abort and the data loaded will be lost.

- **Bit 3 – BLBSET: Boot Lock Bit Set**

If this bit is written to one at the same time as SPMEN, the next SPM instruction within four clock cycles sets boot lock bits and memory lock bits, according to the data in R0. The data in R1 and the address in the Z-pointer are ignored. The BLBSET bit will automatically be cleared upon completion of the lock bit set, or if no SPM instruction is executed within four clock cycles.

An LPM instruction within three cycles after BLBSET and SPMEN are set in the SPMCSR register, will read either the Lock bits or the fuse bits (depending on Z0 in the Z-pointer) into the destination register. See Section 24.7.9 “Reading the Fuse and Lock Bits from Software” on page 248 for details.

- **Bit 2 – PGWRT: Page Write**

If this bit is written to one at the same time as SPMEN, the next SPM instruction within four clock cycles executes page write, with the data stored in the temporary buffer. The page address is taken from the high part of the Z-pointer. The data in R1 and R0 are ignored. The PGWRT bit will auto-clear upon completion of a page write, or if no SPM instruction is executed within four clock cycles. The CPU is halted during the entire page write operation if the NRWW section is addressed.

- **Bit 1 – PGERS: Page Erase**

If this bit is written to one at the same time as SPMEN, the next SPM instruction within four clock cycles executes page erase. The page address is taken from the high part of the Z-pointer. The data in R1 and R0 are ignored. The PGERS bit will auto-clear upon completion of a page erase, or if no SPM instruction is executed within four clock cycles. The CPU is halted during the entire Page Write operation if the NRWW section is addressed.

- **Bit 0 – SPMEN: Self Programming Enable**

This bit enables the SPM instruction for the next four clock cycles. If written to one together with either RWWSRE, BLBSET, PGWRT or PGERS, the following SPM instruction will have a special meaning, see description above. If only SPMEN is written, the following SPM instruction will store the value in R1:R0 in the temporary page buffer addressed by the Z-pointer. The LSB of the Z-pointer is ignored. The SPMEN bit will auto-clear upon completion of an SPM instruction, or if no SPM instruction is executed within four clock cycles. during page erase and page write, the SPMEN bit remains high until the operation is completed.

Writing any other combination than “10001”, “01001”, “00101”, “00011” or “00001” in the lower five bits will have no effect.

24.6 Addressing the Flash during Self-Programming

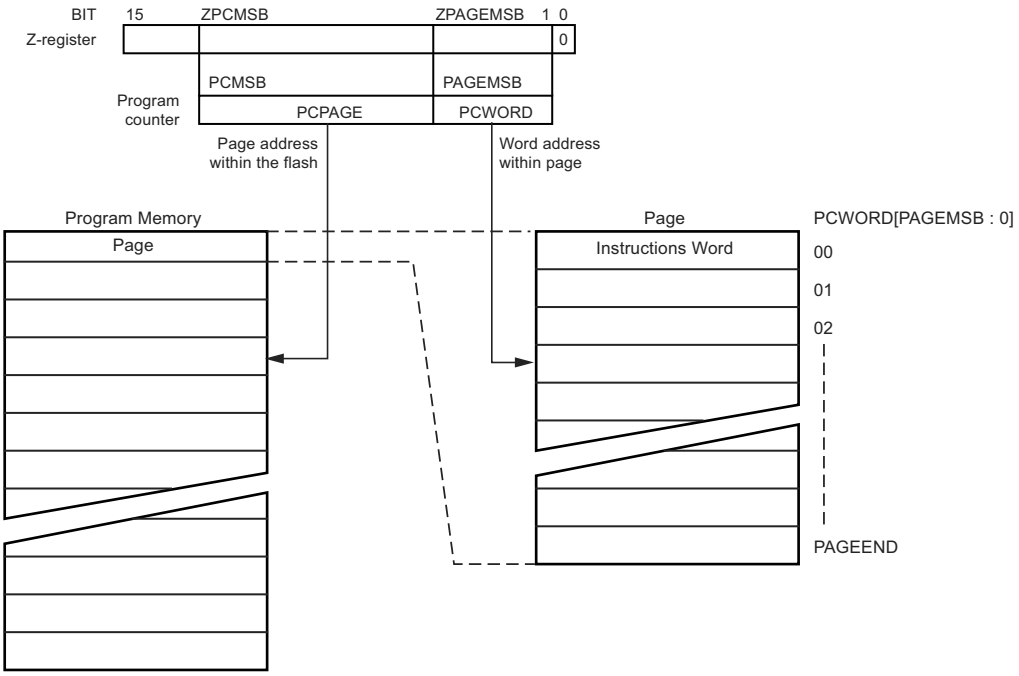
The Z-pointer is used to address the SPM commands.

Bit	15	14	13	12	11	10	9	8
ZH (R31)	**Z15**	**Z14**	**Z13**	**Z12**	**Z11**	**Z10**	**Z9**	**Z8**
ZL (R30)	**Z7**	**Z6**	**Z5**	**Z4**	**Z3**	**Z2**	**Z1**	**Z0**
	7	6	5	4	3	2	1	0

Since the flash is organized in pages (see Table 25-12 on page 260), the program counter can be treated as having two different sections. One section, consisting of the least significant bits, is addressing the words within a page, while the most significant bits are addressing the pages. This is1 shown in Figure 24-3. Note that the page erase and page write operations are addressed independently. Therefore it is of major importance that the boot loader software addresses the same page in both the page erase and page write operation. Once a programming operation is initiated, the address is latched and the Z-pointer can be used for other operations.

The only SPM operation that does not use the Z-pointer is setting the boot loader lock bits. The content of the Z-pointer is ignored and will have no effect on the operation. The LPM instruction does also use the Z-pointer to store the address. Since this instruction addresses the flash byte-by-byte, also the LSB (bit Z0) of the Z-pointer is used.

Figure 24-3. Addressing the Flash during SPM(1)



Note: 1. The different variables used in Figure 24-3 are listed in Table 24-9 on page 252.

24.7 Self-programming the Flash

The program memory is updated in a page by page fashion. Before programming a page with the data stored in the temporary page buffer, the page must be erased. The temporary page buffer is filled one word at a time using SPM and the buffer can be filled either before the page erase command or between a page erase and a page write operation:

Alternative 1, fill the buffer before a page erase

- Fill temporary page buffer
- Perform a page erase
- Perform a page write

Table 24-14. Read-while-write Limit

Section	Pages	Address
Read-while-write section (RWW)	224	0x0000 - 0x6FFF
No read-while-write section (NRWW)	32	0x7000 - 0x7FFF

For details about these two section, see Section 24.3.2 “NRWW – No Read-while-write Section” on page 242 and Section 24.3.1 “RWW – Read-while-write Section” on page 242.

Table 24-15. Explanation of Different Variables used in Figure 24-3 and the Mapping to the Z-pointer

Variable		Corresponding Z-value(1)	Description
PCMSB	14		Most significant bit in the program counter (the program counter is 15 bits PC[14:0]).
PAGEMSB	7		Most significant bit which is used to address the words within one page (128 words in a page requires seven bits PC [6:0]).
ZPCMSB		Z15	Bit in Z-register that is mapped to PCMSB. Because Z0 is not used, the ZPCMSB equals PCMSB + 1.
ZPAGEMSB		Z8	Bit in Z-register that is mapped to PAGEMSB. Because Z0 is not used, the ZPAGEMSB equals PAGEMSB + 1.
PCPAGE	PC[14:7]	Z15:Z8	Program counter page address: Page select, for page erase and page write
PCWORD	PC[6:0]	Z7:Z1	Program counter word address: Word select, for filling temporary buffer (must be zero during page write operation)

Note: 1. Z15:Z13: always ignored
Z0: should be zero for all SPM commands, byte select for the LPM instruction.
See Section 24.6 “Addressing the Flash during Self-Programming” on page 246 for details about the use of Z-pointer during self-programming.

27. ATmega16/32/64/M1/C1 Typical Characteristics

All DC characteristics contained in this datasheet are based on simulations and characterization of similar devices in the same process and design methods. These values are preliminary representing design targets, and will be updated after characterization of actual automotive silicon data.

The following charts show typical behavior. These figures are not tested during manufacturing. All current consumption measurements are performed with all I/O pins configured as inputs and with internal pull-ups enabled. A sine wave generator with rail-to-rail output is used as clock source.

All active- and idle current consumption measurements are done with all bits in the PRR register set and thus, the corresponding I/O modules are turned off. Also the analog comparator is disabled during these measurements. Table 27-1 on page 287 shows the additional current consumption compared to I_{CC} active and I_{CC} idle for every I/O module controlled by the power reduction register. See Section 6.6 “Power Reduction Register” on page 36 for details.

The power consumption in Power-down mode is independent of clock selection.

The current consumption is a function of several factors such as: operating voltage, operating frequency, loading of I/O pins, switching rate of I/O pins, code executed and ambient temperature. The dominating factors are operating voltage and frequency.

The current drawn from capacitive loaded pins may be estimated (for one pin) as $C_L \times V_{CC} \times f$ where C_L = load capacitance, V_{CC} = operating voltage and f = average switching frequency of I/O pin.

The parts are characterized at frequencies higher than test limits. Parts are not guaranteed to function properly at frequencies higher than the ordering code indicates.

The difference between current consumption in power-down mode with watchdog timer enabled and power-down mode with watchdog timer disabled represents the differential current drawn by the watchdog timer.

27.1 Active Supply Current

Figure 27-1. Active Supply Current versus Frequency (0.1 to 1.0MHz)

Figure 27-2. Active Supply Current versus Frequency (1 to 24MHz)

29. Register Summary (Continued)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0xDC)	CANEN2	–	–	ENMOB5	ENMOB4	ENMOB3	ENMOB2	ENMOB1	ENMOB0	162
(0xDB)	CANGIE	ENIT	ENBOFF	ENRX	ENTX	ENERR	ENBX	ENERG	ENOVRT	161
(0xDA)	CANGIT	CANIT	BOFFIT	OVRTIM	BXOK	SERG	CERG	FERG	AERG	160
(0xD9)	CANGSTA	–	OVRG	–	TXBSY	RXBSY	ENFG	BOFF	ERRP	159
(0xD8)	CANGCON	ABRQ	OVRQ	TTC	SYNTTC	LISTEN	TEST	ENA/$\overline{STB}$	SWRES	158
(0xD7)	Reserved	–	–	–	–	–	–	–	–	
(0xD6)	Reserved	–	–	–	–	–	–	–	–	
(0xD5)	Reserved	–	–	–	–	–	–	–	–	
(0xD4)	Reserved	–	–	–	–	–	–	–	–	
(0xD3)	Reserved	–	–	–	–	–	–	–	–	
(0xD2)	LINDAT	LDATA7	LDATA6	LDATA5	LDATA4	LDATA3	LDATA2	LDATA1	LDATA0	196
(0xD1)	LINSEL	–	–	–	–	/LAINC	LINDX2	LINDX1	LINDX0	196
(0xD0)	LINIDR	LP1	LP0	LID5 / LDL1	LID4 / LDL0	LID3	LID2	LID1	LID0	195
(0xCF)	LINDLR	LTXDL3	LTXDL2	LTXDL1	LTXDL0	LRXDL3	LRXDL2	LRXDL1	LRXDL0	195
(0xCE)	LINBRRH	–	–	–	–	LDIV11	LDIV10	LDIV9	LDIV8	194
(0xCD)	LINBRRL	LDIV7	LDIV6	LDIV5	LDIV4	LDIV3	LDIV2	LDIV1	LDIV0	194
(0xCC)	LINBTR	LDISR	–	LBT5	LBT4	LBT3	LBT2	LBT1	LBT0	194
(0xCB)	LINERR	LABORT	LTOERR	LOVERR	LFERR	LSERR	LPERR	LCERR	LBERR	193
(0xCA)	LINENIR	–	–	–	–	LENERR	LENIDOK	LENTXOK	LENRXOK	193
(0xC9)	LINSIR	LIDST2	LIDST1	LIDST0	LBUSY	LERR	LIDOK	LTXOK	LRXOK	192
(0xC8)	LINCR	LSWRES	LIN13	LCONF1	LCONF0	LENA	LCMD2	LCMD1	LCMD0	191
(0xC7)	Reserved	–	–	–	–	–	–	–	–	
(0xC6)	Reserved	–	–	–	–	–	–	–	–	
(0xC5)	Reserved	–	–	–	–	–	–	–	–	
(0xC4)	Reserved	–	–	–	–	–	–	–	–	
(0xC3)	Reserved	–	–	–	–	–	–	–	–	
(0xC2)	Reserved	–	–	–	–	–	–	–	–	
(0xC1)	Reserved	–	–	–	–	–	–	–	–	
(0xC0)	Reserved	–	–	–	–	–	–	–	–	
(0xBF)	Reserved	–	–	–	–	–	–	–	–	
(0xBE)	Reserved	–	–	–	–	–	–	–	–	
(0xBD)	Reserved	–	–	–	–	–	–	–	–	
(0xBC)[5]	PIFR	–	–	–	–	PEV2	PEV1	PEV0	PEOP	132
(0xBB)[5]	PIM	–	–	–	–	PEVE2	PEVE1	PEVE0	PEOPE	132
(0xBA)[5]	PMIC2	POVEN2	PISEL2	PELEV2	PFLTE2	PAOC2	PRFM22	PRFM21	PRFM20	131

Notes:
1. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.
2. I/O registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions.
3. Some of the status flags are cleared by writing a logical one to them. Note that, unlike most other AVRs, the CBI and SBI instructions will only operate on the specified bit, and can therefore be used on registers containing such status flags. The CBI and SBI instructions work with registers 0x00 to 0x1F only.
4. When using the I/O specific commands IN and OUT, the I/O addresses 0x00 - 0x3F must be used. When addressing I/O Registers as data space using LD and ST instructions, 0x20 must be added to these addresses. The ATmega16/32/64/M1/C1 is a complex microcontroller with more peripheral units than can be supported within the 64 location reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 - 0xFF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.
5. These registers are only available on ATmega32/64M1. For other products described in this datasheet, these locations are reserved.

30. Errata

30.1 Errata Summary

30.1.1 ATmega16M1/16C1/32M1/32C1 Rev. C (Mask Revision)

- LIN break delimiter
- ADC with PSC2-synchronized
- ADC amplifier measurement is unstable

30.1.2 ATmega16M1/16C1/32M1/32C1 Rev. B (Mask Revision)

- The AMPCMPx bits return 0
- No comparison when amplifier is used as comparator input and ADC input
- CRC calculation of diagnostic frames in LIN 2.x.
- Wrong TSOFFSET manufacturing calibration value
- PD0-PD3 set to outputs and PD4 pulled down following power-on with external reset active.
- LIN break delimiter
- ADC with PSC2-synchronized
- ADC amplifier measurement is unstable
- PSC emulation
- PSC OCRxx register update according to PLOCK2 usage
- Read/Write instructions of MUXn and REFS1:0

30.1.3 ATmega16M1/16C1/32M1/32C1 Rev. A (Mask Revision)

- Inopportune reset of the CANIDM registers
- The AMPCMPx bits return 0
- No comparison when amplifier is used as comparator input and ADC input
- CRC calculation of diagnostic frames in LIN 2.x
- PD0-PD3 set to outputs and PD4 pulled down following power-on with external reset active
- LIN break delimiter
- ADC with PSC2-synchronized
- ADC amplifier measurement is unstable
- PSC emulation
- Read/Write instructions of MUXn and REFS1:0

30.1.4 Errata Description

1. **Inopportune reset of the CANIDM registers**
   After the reception of a CAN frame in a MOb, the ID mask registers are reset.
   Problem fix / workaround
   Before enabling a MOb in reception, re-initialize the ID mask registers - **CANIDM[4..1]**.
2. **The AMPCMPx bits return 0**
   When they are read the AMPCMPx bits in AMPxCSR registers return 0.
   Problem fix / workaround
   If the reading of the AMPCMPx bits is required, store the AMPCMPx value in a variable in memory before writing in the AMPxCSR register and read the variable when necessary.
3. **No comparison when amplifier is used as comparator input and ADC input**
   When it is selected as ADC input, an amplifier receives no clock signal when the ADC is stopped. In that case, if the amplifier is also used as comparator input, no analog signal is propagated and no comparison is done.
   Problem fix / workaround
   Select another ADC channel rather than the working amplified channel.