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	16KB (8K x 16)
Program Memory Type	FLASH
EEPROM Size	512 x 8
RAM Size	1K 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/atmega16m1-15ad

For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.

Some of the status flags are cleared by writing a logical one to them. Note that, unlike most other AVR's, 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.

The I/O and peripherals control registers are explained in later sections.

4.5 General Purpose I/O Registers

The Atmel® ATmega16/32/64/M1/C1 contains four general purpose I/O registers. These registers can be used for storing any information, and they are particularly useful for storing global variables and status flags.

The general purpose I/O registers, within the address range 0x00 - 0x1F, are directly bit-accessible using the SBI, CBI, SBIS, and SBIC instructions.

4.5.1 General Purpose I/O Register 0 – GPIOR0

Bit	7	6	5	4	3	2	1	0	
	GPIOR07	GPIOR06	GPIOR05	GPIOR04	GPIOR03	GPIOR02	GPIOR01	GPIOR00	GPIOR0
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	

4.5.2 General Purpose I/O Register 1 – GPIOR1

Bit	7	6	5	4	3	2	1	0	
	GPIOR17	GPIOR16	GPIOR15	GPIOR14	GPIOR13	GPIOR12	GPIOR11	GPIOR10	GPIOR1
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	

4.5.3 General Purpose I/O Register 2 – GPIOR2

Bit	7	6	5	4	3	2	1	0	
	GPIOR27	GPIOR26	GPIOR25	GPIOR24	GPIOR23	GPIOR22	GPIOR21	GPIOR20	GPIOR2
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	

Therefore it is recommended not to take the OSCCAL adjustments to a higher frequency than 8MHz in order to keep the PLL in the correct operating range.

The internal PLL is enabled only when the PLLE bit in the register PLLCSR is set. The bit PLOCK from the register PLLCSR is set when PLL is locked.

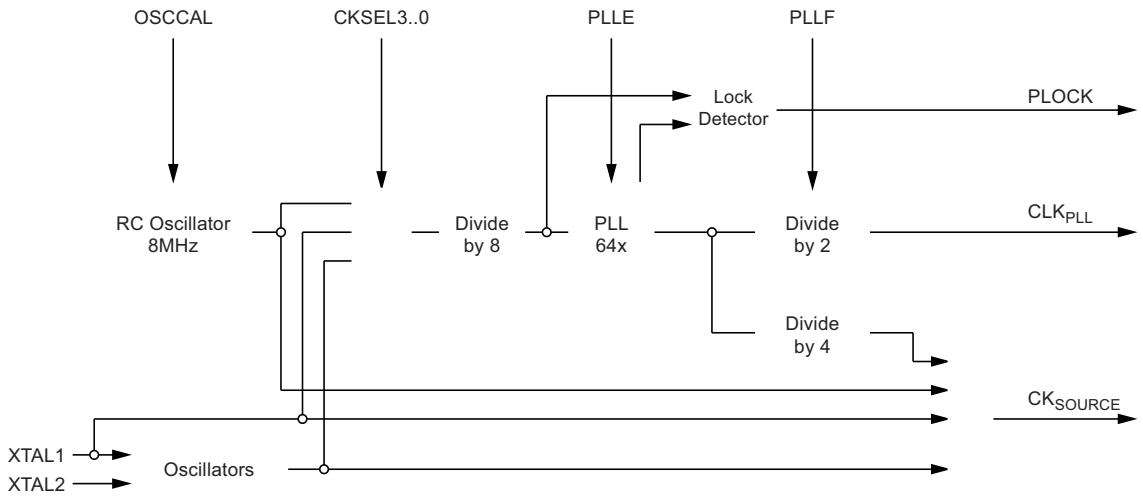
Both internal 8MHz RC Oscillator, Crystal Oscillator and PLL are switched off in Power-down and Standby sleep modes.01/15

Table 5-7. Start-up Times when the PLL is selected as system clock

CKSEL3..0	SUT1..0	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V_{CC} = 5.0V)
0011 RC Osc	00	1K CK	14CK
	01	1K CK	14CK + 4ms
	10	1K CK	14CK + 64ms
	11	16K CK	14CK
0101 Ext Osc	00	1K CK	14CK
	01	1K CK	14CK + 4ms
	10	16K CK	14CK + 4ms
	11	16K CK	14CK + 64ms
0001 Ext Clk	00	6 CK(1)	14CK
	01	6 CK(1)	14CK + 4ms
	10	6 CK(1)	14CK + 64ms
	11	Reserved	

Note: 1. This value do not provide a proper restart; do not use PD in this clock scheme.

Figure 5-3. PLL Clocking System



9.3.5 Alternate Functions of Port E

The Port E pins with alternate functions are shown in Table 9-12.

Table 9-12. Port E Pins Alternate Functions

Port Pin	Alternate Function
PE2	XTAL2 (XTAL Output) ADC0 (Analog Input Channel 0) PCINT26 (Pin Change Interrupt 26)
PE1	XTAL1 (XTAL Input) OC0B (Timer 0 Output Compare B) PCINT25 (Pin Change Interrupt 25)
PE0	RESET# (Reset Input) OCD (On Chip Debug I/O) PCINT24 (Pin Change Interrupt 24)

Note: On the engineering samples (Parts marked AT90PWM324), the ACMPN3 alternate function is not located on PC4. It is located on PE2.

The alternate pin configuration is as follows:

- **PCINT26/XTAL2/ADC0 – Bit 2**

XTAL2: Chip clock oscillator pin 2. Used as clock pin for crystal oscillator or low-frequency crystal oscillator. When used as a clock pin, the pin can not be used as an I/O pin.

ADC0, analog to digital converter, input channel 0.

PCINT26, pin change interrupt 26.

- **PCINT25/XTAL1/OC0B – Bit 1**

XTAL1: Chip clock oscillator pin 1. Used for all chip clock sources except internal calibrated RC oscillator. When used as a clock pin, the pin can not be used as an I/O pin.

OC0B, output compare Match B output: This pin can serve as an external output for the Timer/Counter0 output compare B. The pin has to be configured as an output (DDE1 set “one”) to serve this function. This pin is also the output pin for the PWM mode timer function.

PCINT25, pin change interrupt 25.

- **PCINT24/$\overline{\text{RESET}}$/OCD – Bit 0**

$\overline{\text{RESET}}$, reset pin: When the RSTDISBL Fuse is programmed, this pin functions as a normal I/O pin, and the part will have to rely on power-on reset and brown-out reset as its reset sources. When the RSTDISBL Fuse is unprogrammed, the reset circuitry is connected to the pin, and the pin can not be used as an I/O pin.

If PE0 is used as a reset pin, DDE0, PORTE0 and PINE0 will all read 0.

PCINT24, pin change interrupt 24.

10.2.4 Pin Change Interrupt Flag Register - PCIFR

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	PCIF3	PCIF2	PCIF1	PCIF0	PCIFR
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7..4 - Res: Reserved Bits**

These bits are unused bits in the ATmega16/32/64/M1/C1, and will always read as zero.

- **Bit 3 - PCIF3: Pin Change Interrupt Flag 3**

When a logic change on any PCINT26..24 pin triggers an interrupt request, PCIF3 becomes set (one). If the I-bit in SREG and the PCIE3 bit in PCICR are set (one), the MCU will jump to the corresponding interrupt vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

- **Bit 2 - PCIF2: Pin Change Interrupt Flag 2**

When a logic change on any PCINT23..16 pin triggers an interrupt request, PCIF2 becomes set (one). If the I-bit in SREG and the PCIE2 bit in PCICR are set (one), the MCU will jump to the corresponding Interrupt vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

- **Bit 1 - PCIF1: Pin Change Interrupt Flag 1**

When a logic change on any PCINT15..8 pin triggers an interrupt request, PCIF1 becomes set (one). If the I-bit in SREG and the PCIE1 bit in PCICR are set (one), the MCU will jump to the corresponding Interrupt vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

- **Bit 0 - PCIF0: Pin Change Interrupt Flag 0**

When a logic change on any PCINT7..0 pin triggers an interrupt request, PCIF0 becomes set (one). If the I-bit in SREG and the PCIE0 bit in PCICR are set (one), the MCU will jump to the corresponding Interrupt vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

10.2.5 Pin Change Mask Register 3 – PCMSK3

Bit	7	6	5	4	3	2	1	0	
	-	-	-	-	-	PCINT26	PCINT25	PCINT24	PCMSK3
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..3 – Res: Reserved Bit**

These bits are unused bits in the ATmega16/32/64/M1/C1, and will always read as zero.

- **Bit 2..0 – PCINT26..24: Pin Change Enable Mask 26..24**

Each PCINT26..24-bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT26..24 is set and the PCIE3 bit in PCICR is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT23..24 is cleared, pin change interrupt on the corresponding I/O pin is disabled.

10.2.6 Pin Change Mask Register 2 – PCMSK2

Bit	7	6	5	4	3	2	1	0	
	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16	PCMSK2
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 – PCINT23..16: Pin Change Enable Mask 23..16**

Each PCINT23..16-bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT23..16 is set and the PCIE2 bit in PCICR is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT23..16 is cleared, pin change interrupt on the corresponding I/O pin is disabled.

The OCR0x registers are double buffered when using any of the pulse width modulation (PWM) modes. For the normal and clear timer on compare (CTC) modes of operation, the double buffering is disabled. The double buffering synchronizes the update of the OCR0x compare registers to either top or bottom of the counting sequence. The synchronization prevents the occurrence of odd-length, non-symmetrical PWM pulses, thereby making the output glitch-free.

The OCR0x register access may seem complex, but this is not case. When the double buffering is enabled, the CPU has access to the OCR0x buffer register, and if double buffering is disabled the CPU will access the OCR0x directly.

12.4.1 Force Output Compare

In non-PWM waveform generation modes, the match output of the comparator can be forced by writing a one to the force output compare (FOC0x) bit. Forcing compare match will not set the OCF0x flag or reload/clear the timer, but the OC0x pin will be updated as if a real compare match had occurred (the COM0x1:0 bits settings define whether the OC0x pin is set, cleared or toggled).

12.4.2 Compare Match Blocking by TCNT0 Write

All CPU write operations to the TCNT0 register will block any compare match that occur in the next timer clock cycle, even when the timer is stopped. This feature allows OCR0x to be initialized to the same value as TCNT0 without triggering an interrupt when the Timer/Counter clock is enabled.

12.4.3 Using the Output Compare Unit

Since writing TCNT0 in any mode of operation will block all compare matches for one timer clock cycle, there are risks involved when changing TCNT0 when using the output compare unit, independently of whether the Timer/Counter is running or not. If the value written to TCNT0 equals the OCR0x value, the compare match will be missed, resulting in incorrect waveform generation. Similarly, do not write the TCNT0 value equal to BOTTOM when the counter is downcounting.

The setup of the OC0x should be performed before setting the data direction register for the port pin to output. The easiest way of setting the OC0x value is to use the force output compare (FOC0x) strobe bits in normal mode. The OC0x registers keep their values even when changing between waveform generation modes.

Be aware that the COM0x1:0 bits are not double buffered together with the compare value. Changing the COM0x1:0 bits will take effect immediately.

12.5 Compare Match Output Unit

The compare output mode (COM0x1:0) bits have two functions. The waveform generator uses the COM0x1:0 bits for defining the output compare (OC0x) state at the next compare match. Also, the COM0x1:0 bits control the OC0x pin output source. Figure 12-4 shows a simplified schematic of the logic affected by the COM0x1:0 bit setting. The I/O registers, I/O bits, and I/O pins in the figure are shown in bold. Only the parts of the general I/O port control registers (DDR and PORT) that are affected by the COM0x1:0 bits are shown. When referring to the OC0x state, the reference is for the internal OC0x register, not the OC0x pin. If a system reset occur, the OC0x register is reset to “0”.

Table 12-6 shows the COM0B1:0 bit functionality when the WGM02:0 bits are set to fast PWM mode.

Table 12-6. Compare Output Mode, Fast PWM Mode(1)

COM0B1	COM0B0	Description
0	0	Normal port operation, OC0B disconnected.
0	1	Reserved
1	0	Clear OC0B on compare match, set OC0B at TOP
1	1	Set OC0B on compare match, clear OC0B at TOP

Note: 1. A special case occurs when OCR0B equals TOP and COM0B1 is set. In this case, the compare match is ignored, but the set or clear is done at TOP. See Section 12.6.3 “Fast PWM Mode” on page 83 for more details.

Table 12-7 shows the COM0B1:0 bit functionality when the WGM02:0 bits are set to phase correct PWM mode.

Table 12-7. Compare Output Mode, Phase Correct PWM Mode(1)

COM0B1	COM0B0	Description
0	0	Normal port operation, OC0B disconnected.
0	1	Reserved
1	0	Clear OC0B on compare match when up-counting. Set OC0B on compare match when down-counting.
1	1	Set OC0B on compare match when up-counting. Clear OC0B on compare match when down-counting.

Note: 1. A special case occurs when OCR0B equals TOP and COM0B1 is set. In this case, the compare match is ignored, but the set or clear is done at TOP. See Section 12.6.4 “Phase Correct PWM Mode” on page 84 for more details.

- **Bits 3, 2 – Res: Reserved Bits**

These bits are reserved bits in the ATmega16/32/64/M1/C1 and will always read as zero.

- **Bits 1:0 – WGM01:0: Waveform Generation Mode**

Combined with the WGM02 bit found in the TCCR0B 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 12-8. Modes of operation supported by the Timer/Counter unit are: Normal mode (counter), clear timer on compare match (CTC) mode, and two types of pulse width modulation (PWM) modes (see Section 12.6 “Modes of Operation” on page 81).

Table 12-8. Waveform Generation Mode Bit Description

Mode	WGM02	WGM01	WGM00	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on(1)(2)
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, phase correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	TOP	MAX
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, phase correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCRA	TOP	TOP

Notes: 1. MAX = 0xFF
2. BOTTOM = 0x00

13.5.1 Input Capture Trigger Source

The trigger sources for the input capture unit are the *Input Capture pin* (ICP1A and ICP1B).

Be aware that changing trigger source can trigger a capture. The input capture flag must therefore be cleared after the change.

The *Input Capture pin* (ICPn) IS sampled using the same technique as for the Tn pin (Figure 11-1 on page 75). The edge detector is also identical. However, when the noise canceler is enabled, additional logic is inserted before the edge detector, which increases the delay by four system clock cycles. Note that the input of the noise canceler and edge detector is always enabled unless the Timer/Counter is set in a Waveform Generation mode that uses ICRn to define TOP.

An input capture can be triggered by software by controlling the port of the ICPn pin.

13.5.2 Noise Canceler

The noise canceler improves noise immunity by using a simple digital filtering scheme. The noise canceler input is monitored over four samples, and all four must be equal for changing the output that in turn is used by the edge detector.

The noise canceler is enabled by setting the *Input Capture Noise Canceler* (ICNCn) bit in *Timer/Counter Control Register B* (TCCRnB). When enabled the noise canceler introduces additional four system clock cycles of delay from a change applied to the input, to the update of the ICRn register. The noise canceler uses the system clock and is therefore not affected by the prescaler.

13.5.3 Using the Input Capture Unit

The main challenge when using the input capture unit is to assign enough processor capacity for handling the incoming events. The time between two events is critical. If the processor has not read the captured value in the ICRn register before the next event occurs, the ICRn will be overwritten with a new value. In this case the result of the capture will be incorrect.

When using the input capture interrupt, the ICRn register should be read as early in the interrupt handler routine as possible. Even though the input capture interrupt has relatively high priority, the maximum interrupt response time is dependent on the maximum number of clock cycles it takes to handle any of the other interrupt requests.

Using the input capture unit in any mode of operation when the TOP value (resolution) is actively changed during operation, is not recommended.

Measurement of an external signal's duty cycle requires that the trigger edge is changed after each capture. Changing the edge sensing must be done as early as possible after the ICRn register has been read. After a change of the edge, the input capture flag (ICFn) must be cleared by software (writing a logical one to the I/O bit location). For measuring frequency only, the clearing of the ICFn flag is not required (if an interrupt handler is used).

13.5.4 Using the Input Capture Unit as TCNT1 Retrigger Input

TCNT1 counts from BOTTOM to TOP. The TOP value can be a fixed value, ICR1, or OCR1A. When enabled the retrigger input forces to reach the TOP value. It means that ICF1 output is ored with the TOP signal.

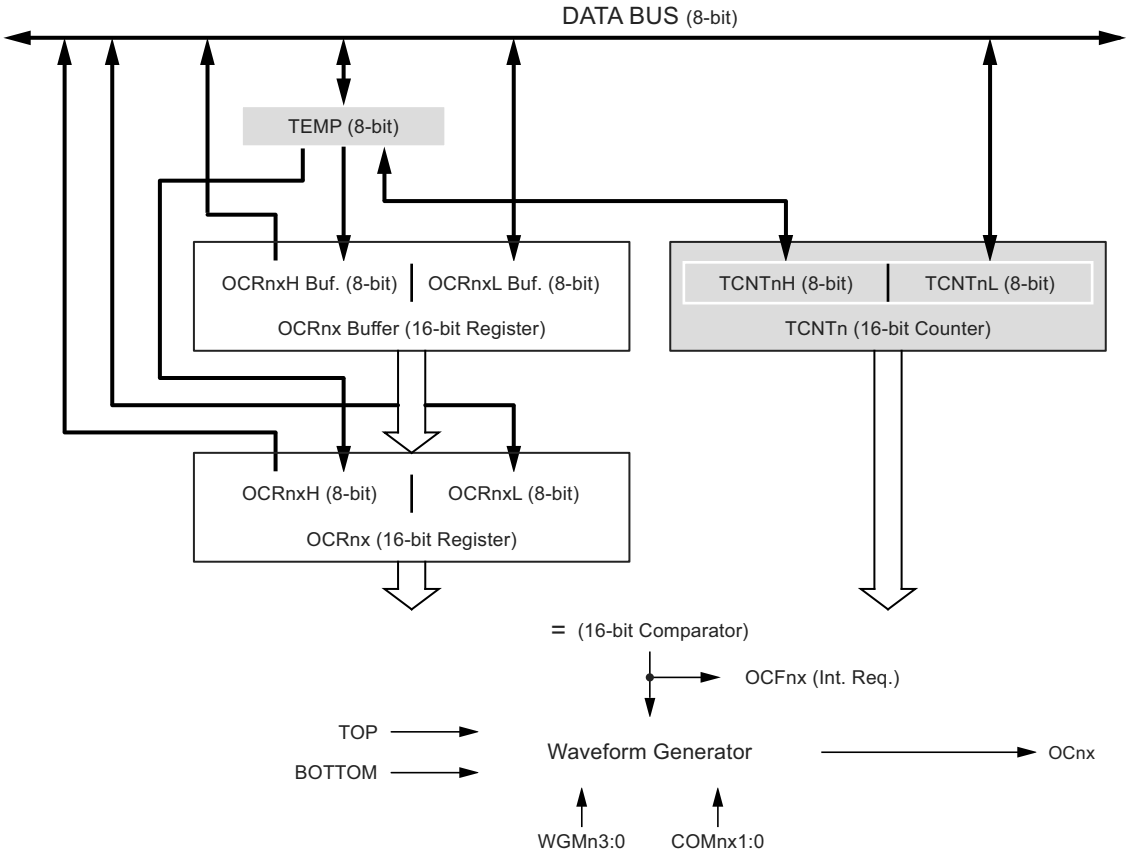
13.6 Output Compare Units

The 16-bit comparator continuously compares TCNTn with the *Output Compare Register* (OCRnx). If TCNT equals OCRnx the comparator signals a match. A match will set the *Output Compare Flag* (OCFnx) at the next timer clock cycle. If enabled (OCIEnx = 1), the output compare flag generates an output compare interrupt. The OCFnx flag is automatically cleared when the interrupt is executed. Alternatively the OCFnx flag can be cleared by software by writing a logical one to its I/O bit location. The waveform generator uses the match signal to generate an output according to operating mode set by the *Waveform Generation mode* (WGMn3:0) bits and *Compare Output mode* (COMnx1:0) bits. The TOP and BOTTOM signals are used by the waveform generator for handling the special cases of the extreme values in some modes of operation (see Section 13. "16-bit Timer/Counter1 with PWM" on page 92)

A special feature of output compare unit A allows it to define the Timer/Counter TOP value (i.e., counter resolution). In addition to the counter resolution, the TOP value defines the period time for waveforms generated by the waveform generator.

Figure 13-4 shows a block diagram of the output compare unit. The small "n" in the register and bit names indicates the device number (n = n for Timer/Counter n), and the "x" indicates output compare unit (x). The elements of the block diagram that are not directly a part of the output compare unit are gray shaded.

Figure 13-4. Output Compare Unit, Block Diagram



The OCRnx register is double buffered when using any of the twelve *Pulse Width Modulation* (PWM) modes. For the normal and *Clear Timer on Compare* (CTC) modes of operation, the double buffering is disabled. The double buffering synchronizes the update of the OCRnx compare register to either TOP or BOTTOM of the counting sequence. The synchronization prevents the occurrence of odd-length, non-symmetrical PWM pulses, thereby making the output glitch-free.

The OCRnx register access may seem complex, but this is not case. When the double buffering is enabled, the CPU has access to the OCRnx buffer register, and if double buffering is disabled the CPU will access the OCRnx directly. The content of the OCR1x (buffer or compare) register is only changed by a write operation (the Timer/Counter does not update this register automatically as the TCNT1 and ICR1 register). Therefore OCR1x is not read via the high byte temporary register (TEMP). However, it is a good practice to read the low byte first as when accessing other 16-bit registers. Writing the OCRnx registers must be done via the TEMP register since the compare of all 16 bits is done continuously. The high byte (OCRnxH) has to be written first. When the high byte I/O location is written by the CPU, the TEMP register will be updated by the value written. Then when the low byte (OCRnxL) is written to the lower eight bits, the high byte will be copied into the upper 8-bits of either the OCRnx buffer or OCRnx compare register in the same system clock cycle.

For more information of how to access the 16-bit registers refer to Section 13.2 “Accessing 16-bit Registers” on page 94.

13.6.1 Force Output Compare

In non-PWM waveform generation modes, the match output of the comparator can be forced by writing a one to the *Force Output Compare* (FOCnx) bit. Forcing compare match will not set the OCFnx Flag or reload/clear the timer, but the OCnx pin will be updated as if a real compare match had occurred (the COMn1:0 bits settings define whether the OCnx pin is set, cleared or toggled).

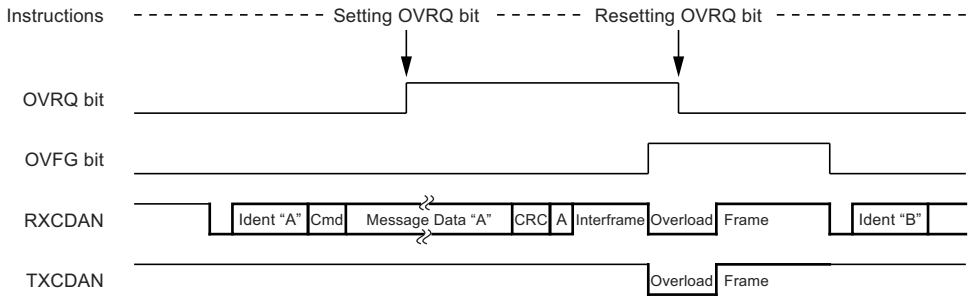
13.6.2 Compare Match Blocking by TCNTn Write

All CPU writes to the TCNTn register will block any compare match that occurs in the next timer clock cycle, even when the timer is stopped. This feature allows OCRnx to be initialized to the same value as TCNTn without triggering an interrupt when the Timer/Counter clock is enabled.

16.4.5 Overload Frame

An overload frame is sent by setting an overload request (OVRQ). After the next reception, the CAN channel sends an overload frame in accordance with the CAN specification. A status or flag is set (OVRF) as long as the overload frame is sent.

Figure 16-9. Overload Frame



16.5 Message Objects

The MOb is a CAN frame descriptor. It contains all information to handle a CAN frame. This means that a MOb has been outlined to allow to describe a CAN message like an object. The set of MObs is the front end part of the "mailbox" where the messages to send and/or to receive are pre-defined as well as possible to decrease the work load of the software.

The MObs are independent but priority is given to the lower one in case of multi matching. The operating modes are:

- Disabled mode
- Transmit mode
- Receive mode
- Automatic reply
- Frame buffer receive mode

16.5.1 Number of MObs

This device has 6 MObs, they are numbered from 0 up to 5 (i=5).

16.5.2 Operating Modes

There is **no** default mode after RESET.

Every MOb has its own fields to control the operating mode. Before enabling the CAN peripheral, each MOb must be configured (ex: disabled mode - CONMOB=00).

Table 16-1. MOb Configuration

MOb Configuration		Reply Valid	RTR Tag	Operating Mode
0	0	x	x	Disabled
0	1	x	0	Tx Data Frame
		x	1	Tx Remote Frame
1	0	x	0	Rx Data Frame
		0	1	Rx Remote Frame
		1		Rx Remote Frame then, Tx Data Frame (reply)
1	1	x	x	Frame Buffer Receive Mode

16.5.2.1 Disabled

In this mode, the MOb is "free".

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

17.4.6.3 Rx and TX Response Functions

These functions are initiated by the slave task of a LIN node. They must be used after sending an header (master task) or after receiving an header (considered as belonging to the slave task). When the TX response order is sent, the transmission begins. A Rx response order can be sent up to the reception of the last serial bit of the first byte (before the stop-bit).

In LIN 1.3, the header slot configures the LINDLR register. In LIN 2.1, the user must configure the LINDLR register, either LRXDL[3..0] for *Rx Response* either LTXDL[3..0] for *Tx Response*.

When the command starts, the controller checks the LIN13 bit of the LINCR register to apply the right rule for computing the checksum. Checksum calculation over the DATA bytes and the PROTECTED IDENTIFIER byte is called enhanced checksum and it is used for communication with LIN 2.1 slaves. Checksum calculation over the DATA bytes only is called classic checksum and it is used for communication with LIN 1.3 slaves. Note that identifiers 60 (0x3C) to 63 (0x3F) shall always use classic checksum.

At the end of this reception or transmission, the controller automatically returns to *Rx Header / LIN Abort* state (i.e. LCMD[1..0] = 00) after setting the appropriate flags.

If an LIN error occurs, the reception or the transmission is stopped, the appropriate flags are set and the LIN bus is left to recessive state.

During these functions, the controller is responsible for:

- The initialization of the checksum operator,
- The transmission or the reception of ‘n’ data with the update of the checksum calculation,
- The transmission or the checking of the CHECKSUM field,
- The checking of the Frame_Time_Out,
- The checking of the LIN communication integrity.

While the controller is sending or receiving a response, BREAK and SYNCH fields can be detected and the identifier of this new header will be recorded. Of course, specific errors on the previous response will be maintained with this identifier reception.

17.4.6.4 Handling Data of LIN response

A FIFO data buffer is used for data of the LIN response. After setting all parameters in the LINSEL register, repeated accesses to the LINDAT register perform data read or data write (c.f. Section 17.5.15 “Data Management” on page 189).

Note that LRXDL[3..0] and LTXDL[3..0] are not linked to the data access.

17.4.7 UART Commands

Setting the LCMD[2] bit in LINENR register enables UART commands.

Tx Byte and Rx Byte services are independent as shown in Table 17-1 on page 178.

- Byte transfer: the UART is selected but both Rx and Tx services are disabled,
- Rx Byte: only the Rx service is enable but Tx service is disabled,
- Tx Byte: only the Tx service is enable but Rx service is disabled,
- Full duplex: the UART is selected and both Rx and Tx services are enabled.

This combination of services is controlled by the LCMD[1..0] bits of LINENR register (c.f. Figure 17-5 on page 178).

17.4.7.1 Data Handling

The FIFO used for LIN communication is disabled during UART accesses. LRXDL[3..0] and LTXDL[3..0] values of LINDLR register are then irrelevant. LINDAT register is then used as data register and LINSEL register is not relevant.

17.4.7.2 Rx Service

Once this service is enabled, the user is warned of an in-coming character by the LRXOK flag of LINSIR register. Reading LINDAT register automatically clears the flag and makes free the second stage of the buffer. If the user considers that the in-coming character is irrelevant without reading it, he directly can clear the flag (see specific flag management described in Section 17.6.2 “LIN Status and Interrupt Register - LINSIR” on page 192). The intrinsic structure of the Rx service offers a 2-byte buffer. The fist one is used for serial to parallel conversion, the second one receives the result of the conversion. This second buffer byte is reached reading LINDAT register. If the 2-byte buffer is full, a new in-coming character will overwrite the second one already recorded. An OVRERR error in LINERR register will then accompany this character when read. A FERR error in LINERR register will be set in case of framing error.

17.5.14 Message Filtering

Message filtering based upon the whole identifier is not implemented. Only a status for frame headers having 0x3C, 0x3D, 0x3E and 0x3F as identifier is available in the LINSIR register.

Table 17-4. Frame Status

LIDST[2..0]	Frame Status
$0xx_b$	No specific identifier
100_b	60 (0x3C) identifier
101_b	61 (0x3D) identifier
110_b	62 (0x3E) identifier
111_b	63 (0x3F) identifier

The LIN protocol says that a message with an identifier from 60 (0x3C) up to 63 (0x3F) uses a classic checksum (sum over the data bytes only). Software will be responsible for switching correctly the LIN13 bit to provide/check this expected checksum (the insertion of the ID field in the computation of the CRC is set - or not - just after entering the Rx or Tx response command).

17.5.15 Data Management

17.5.15.1 LIN FIFO Data Buffer

To preserve register allocation, the LIN data buffer is seen as a FIFO (with address pointer accessible). This FIFO is accessed via the LINDX[2..0] field of LINSEL register through the LINDAT register.

LINDX[2..0], the data index, is the address pointer to the required data byte. The data byte can be read or written. The data index is automatically incremented after each LINDAT access if the LAINC (active low) bit is cleared. A roll-over is implemented, after data index=7 it is data index=0. Otherwise, if LAINC bit is set, the data index needs to be written (updated) before each LINDAT access.

The first byte of a LIN frame is stored at the data index=0, the second one at the data index=1, and so on. Nevertheless, LINSEL must be initialized by the user before use.

17.5.15.2 UART Data Register

The LINDAT register is the data register (no buffering - no FIFO). In write access, LINDAT will be for data out and in read access, LINDAT will be for data in.

In UART mode the LINSEL register is unused.

17.5.16 OCD Support

This section describes the behavior of the LIN/UART controller stopped by the OCD (i.e. I/O view behavior in AVR Studio®)

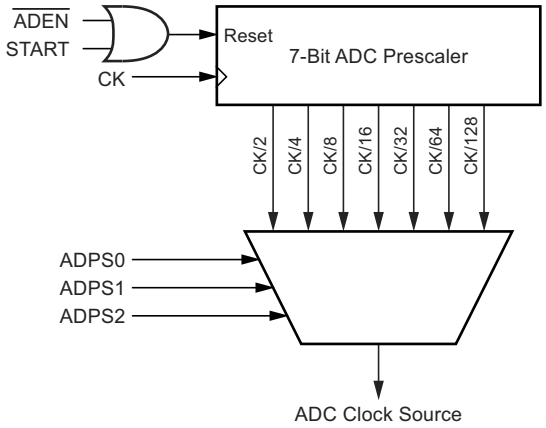
1. LINCR:
   - LINCR[6..0] are R/W accessible,
   - LSWRES always is a self-reset bit (needs 1 micro-controller cycle to execute)
2. LINSIR:
   - LIDST[2..0] and LBUSY are always Read accessible,
   - LERR and LxxOK bit are directly accessible (unlike in execution, set or cleared directly by writing 1 or 0).
   - Note that clearing LERR resets all LINERR bits and setting LERR sets all LINERR bits.
3. LINENR:
   - All bits are R/W accessible.
4. LINERR:
   - All bits are R/W accessible,
   - Note that LINERR bits are ORed to provide the LERR interrupt flag of LINSIR.
5. LINBTR:
   - LBT[5..0] are R/W access only if LDISR is set,
   - If LDISR is reset, LBT[5..0] are unchangeable.

Using the ADC interrupt flag as a trigger source makes the ADC start a new conversion as soon as the ongoing conversion has finished. The ADC then operates in free running mode, constantly sampling and updating the ADC data register. The first conversion must be started by writing a logical one to the ADSC bit in ADCSRA. In this mode the ADC will perform successive conversions independently of whether the ADC Interrupt Flag, ADIF is cleared or not. The free running mode is not allowed on the amplified channels.

If auto triggering is enabled, single conversions can be started by writing ADSC in ADCSRA to one. ADSC can also be used to determine if a conversion is in progress. The ADSC bit will be read as one during a conversion, independently of how the conversion was started.

18.4 Prescaling and Conversion Timing

Figure 18-3. ADC Prescaler



By default, the successive approximation circuitry requires an input clock frequency between 50kHz and 2MHz to get maximum resolution. If a lower resolution than 10 bits is needed, the input clock frequency to the ADC can be higher than 2MHz to get a higher sample rate.

The ADC module contains a prescaler, which generates an acceptable ADC clock frequency from any CPU frequency above 100kHz. The prescaling is set by the ADPS bits in ADCSRA. The prescaler starts counting from the moment the ADC is switched on by setting the ADEN bit in ADCSRA. The prescaler keeps running for as long as the ADEN bit is set, and is continuously reset when ADEN is low.

When initiating a single ended conversion by setting the ADSC bit in ADCSRA, the conversion starts at the following rising edge of the ADC clock cycle. See Section 18.5 “Changing Channel or Reference Selection” on page 202 for details on differential conversion timing.

A normal conversion takes 15.5 ADC clock cycles. The first conversion after the ADC is switched on (ADEN in ADCSRA is set) takes 25 ADC clock cycles in order to initialize the analog circuitry.

The actual sample-and-hold takes place 3.5 ADC clock cycles after the start of a normal conversion and 13.5 ADC clock cycles after the start of an first conversion. When a conversion is complete, the result is written to the ADC data registers, and ADIF is set. In single conversion mode, ADSC is cleared simultaneously. The software may then set ADSC again, and a new conversion will be initiated on the first rising ADC clock edge.

When auto triggering is used, the prescaler is reset when the trigger event occurs. This assures a fixed delay from the trigger event to the start of conversion. In this mode, the sample-and-hold takes place two ADC clock cycles after the rising edge on the trigger source signal. Three additional CPU clock cycles are used for synchronization logic.

In free running mode, a new conversion will be started immediately after the conversion completes, while ADSC remains high. For a summary of conversion times, see Table 18-1 on page 202.

As soon as a conversion is requested thanks to the ADSC bit, the analog to digital conversion is started. In case the amplifier output is modified during the sample phase of the ADC, the on-going conversion is aborted and restarted as soon as the output of the amplifier is stable. This ensure a fast response time. The only precaution to take is to be sure that the trig signal (PSC) frequency is lower than ADCclk/4.

Figure 18-15. Amplifier Synchronization Timing Diagram with Change on Analog Input Signal

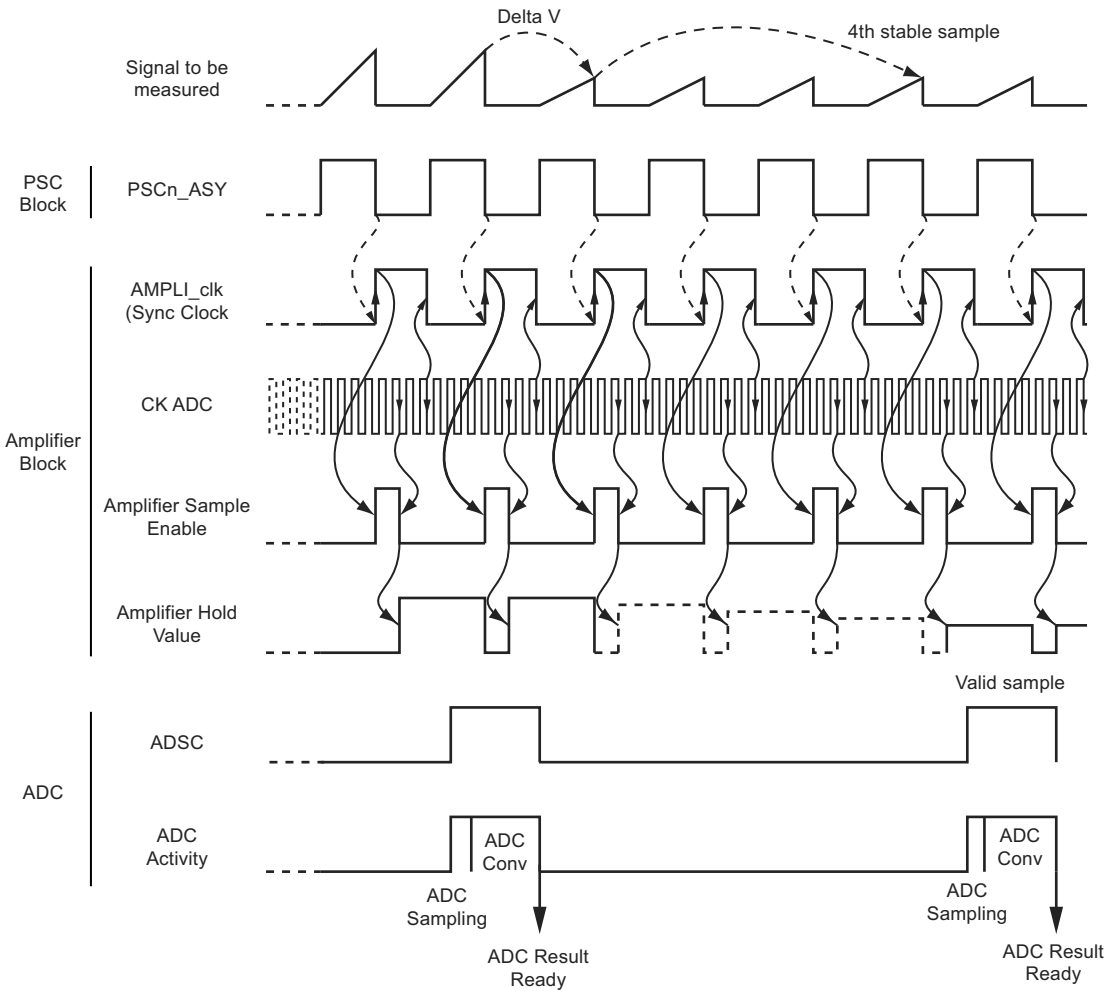
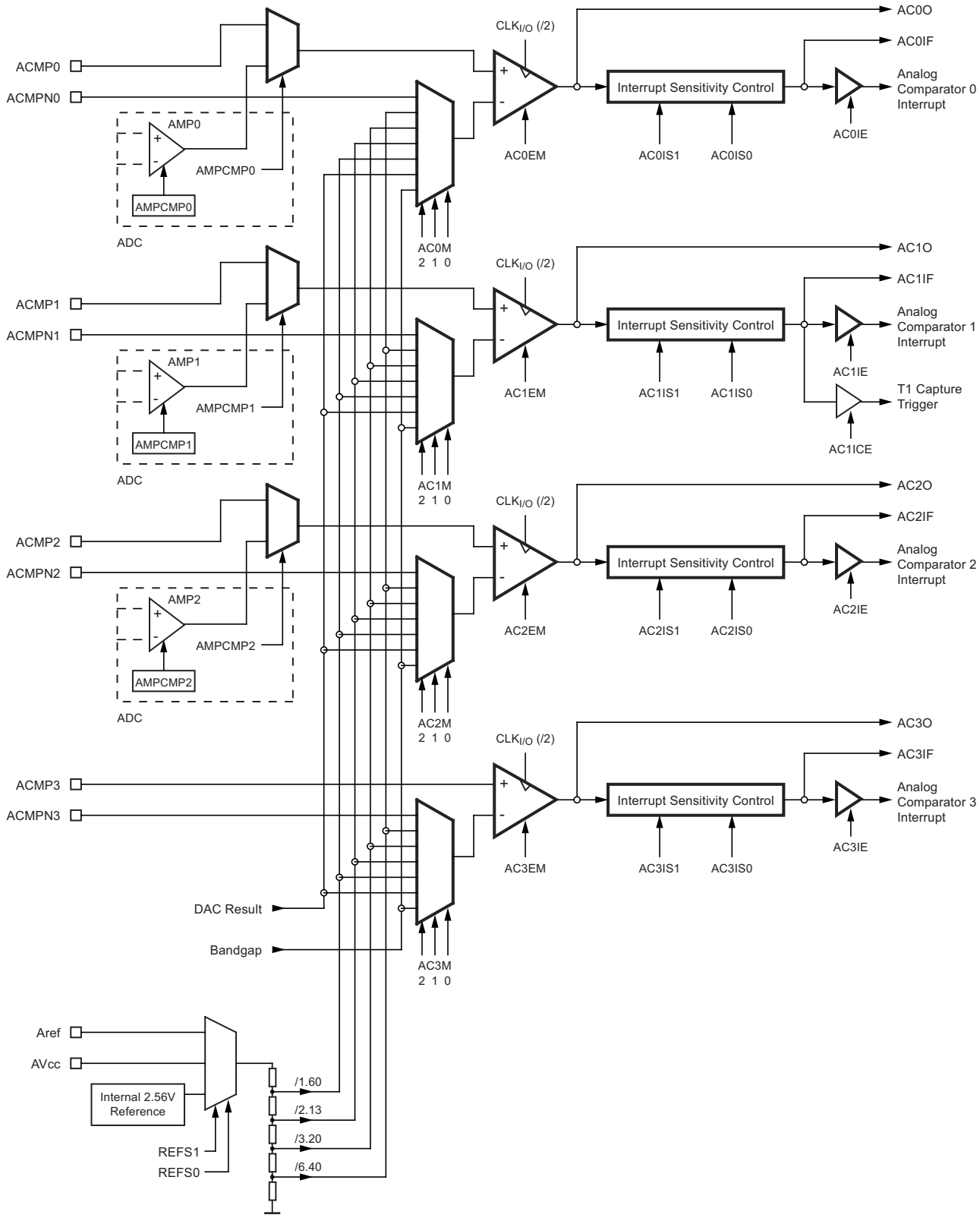


Figure 20-1. Analog Comparator Block Diagram(1)(2)



Notes: 1. ADC multiplexer output: see Table 18-5 on page 211.

2. Refer to Figure 1-1 on page 3 and for analog comparator pin placement.

3. The voltage on Vref is defined in 18-4 “ADC Voltage Reference Selection” on page 210

25. Memory Programming

25.1 Program and Data Memory Lock Bits

The ATmega16/32/64/M1/C1 provides six Lock bits which can be left unprogrammed (“1”) or can be programmed (“0”) to obtain the additional features listed in Table 25-2. The Lock bits can only be erased to “1” with the chip erase command.

Table 25-1. Lock Bit Byte(1)

Lock Bit Byte	Bit No	Description	Default Value
	7	–	1 (unprogrammed)
	6	–	1 (unprogrammed)
BLB12	5	Boot lock bit	1 (unprogrammed)
BLB11	4	Boot lock bit	1 (unprogrammed)
BLB02	3	Boot lock bit	1 (unprogrammed)
BLB01	2	Boot lock bit	1 (unprogrammed)
LB2	1	Lock bit	1 (unprogrammed)
LB1	0	Lock bit	1 (unprogrammed)

Notes: 1. “1” means unprogrammed, “0” means programmed.

Table 25-2. Lock Bit Protection Modes(1)(2)

Memory Lock Bits			
LB Mode	LB2	LB1	Protection Type
1	1	1	No memory lock features enabled.
2	1	0	Further programming of the flash and EEPROM is disabled in parallel and serial programming mode. The fuse bits are locked in both serial and parallel programming mode(1).
3	0	0	Further programming and verification of the flash and EEPROM is disabled in parallel and serial programming mode. The boot lock bits and fuse bits are locked in both serial and parallel programming mode(1).

Notes: 1. Program the fuse bits and boot lock bits before programming the LB1 and LB2.
2. “1” means unprogrammed, “0” means programmed.

Table 25-3. Lock Bit Protection Modes(1)(2).

BLB0 Mode	BLB02	BLB01	
1	1	1	No restrictions for SPM or LPM accessing the Application section.
2	1	0	SPM is not allowed to write to the Application section.
3	0	0	SPM is not allowed to write to the Application section, and LPM executing from the Boot Loader section is not allowed to read from the Application section. If Interrupt Vectors are placed in the Boot Loader section, interrupts are disabled while executing from the Application section.
4	0	1	LPM executing from the Boot Loader section is not allowed to read from the Application section. If Interrupt Vectors are placed in the Boot Loader section, interrupts are disabled while executing from the Application section.

Notes: 1. Program the Fuse bits and Boot Lock bits before programming the LB1 and LB2.
2. “1” means unprogrammed, “0” means programmed

25.8.15 Parallel Programming Characteristics

Figure 25-7. Parallel Programming Timing, Including some General Timing Requirements

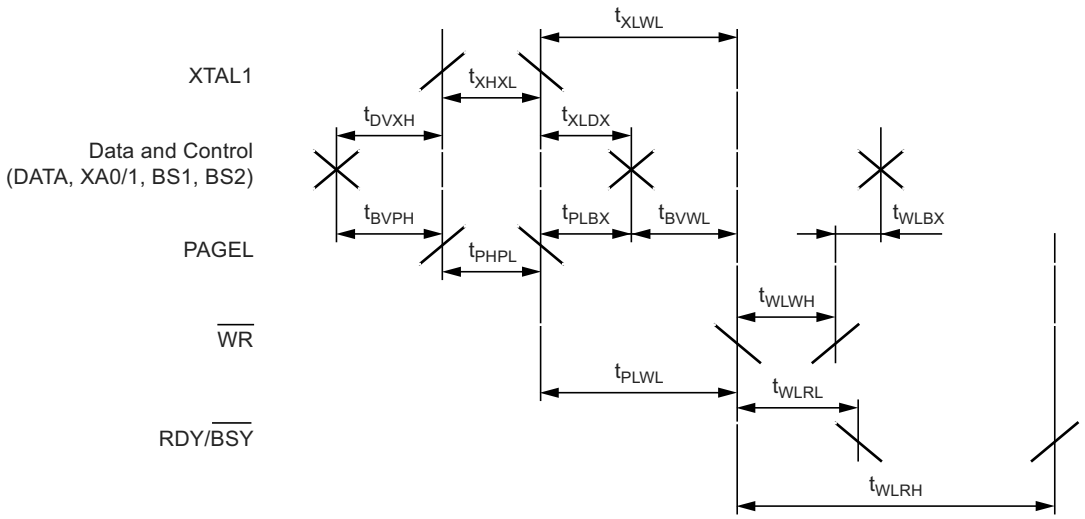
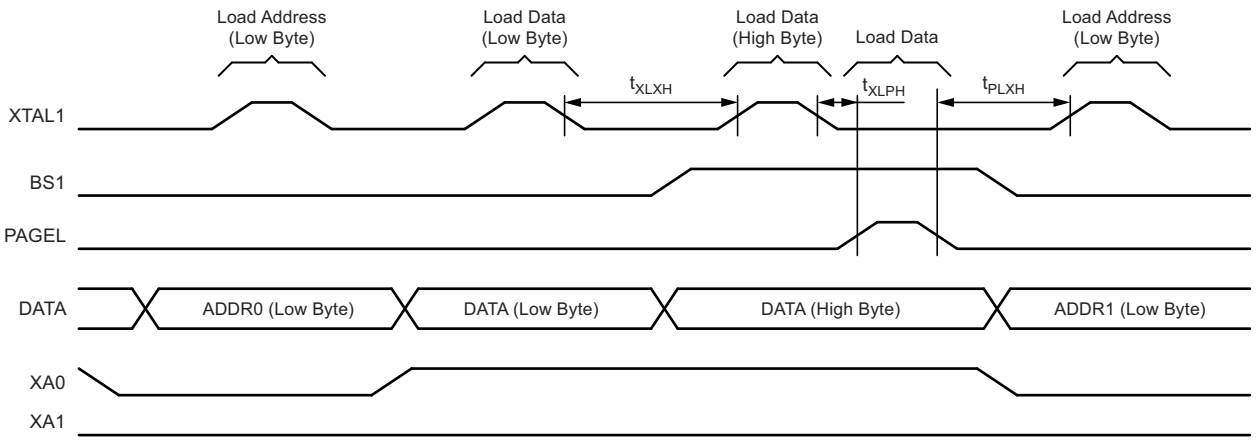


Figure 25-8. Parallel Programming Timing, Loading Sequence with Timing Requirements(1)



Note: 1. The timing requirements shown in Figure 25-7 (i.e., t_{DVXH}, t_{XHXL}, and t_{XLDX}) also apply to loading operation.

26. Electrical Characteristics

All DC/AC 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.

26.1 Absolute Maximum Ratings

Stresses beyond those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Parameters	Min.	Typ.	Max.	Unit
Operating temperature	–40		+125	°C
Storage temperature	–65		+150	°C
Voltage on any pin except $\overline{\text{RESET}}$ with respect to ground	–0.5		V_{CC} + 0.5	V
Voltage on $\overline{\text{RESET}}$ with respect to ground	–0.5		+13	V
Maximum operating voltage			6	V
DC current per I/O pin		40		mA
DC current V_{CC} and GND pins		200		mA
Injection current at V_{CC} = 0V to 5V		±5(1)		mA

Note: 1. Maximum current per port = ±30mA

26.2 DC Characteristics

T_A = –40°C to +125°C, V_{CC} = 2.7V to 5.5V (unless otherwise noted)

Parameter	Condition	Symbol	Min.	Typ.	Max.	Unit
Input low voltage	Port B, C and D and XTAL1, XTAL2 pins as I/O	V_{IL}	–0.5		$0.2V_{CC}$(1)	V
Input high voltage	Port B, C and D and XTAL1, XTAL2 pins as I/O	V_{IH}	$0.6V_{CC}$(2)		V_{CC} + 0.5	V
Input low voltage	XTAL1 pin, external clock Selected	V_{IL1}	–0.5		$0.1V_{CC}$(1)	V
Input high voltage	XTAL1 pin, external clock selected	V_{IH1}	$0.8V_{CC}$(2)		V_{CC} + 0.5	V

Notes:
1. “Max” means the highest value where the pin is guaranteed to be read as low
2. “Min” means the lowest value where the pin is guaranteed to be read as high
3. Although each I/O port can sink more than the test conditions (10mA at V_{CC} = 5V, 6mA at V_{CC} = 3V) under steady state conditions (non-transient), the following must be observed:
   1] The sum of all IOL, for ports B0 - B1, C2 - C3, D4, E1 - E2 should not exceed 70mA.
   2] The sum of all IOL, for ports B6 - B7, C0 - C1, D0 -D3, E0 should not exceed 70mA.
   3] The sum of all IOL, for ports B2 - B5, C4 - C7, D5 - D7 should not exceed 70mA.
   If IOL exceeds the test condition, VOL may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test condition.
4. Although each I/O port can source more than the test conditions (10mA at V_{CC} = 5V, 8mA at V_{CC} = 3V) under steady state conditions (non-transient), the following must be observed:
   1] The sum of all IOH, for ports B0 - B1, C2 - C3, D4, E1 - E2 should not exceed 100mA.
   2] The sum of all IOH, for ports B6 - B7, C0 - C1, D0 -D3, E0 should not exceed 100mA.
   3] The sum of all IOH, for ports B2 - B5, C4 - C7, D5 - D7 should not exceed 100mA.
   If IOH exceeds the test condition, VOH may exceed the related specification. Pins are not guaranteed to source current greater than the listed test condition.
5. Minimum V_{CC} for power-down is 2.5V.
6. The analog comparator Propogation Delay equals 1 comparator clock plus 30nS. See Section 20. “Analog Comparator” on page 225 for comparator clock definition.

4. CRC calculation of diagnostic frames in LIN 2.x.
   Diagnostic frames of LIN 2.x use “classic checksum” calculation. Unfortunately, the setting of the checksum model is enabled when the HEADER is transmitted/received. Usually, in LIN 2.x the LIN/UART controller is initialized to process “enhanced checksums” and a slave task does not know what kind of frame it will work on before checking the ID.
   Problem fix / workaround
   This workaround is to be implemented only in case of transmission/reception of diagnostics frames.
   a. Slave task of master node:
      Before enabling the HEADER, the master must set the appropriate LIN13 bitvalue in LINCR register.
   b. For slaves nodes, the workaround is in 2 parts:
      - Before enabling the RESPONSE, use the following function:

```
      void lin_wa_head(void) {
unsigned char temp;
      temp = LINBTR;
      LINCR =     ;       // It is not a RESET !
      LINBTR = ( <<LDISR)|temp;
      LINCR = ( <<LIN13)|( <<LENA)|( <<LCMD2)|( <<LCMD1)|( <<LCMD0);
      LINDLR =     ;      // If it isn't already done
}
```

      - Once the RESPONSE is received or sent (having RxOK or TxOK as well as LERR), use the following function:

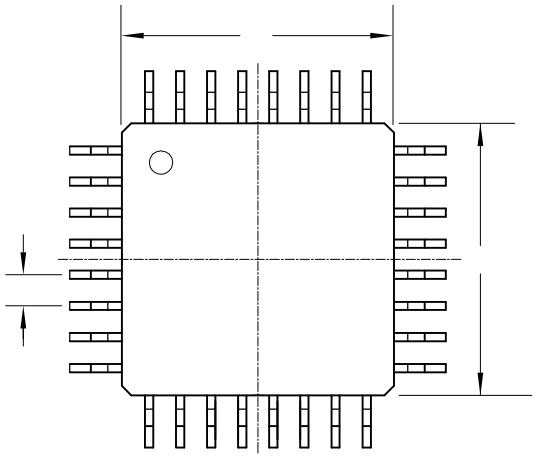
```
void lin_wa_tail(void)  {
      LINCR =     ;       // It is not a RESET !
      LINBTR =     ;
      LINCR = ( <<LIN13)|( <<LENA)|( <<LCMD2)|( <<LCMD1)|( <<LCMD0);
}
```

   The time-out counter is disabled during the RESPONSE when the workaround is set.
5. **Wrong TSOFFSET manufacturing calibration value.**
   Erroneous value of TSOFFSET programmed in signature byte.
   (TSOFFSET was introduced from REVB silicon).
   Problem fix / workaround
   To identify RevB with wrong TSOFFSET value, check device signature byte at address 0X3F if value is not 0X42 (Ascii code ‘B’) then use the following formula.
   TS_OFFSET(True) = (150*(1-TS_GAIN))+TS_OFFSET.
6. **PD0-PD3 set to outputs and PD4 pulled down following power-on with external reset active.**
   At power-on with the external reset signal active the four I/O lines PD0-PD3 may be forced into an output state. Normally these lines should be in an input state. PD4 may be pulled down with internal 220kΩ resistor. Following release of the reset line (whatever is the startup time) with the clock running the I/Os PD0-PD4 will adopt their intended input state.
   Problem fix / workaround
   None
7. **LIN Break Delimitter**
   In SLAVE MODE, a BREAK field detection error can occur under following conditions. The problem occurs if 2 conditions occur simultaneously:
   a. The DOMINANT part of the BREAK is (N+0.5)*Tbit long with N=13, 14,15, ...
   b. The RECESSIVE part of the BREAK (BREAK DELIMITER) is equal to 1*Tbit. (see note below)

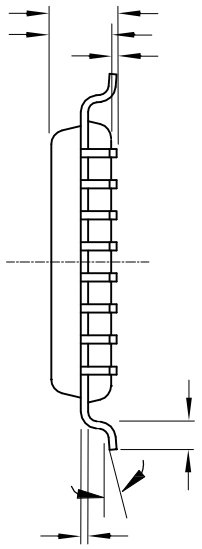
   The BREAK_high is not detected, and the 2nd bit of the SYNC field is interpreted as the BREAK DELIMITER. The error is detected as a framing error on the first bits of the PID or on subsequent Data or a Checksum error.
   There is no error if BREAK_high is greater than 1*Tbit + 18%. There is no problem in Master mode.

Note: LIN2.1 Protocol Specification paragraph 2.3.1.1 Break field says: “A break field is always generated by the master task(in the master node) and it shall be at least 13 nominal bit times of dominant value, followed by a break delimiter, as shown in Figure 30-1 on page 308. **The break delimiter shall be at least one nominal bit time long.**”

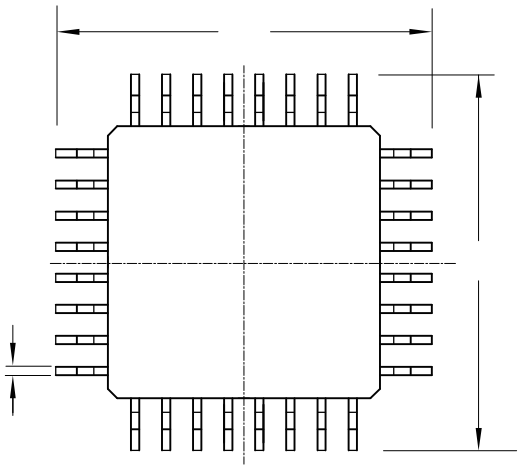
Figure 32-1. MA



Top View



Side View



Bottom View

COMMON DIMENSIONS (Unit of Measure = mm)				
Symbol	**MIN**	**NOM**	**MAX**	**NOTE**
A	----	----	1.20	
A1	0.05	----	0.15	
A2	0.95	1.00	1.05	
D/E	8.75	9.00	9.25	
D1/E1	6.90	7.00	7.10	2
C	0.09	----	0.20	
L	0.45	----	0.75	
b	0.30	----	0.45	
e	0.80 TYP.			
n	32			

02/29/12

Atmel Package Drawing Contact: packagedrawings@atmel.com	TITLE	GPC	DRAWING NO.	REV.
	MA, 32 Lds - 0.80mm Pitch, 7x7x1.00mm Body size Thin Profile Plastic Quad Flat Package (TQFP)	AUT	MA	C